IQUERY: A Trustworthy and Scalable Blockchain Analytics Platform

Lingling Lu, Zhenyu Wen, *Member, IEEE*, Ye Yuan, Binru Dai, Peng Qian, Changting Lin, Qinming He, *Member, IEEE*, Zhenguang Liu, Jianhai Chen, *Member, IEEE*, Rajiv Ranjan, *Senior Member, IEEE*

Abstract—Blockchain, a distributed and shared ledger, provides a credible and transparent solution to increase application auditability by querying the immutable records written in the ledger. Unfortunately, existing query APIs offered by the blockchain are inflexible and unscalable. Some studies propose off-chain solutions to provide more flexible and scalable query services. However, the query service providers (SPs) may deliver fake results without executing the real computation tasks and collude to cheat users. In this paper, we propose a novel intelligent blockchain analytics platform termed IQUERY, in which we design a game theory based smart contract to ensure the trustworthiness of the query results at a reasonable monetary cost. Furthermore, the contract introduces the second opinion game that employs a randomized SP selection approach coupled with non-ordered asynchronous querying primitive to prevent collusion. We achieve a fixed price equilibrium, destroy the economic foundation of collusion, and can incentivize all rational SPs to act diligently with proper financial rewards. In particular, IQUERY can flexibly support semantic and analytical queries for generic consortium or public blockchains, achieving query scalability to massive blockchain data. Extensive experimental evaluations show that IQUERY is significantly faster than state-of-the-art systems. Specifically, in terms of the conditional, analytical, and multi-origin query semantics, IQUERY is 2 ×, 7 ×, and 1.5 × faster than advanced blockchain and blockchain databases. Meanwhile, to guarantee 100% trustworthiness, only two copies of query results need to be verified in IQUERY, while IQUERY's latency is $2 \sim 134$ × smaller than the state-of-the-art systems.

Index Terms—Blockchain, second opinion smart contract, game theory, data analytics, query platform

1 INTRODUCTION

BLOCKCHAIN revolutionizes the traditional centralized application mode and overcomes the trust challenges in multiple application scenarios [1], [2], [3]. For example, once a payment system writes user payment behaviors into a blockchain network, any authorized user is capable of querying the immutable records in the blockchain to achieve accountability and transparency of transactions. Nowadays, employing a smart contract to query ledger data from bookkeeping nodes is a general method for blockchain data query. However, existing query methods struggle to search for accurate and comprehensive ledger data due to their lack of flexibility and scalability [4], [5].

Prior works [6], [7], [8] concentrated on improving the analysis capability of bookkeeping nodes for querying ledger data. Yet, these works have difficulties in providing rich semantics for blockchain data query and yielding poor performance in handling large-scale data [9], [10]. For example, Nathan et al. [7] leverage the serializable snapshot isolation (SSI) to design a blockchain relational database for permissioned blockchain query. Cai et al. [11] propose a SGX-based public blockchain query framework that supports single keyword and boolean queries. Nevertheless, such blockchain relational databases and query frameworks suffer from the **flexibility issue** due to a lack of fine-grained query semantics. Additionally, they are ungeneralized, designed explicitly for consortium or public blockchain, and therefore cannot adapt to flexible application scenarios.

Furthermore, SEBDB [8] optimizes the SQL-like query semantics with novel indices, while LineageChain [6] enables provenance queries by storing provenance in a Merkle DAG structure. Unfortunately, SEBDB and LineageChain still suffer from the **scalability issue** and struggle to deal with the query demand of massive application data. Although these efforts enrich blockchain query semantics, the query service is tightly coupled with the blockchain transaction system. Executing too many query transactions at the blockchain layer will result in consensus performance degradation and lower transaction throughput for bookkeeping.

After scrutinizing the existing related works [11], [12], [13], we found that the rich semantic query service in bookkeeping nodes can be extracted for supporting the ledger data query. Heuristically, we consider decoupling the rich semantic query service from bookkeeping nodes into a blockchain query layer, expecting to make a breakthrough in the technology of blockchain data query. Towards the goal of solving the **flexibility** and **scalability** issues mentioned above, we further identify some key requirements to be ful-

L. Lu, P. Qian, Q. He, Z. Liu, J. Chen are with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, E-mail: lulingling@email.cufe.edu.cn, pqian@zju.edu.cn, hqm@zju.edu.cn, liuzhenguang2008@gmail.com, chenjh919@zju.edu.cn

Z. Wen is the corresponding author and is with the Institute of Cyberspace Security, Zhejiang University of Technology, China, E-mail: zhenyuwen@zjut.edu.cn

Y. Yuan is with Beijing Institute of Technology, China, E-mail: yuanye@mail.neu.edu.cn

B. Dai is with the School of Economics, Zhejiang University, China, E-mail: ddbr1997@163.com

C. Lin is with Binjiang Institute of Zhejiang University, China, E-mail: linchangting@gmail.com

R. Ranjan is with Newcastle University, United Kingdom, E-mail: Rajiv.Ranjan@newcastle.ac.uk

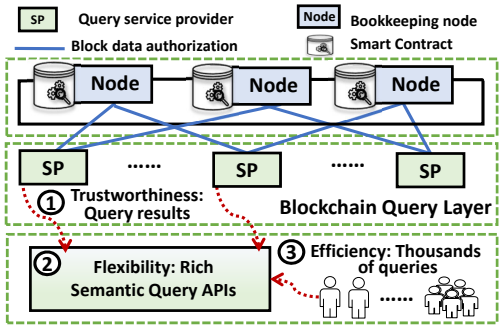


Fig. 1. Trustworthiness, flexibility and efficiency requirements for IQUERY.

filled when designing a decoupled blockchain query layer.

Trustworthiness of Query (R1): We first need to guarantee the trustworthiness of queries when designing a **scalable and decoupled** blockchain query layer. Specifically, suppose a user doubts the transactions on the blockchain; the user can verify them by submitting a query request to an authorized query service provider (SP), who listens to bookkeeping nodes (Fig. 1). SP will receive rewards from the user for providing the query service requested by the user. Unfortunately, the integrity of some blockchain query service providers is relatively low. In order to maximize their benefits, some SPs may deliver fake information instead of making efforts to query the blockchain data. Thus, to ensure the trustworthiness of queries, it is best for users to send the same requests to multiple SPs and verify the consistency of their returned results. If users only choose a limited number of SPs, some providers may conspire together to provide fake information. Existing solutions [14], [15] overcome this issue by sending query requests to all providers, which is costly and impractical. Therefore, a scalable and economical solution for ensuring query trustworthiness is much coveted.

Flexibility of Query (R2): We propose to build a query layer that flexibly supports running generic, user-defined consortium or public blockchain. Additionally, the query service needs to support queries with sufficient semantics to address the **flexibility** issue. For instance, the fine-grained query semantic GetPayment($in(price:[$num_s$, num_e])) gets all payments within a price window, and the analytical query semantic GetTotal($eq($userId :?)) gets the total payment value of a particular user.

Efficiency of Query (R3): A more efficient query service is desired for **both** flexibility and scalability issues because a real-time analysis is essential for blockchain query services. However, according to the prior research [10], Fabric can only handle less than 200 queries per second. Ethereum transactions have a gas limit and cannot perform complex queries in one transaction [3]. Consequently, consortium and public blockchain both fail to support a substantial number of application users simultaneously performing thousands of queries to access the blockchain transactions [5].

In this paper, we propose IQUERY, a truly distributed blockchain query platform. To guarantee the trustworthiness of queries (**R1**), we design a game theory based method that employs a *second opinion* mechanism to ensure the correctness of query results while minimizing the number of query SPs. To tackle the inflexibility of query (**R2**), we introduce a set of rich semantic query APIs that utilize logical operators, condition parameters and Select-Aggregate functions to answer complex queries in off-chain databases. Finally, we maintain these databases that extract key information from blocks to improve query efficiency (**R3**).

According to Bernoulli analysis, when the proportion of honest SPs in the network is 80%, our second opinion mechanism only needs to compare **two** copies of query results to ensure 100% trustworthiness. In contrast, state-of-the-art systems require **four** copies of query results to achieve query confidence of 95%. Furthermore, empirical results show that the latency of IQUERY is at millisecond level and 2 ~ 134 × smaller than the state-of-the-art methods. In summary, the contributions are:

- We propose a decentralized and scalable blockchain analytics platform IQUERY. Specifically, rich query semantics, including conditional, analytical, and multi-origin query methods, are introduced to offer flexible, efficient, and verifiable query services for generic types of blockchain users (§3 and §4).
- We design a second opinion smart contract to minimize the number of query SPs while guaranteeing the trustworthiness of query results. In particular, the game theory based method that combines incentive mechanisms and auditable smart contract is formally verified (§5).
- We categorize SPs into three different types with various abilities matching realistic scenarios and present corresponding countermeasures to prevent collusion in IQUERY. Specifically, the randomized SP selection approach coupled with a non-ordered asynchronous querying primitive makes it impossible for the first colluder to infer who the user will choose next, thus motivating rational SPs to be honest. Additionally, the self-interest hypothesis can easily prove the impracticality of SPs colluding together and the reputation mechanism removes malicious irrational SPs from the system (§5.3).
- Extensive experiments demonstrate the effectiveness and performance advantages of IQUERY compared with state-of-the-art systems (§6). IQUERY is released on Github for public use[1].

The rest of the paper is organized as follows. In section 2, we give a brief introduction of blockchain data query and game theory models. Then, we describe the design and architecture of IQUERY in section 3, followed by query methods with rich semantics in section 4. In section 5, we give the details of how the second opinion smart contract guarantees the trustworthiness in IQUERY. In section 6, we conduct extensive experiments to evaluate the performance of IQUERY. Finally, we conclude our paper in section 7.

2 Related Work

2.1 Blockchain Data Query

Blockchain. Ethereum [16] is the first public blockchain to introduce the smart contract. Users can read and

1. https://github.com/lulinglingcufe/iQuery

TABLE 1
Comparison of Query in Blockchain Systems

Catalog	Representative Systems	Decoupled Query	Semantic Query	Analytical Query	Authenticated Query
Blockchain	Fabric [5], Ethereum [16], FISCO BCOS [17]	×	Weak	Weak	×
Blockchain Database	LineageChain [6], Blockchain relational database [7], SEBDB [8], FalconDB [18]	×	Strong	Weak	Blockchain relational database: ×, Others: ✓
Public Blockchain Query Layer	Etherscan [19], EthScope [20], Blocksci [21], EtherQL [22], vChain [2], GEM^2 [23] VQL [24], SGX-based query framework [11]	✓	Etherscan, Blocksci EtherQL: Strong Others: Weak	Etherscan, Blocksci: Strong Others: Weak	SGX-based query framework, vChain, GEM^2, VQL:✓ Others:×
Generic Blockchain Query Layer	IQUERY	✓	Strong	Strong	Flexible and efficient methods to ensure trustworthiness

write blockchain data via the smart contract. Consortium blockchains, e.g., Fabric [5] and FISCO BCOS [17], achieve better performance while meeting security requirements in commercial scenarios. Nevertheless, these popular blockchain systems lack rich query semantics.

Blockchain Database. Some works leverage rich features in relational databases to enhance blockchain systems, thus directly handling query requests. LineageChain [6] enables fine-grained and secure provenance queries in the smart contract via Merkle DAG. Blockchain relational database [7] and SEBDB [8] apply SQL-like language to access data. FalconDB [18] and SEBDB employ authenticated data structures (ADS) to ensure the correctness of queries. Unfortunately, these databases lack optimization methods for blockchain data analysis.

Public Blockchain Query Layer. Some works leverage the query layer decoupled from the blockchain transaction system to improve query efficiency. Etherscan [19] is a widely-used Ethereum explorer. EthScope [20] is an efficient query framework for Ethereum attack replays. However, they do not support authenticated queries. The SGX-based query framework [11] relies on trusted hardware to deliver authenticated queries, while other works depend on ADS to achieve authenticated queries. Besides, these works only support public blockchain, lacking a solution for consortium blockchain. These works have two limitations: (1) high cost for users to verify the integrity of query transactions and (2) limited query semantics. Specifically, vChain [2] and GEM^2 [23] change the block structure, construct ADS, embed them into each block and use off-chain SPs to answer queries. Despite making progress in query throughput, the cost for users to cryptographically verify the query results is excessive. Precisely, the latency of users verifying query results in vChain and GEM^2 is at the *second level*. In contrast, the latency of users verifying query results in IQUERY is at the *microsecond level*. VQL [24] requires miners to verify the query layer's data consistency, which significantly takes up miners' computing power. Furthermore, authenticated query semantics are *limited to* boolean range queries and time-window queries in vChain, GEM^2 and VQL. Compared with existing works [2], [11], [23], IQUERY offers a game theory based authenticated query method that improves the trustworthiness of *any* query semantics.

Comparison. Table 1 compares blockchain systems, blockchain databases, public blockchain query layer and IQUERY. Among these state-of-the-art systems, IQUERY is the first off-chain query platform that achieves semantic, analytic and authenticated query methods, significantly improving trustworthiness, flexibility and efficiency for querying generic types of blockchains.

2.2 Game Theory Models in Trustworthy Queries

Many works use game theory in replication based verifiable computation. We compare the second opinion model with other game theory models used in trustworthy queries regarding (1) threat models (the number of chosen SPs), (2) querying primitive, (3) monetary cost, and (4) protocol implementation.

The Prisoner's Dilemma Game. Dong et al. [14] proposed a contract that induces the Prisoner's Dilemma game between two SPs. The contract employs synchronized querying primitive, thus giving dishonest SPs a chance to collude before returning query results. To prevent collusion, the incentive mechanism in the contract encourages colluders to report collusion to the client user. If another SP does cheat, the reporting SP will get another SP's deposit.

The Multi-Player Outsourcing Game. Alptekin [25] proposed a multi-player outsourced computation game to ensure the outsourced job is computed correctly. The game motivates rational SPs to be honest and enforces a bound on the damage malicious SPs can cause. We remove malicious irrational SPs from the IQUERY pool via the reputation mechanism in the second opinion model.

Comparison. (1) Threat Model: Both [14] and IQUERY present a two-player game aiming at minimizing the number of SPs while guaranteeing the trustworthiness of query results. [25] proposed a multi-player game. The multi-player threat model is less efficient than IQUERY in the blockchain query scenarios. Because the cost of querying multiple SPs is a great economic burden for users. (2) Querying Primitive: Because [14] and [25] employ synchronized querying primitive, rewards and fines are required to prevent collusion. The second opinion model employs non-ordered asynchronous querying primitive, making it impossible for the first SP to infer who the user will choose next and collude. (3) Monetary Cost: The second opinion contract

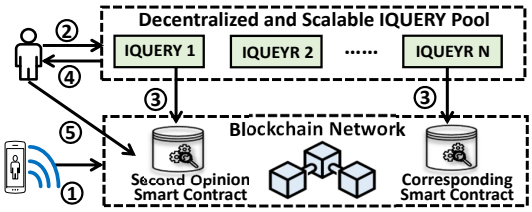


Fig. 2. The life-cycle for querying ledger data via IQUERY.

does not require deposits from SPs, making it a more attractive design for SPs who offer services for thousands of users. In this situation, the deposits will be a prohibitively high financial cost for SPs. (4) Protocol Implementation: [25] relies on a trusted centralized bank to execute the payment protocol, while IQUERY and [14] implement the protocol in the smart contract for self-enforcing payment.

3 Design and Architecture of IQUERY

In this section, we describe the overview of how users get trustworthy query results form IQUERYS, followed by the main components of the IQUERY system.

3.1 System Overview

Fig. 2 illustrates the lifecycle of querying ledger data via IQUERY. Each IQUERY has a SP to offer blockchain query service and we design the blockchain analytics platform as a decentralized and scalable IQUERY pool. First, we write transactions into blockchain (①, more details in §3.2). Then, users select an IQUERY from the decentralized IQUERY pool. IQUERY is able to answer complex queries with rich query APIs (②). Next, in order to ensure the trustworthiness of query results, SP utilizes a contract manager for two types of smart contracts (③). (i) if users want to query the ledger directly, the contract manager will employ APIs of the *corresponding smart contract* to execute simple queries on-chain and retrieve trustworthy results, (ii) users utilize the rich query APIs to perform fine-grained query semantics off-chain while executing the game theory based *second opinion smart contract* to ensure trustworthiness. Afterwards, IQUERY returns query results with its attached signature (④). Finally, if users confirm the correctness of query results, the *second opinion smart contract* will give the reward honest IQUERYS (⑤).

3.2 Write Transactions to Blockchain

We prepare transactions in advance for queries by writing transactions to blockchain. Note that IQUERY can flexibly support both consortium and public blockchain. Therefore, We adopt Fabric [5], the most widely used consortium blockchain platform, and Ethereum [16], the largest decentralized application platform, to write payment transaction records in step ① of system overview in §3.1. Besides, the payment transactions are also example transactions for queries in §4 and datasets in §6.1.

Payment transactions. Table 2 shows the transactions in a simple payment scenario. First, a user initializes his/her blockchain identity by submitting $tx_{initUser}$. There are two

TABLE 2
Examples of Payment Transactions Used in §4 (Queries) and §6.1 (Dataset)

Transaction Type	Correlated User Identity
initUser (T_1)	user@company.cn user@supermarket.com
initCommodity (T_2)	user@company.cn
rechargeToken (T_3)	user@supermarket.com
commodityConsumption (T_4)	user@supermarket.com

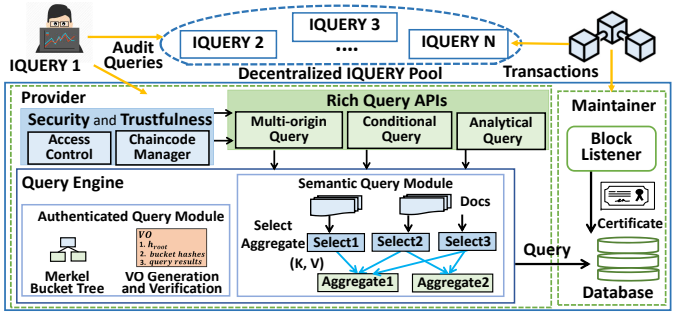


Fig. 3. Architecture of IQUERY.

types of users: company (*user@company.cn*) and supermarket (*user@supermarket.com*). If the user belongs to the company organization, the user will have a unique id in the blockchain generated from *user@company.cn*. Secondly, this user submits $tx_{initCommodity}$ to init commodities with price and id. Then, the commodities attached with RFID tags and provided by the company user are shipped to the supermarket for sale. Next, a user identified as *user@supermarket.com* submits $tx_{rechargeToken}$ to recharge his/her smart card. Finally, the user of the supermarket applies a smart card to buy these commodities through $tx_{commodity\ Consumption}$.

Blockchain consensus. Fabric uses a practical byzantine fault tolerance (PBFT) consensus [5], and Ethereum employs proof of work (POW) or proof of stake (POS) consensus [16]. Notably, the blockchain transaction system consists of the application, execution, data, and consensus layers [9]. We implement a game theory based query protocol in the second opinion smart contract, incentivizing IQUERY to deliver honest query services. Regardless of how we implement the application logic in a smart contract, the underlying consensus layer guarantees the secure execution of smart contract transactions, ensuring that the IQUERY analytics platform is trustworthy.

3.3 Query Service Provider

The decentralized IQUERY pool in Fig. 2 is composed of a set of IQUERY entities for parallel query processing. To be precise, Fig. 3 illustrates that each IQUERY entity consists of a *query service provider* and an *off-chain database maintainer*.

The audit query requests are first handled by a provider as shown in Fig. 3. The provider guarantees secure and trustworthy query services through *Access Control* and *Contract Manager*. The Access Control defines users' query permissions to protect the privacy of our blockchain analytics platform [21], [26]. For example, an ordinary user can only query transactions related to him/herself, but a VIP user such as a bank can query its customers' transactions. The second opinion smart contract is a game theory based

mechanism that incentivizes each IQUERY to deliver query services and ensure the trustworthiness of query results (see §5).

The requests that pass the security and trustworthiness check can call rich query APIs. These APIs provide users with flexible query methods including conditional, analytical and multi-origin queries. A query engine is responsible for improving query efficiency and supporting rich query semantics. Specifically, Select-Aggregate functions support conditional query (see §4.1) and analytical query (see §4.2). In order to support results verification among multiple SPs, Merkle Bucket Tree based authenticated query module supports multi-origin query (see §4.3).

3.4 Off-Chain Database Maintainer

There are two reasons for maintaining an off-chain database: 1) the inflexibility of blockchain database affects the query efficiency (e.g., attributes in the value field cannot be indexed in blockchain $StateDB_{peer}$ [9]), 2) it is a significant burden for the transaction processing lifecycle to directly perform queries via the smart contract [5], [24]. Thus, we design an off-chain database maintainer that reads ledger data using a valid certificate or public key in the blockchain network.

Maintain Off-Chain Database. IQUERY maintains two *local* data-bases, i.e., $StateDB_{iQuery}$ and $HistoryDB_{iQuery}$ that extract key information from blockchain network to improve query efficiency. IQUERY obtains transactions from the blockchain network by a *Blocklistener*. In particular, it registers a $BlockEvent(\,)$ SDK API [27] to read *raw block data* from a peer node and inserts the update (i.e., $State_{\triangle}$) to local databases.

4 QUERY METHODS WITH RICH SEMANTICS

In this section, we use the **payment transactions** as an example to illustrate the technical details of rich query APIs including conditional, analytical and multi-origin query methods.

4.1 Conditional Query Method

IQUERY improves query flexibility with general, practical and sufficient condition parameters [28], [29], [30] for target records queries.

In conditional queries, users can add additional constraints in the query selector. $q = \langle op\{c_1, c_2 ... c_n\}, db\rangle$ is an example of a conditional query, where op is the *logical operator*, and c_n is the condition parameter. The logical operators include $\$and$, $\$or$, $\$eq$ and $\$gt$, etc.

Temporal Query is a special case of conditional query which allows users to query records within a certain time period, i.e., $q = \langle [t_{start}, t_{end}], db\rangle$. With the increasing block number, we formalize the temporal query as: $q = \langle [b_{start}, b_{end}], db\rangle$. $[t_{start}, t_{end}]$ is the time interval parameters (e.g., $t_{start} \leq record.timestamp \leq t_{end}$) and $[b_{start}, b_{end}]$ is the block interval parameter (e.g., $b_{start} \leq record.num_{block} \leq b_{end}$). Note that db is the target database table applied with indexes to improve query efficiency.

We use conditional query Q_1 to illustrate how IQUERY improves query performance with correlated read set.

Example 1: Q_1 is a join query that searches the complete $tx_{commodityConsumption}$ with detailed commodity and user information. To perform the query, client users send a logical query selector in the form of Q_1 but specifies commodityId to IQUERY. Then it will search the $db_{ConsumptionJoinCommodity}$ table in $HistoryDB_{\text{IQUERY}}$ with a customized index to improve query speed and return qualified records.

$$Q_1 : \langle \$eq(Commodity_{id} :?), db_{ConsumptionJoinCommodity}\rangle$$

Although, $tx_{commodityConsumption}$ in Table 2 only records commodityID and userID instead of the detailed commodity and user information. To improve the performance of Q_1, when updating IQUERY databases, we first identify the correlated read set [31] $rs = \{(key_{commodityID}, version), (key_{userID}, version)\}$ in $tx_{commodityConsumption}$. We assign the former element in rs k_1 and the latter k_2. Then we search the value of k_1 and k_2 from $State_{\text{IQUERY}}$. The response $(key_{commodityID}, value)$ and $(key_{userID}, value)$ are combined together to form a more user-readable record. Besides, IQUERY also attaches the basic information of transactions, i.e., timestamp, Txid, $block_{num}$ and tx_{num}. The record will be inserted into $db_{ConsumptionJoinCommodity}$ of $HistoryDB_{\text{IQUERY}}$.

4.2 Analytical Query Method

In analytical queries, users can send logical query selectors to IQUERY for elaborate blockchain data analysis. Moreover, IQUERY enhances the performance of ledger data analytics by using Select-Aggregate style query functionalities.

Select Records to View. In the selection stage, IQUERY uses a customized $Select(\cdot)$ function to construct the view in CouchDB [32]. The record selection consists of the following steps. ① $View_{data} = \{\sum_{i=1}^{n}(k_i, v_i) | (k_i, v_i) \in Select(doc_{data})\}$: Let data view be an ordered collection of key value pairs, where $Select(doc_{record})$ function generates each (k,v). ② $Select(doc_{record}) \rightarrow (key_{attribute1}, value_{attribute2})$: $Select(\cdot)$ function has a user-defined selection logic. It scans all the records in a database and returns a key-value pair where the key and value are attributes of the qualified records selected by the selection logic.

Aggregate View by Feature. In the aggregation stage, IQUERY analyzes the blockchain data characteristics by calculating selected records' features from aggregated $View_{data}$. $Aggregate(keys, values) \rightarrow \{\sum_{i=1}^{n} feature_i\}$: $Aggregate$ function takes selected (k,v) pairs in $View_{data}$, and outputs the features of these pairs.

We use analytical query Q_2 to illustrate how IQUERY integrates Select-Aggregate style functionalities.

Example 2: Q_2 is an analytical query that analyzes the total value of the recharge token in a specific time period $[t_{start}, t_{end}]$. To perform the query, client users send a logical query selector in the form of Q_2 but specifies the time period to IQUERY.

$$Q_2 : GetTotal(\$in([t_{start}, t_{end}]), db_{rechargeToken})$$

To answer Q_2, IQUERY forms the $View_{rechargeToken}$ based on $db_{rechargeToken}$ in the selection stage. The Select function filters all valid transactions and sets the *timestamp*

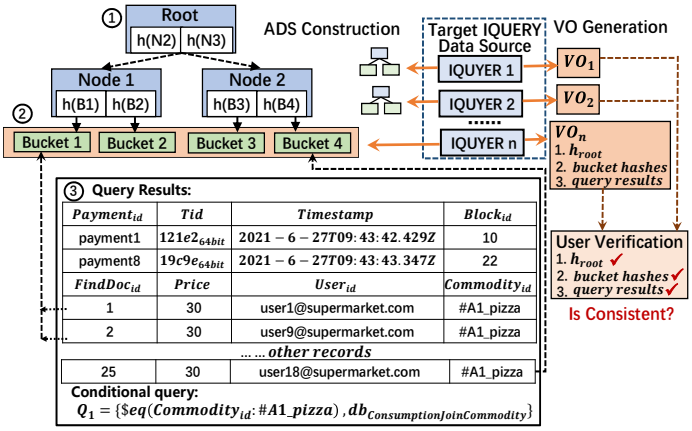


Fig. 4. Target IQUERY$_n$ constructs a MBT-based ADS from rich query results and generates VO_n.

of a record as key and *recharge token* as value. When IQUERY receives Q_2 request, it forms a time window $[t_{start}, t_{end}]$ to select the inputs for Aggregate function from $View_{commodity}$. Finally, all token values are added by $Sum(\cdot)$ to obtain total values in the aggregation stage.

4.3 Multi-Origin Query Method

We design a multi-origin query method, allowing a client user to validate whether multiple IQUERYS **have answered consistently**. The multi-origin query method returns a set of data attached with a cryptographic proof, known as a verification object (VO), for client users to verify the *completeness* and *soundness* of query results from the target IQUERY data source. *Example 3* indicates the usage of VO.

ADS Construction. IQUERY applies Merkle Bucket Tree (MBT) for ADS constructions. The sequentiality of blocks and transactions contributes to the identical ADS compared in two distinct IQUERY data sources. We define the buckets in MBT as follows.

Definition 1 (Records in a bucket). The query results of Q $\langle op\{c_1, c_2...c_n\}, db\rangle$ are a set of records. The serial number of a record in the query results is $FindDoc_{id}$ and it is computed through Eq.1. We place these records in a set of buckets to construct MBT, and the capacity of a bucket is denoted as cap. The serial number of a bucket is $bucket_{id}$ and the position of the records in the bucket is po.

$$FindDoc_{id} = cap \times (bucket_{id} - 1) + po \quad (1)$$

After all records in the query results are placed in the buckets, hash buckets are computed and served as leaf nodes of a MBT. Then MBT is constructed from bottom to up. Moreover, h_{root} is the root of a MBT and the internal nodes are formed by the cryptographic hashes computed from their intermediate children. Besides, the children of an internal node are called *fanout* [33].

VO Generation. After constructing the ADS, IQUERY can generate VO from ADS. As shown in Fig. 4, VO consists of (1) h_{root}, (2) the bucket hashes in the MBT, and (3) the ordered query results set. A user can (i) recompute the bucket hashes based on the query results set to ensure the *completeness* of the query, (ii) compare the h_{root} from distinct IQUERY to check the *soundness* of query results, because h_{root} in VO is the digest of ADS.

Example 3: Q_3 is a multi-origin query that sends Q to multiple IQUERY data sources in the decentralized IQUERY pool, e.g., IQUERY$_1$, IQUERY$_2$, ..., IQUERY$_n$. A user can decide to send Q to how many IQUERYS, which defines the n in Q_3. In this example, we set Q to Q_1, a conditional join query defined in §4.1. Then a client user sends the multi-origin query to target IQUERYS.

$$Q_3 : \langle \text{IQUERY}_n, Q\rangle, n = 1, 2, 3...$$

In the IQUERY$_n$ server side, Fig. 4 shows how target IQUERY$_n$ constructs a MBT-based ADS from the rich query result of Q_1, then generates VO_n. The fanout is 2, and the capacity is 8. In *Example 3*, IQUERY$_n$ receives a multi-origin query requests from the client user and finds that the query result of Q_1 contains 25 query records as shown in *Query Results* (③) of Fig. 4. A record includes many fields: $Payment_{id}$, Txid, Timestamp, $Block_{id}$, Price, $User_{id}$ and $Commodity_{id}$, etc. More importantly, the returned doc has a $FindDoc_{id}$ which identifies the serial number of the doc in the result set. IQUERY$_n$ uses the $FindDoc_{id}$ to compute which bucket to place the record. We take the last record in the query results with the $FindDoc_{id}$ 25 as an example. According to Eq. 1, $25 = 8 \times (4 - 1) + 1$, so this record should be put in the 1st position of the 4th bucket. When all the records are placed in the bucket. IQUERY$_n$ computes hash buckets and can construct the MBT from bottom to up and establish ADS. IQUERY$_n$ then delivers VO_n to the client user.

In the client user side, we show how to verify query results with VO_n. We assume client users receive VO_n from IQUERY$_n$. If h_{root} in VO_1 and h_{root} in VO_2 are the same, the user can ensure the *soundness* of query results. Because the IQUERYS from distinct query origins return the same proof of query integrity to prove the query results are consistent.

5 The Second Opinion Smart Contract

We design a game theory based method to minimize the number of query SPs and reduce the monetary cost of users while guaranteeing the trustworthiness of query results. The proposed query protocol is implemented in an auditable self-enforcing blockchain smart contract and destroys the economic foundation of SP collusion.

5.1 Threat Model

IQUERY receives rewards from the user for providing correct query results requested by the user, but performing query computations incurs cost. There is a potential risk of IQUERY giving incomplete or fake results without executing real queries. To ensure the trustworthiness of query results, we can query multiple IQUERYS. If the majority of query results are consistent, we assume the result is correct. Unfortunately, users cannot query all available SPs to obtain the majority of query results due to excessive financial cost. Alternatively, users can sample some IQUERYS to obtain the required query results. In such circumstances, some IQUERYS may collude and deliver fake results to save cost as shown in Fig. 5. Therefore, we need a new query protocal to ensure the trustworthiness of query results while minimizing the number of IQUERY.

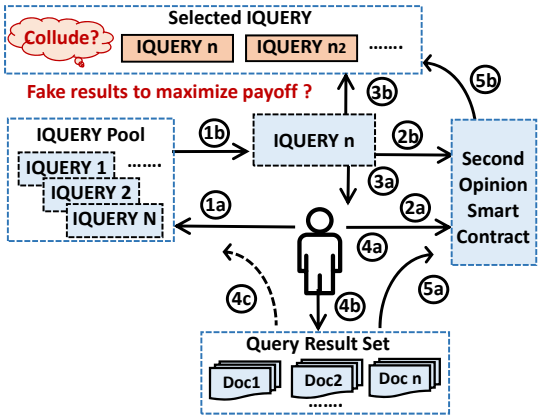


Fig. 5. Protocol of the user and IQUERYS using the second opinion smart contract to complete the query procedure

We also make the following assumptions.

- Rational IQUERY always acts in a way that maximizes its benefit (reward).
- We do not consider the incorrect query results caused by system failures, i.e., if an IQUERY is an honest provider, it should make the best efforts to deliver the correct query results. Otherwise, it is a conspirator.
- An IQUERY will choose a high or low effort when computing the query results. A high effort will always lead to a correct result. A low effort will lead to a wrong result.
- If two IQUERYS return consistent query results, we consider other IQUERYS who give inconsistent results are cheating users.

5.2 Protocol of Using the Contract for Queries

We implement the mechanism of second opinion in the smart contract [1], [2], [24], achieving a protocol between users and IQUERYS to complete the query procedure. Based on our design, users only need to control incentive monetary parameters while selecting two IQUERYS for queries and then compares their query results to ensure trustworthiness. Moreover, the incentive mechanism encourages IQUERY to deliver correct results to maximize its benefits.

Monetary Variables. c is the cost for an IQUERY to compute a query request. d is the amount that a user agrees to pay to IQUERY for computing the query task. A reward p is sent to IQUERYS who make consistent query results. v is user utility when the user gets a correct query result.

Fig. 5 illustrates how the user and IQUERYS use the second opinion smart contract to complete the query procedure. We present the contract and the protocol below.

① First a user searches the IQUERY pool (1a) and IQUERY_n is selected to provide query service (1b).

② The second opinion contract should be signed between the user and the selected IQUERY_n. The user registers a service order O_{query} and sets the incentive parameters d and p in the smart contract (2a). d and p are calculated based on Theorem 2 (see §5.4). If IQUERY_n agrees with the incentive parameters and signs the contract, IQUERY_n sets the id of the query request order O_{query} in the contract. The content of O_{query} will not be recorded in the contract (2b).

③ Next, IQUERY_n chooses to make a high or low effort to compute user's query result (3a), IQUERY_n is later placed in the selected IQUERY list (3b).

④ Upon receiving the query result from IQUERY_n, the user pays IQUERY_n the agreed amount d via the contract (4a). Note that the query result is attached with IQUERY_n's signature for the message, so IQUERY_n cannot refuse to acknowledge that this result has been sent to the user. Then the user puts the returned documents doc_n in the query result set, if there exist two identical query results, the user goes to step ⑤ (4b). Otherwise, the user repeats step ①, i.e., the user continues to searching for another query result until the user finds two consistent results (4c).

⑤ In the game theory analysis of the second opinions model, an incentive parameter p satisfying Theorem 2 will motivate IQUERYS to deliver honest query results to maximize their payoffs (see §5.4). Therefore, upon receiving two identical query results from distinct IQUERYS (5a) in non-ordered asynchronous queries, the user considers the query results correct. By using the contract, the user rewards the two IQUERYS each $\frac{p}{2}$ for being honest (5b).

5.3 Countermeasures to Prevent Collusion

Instead of forbidding or preventing collusion through cryptography, we destroy the economic foundation of collusion [14]. Using financial incentives is a reasonable solution for preventing collusion and related works haven been studied in economics for many years [34]. We categorize SPs into three representative types with various abilities matching realistic scenarios and discuss how the second opinion mechanism motivates SPs to stay honest and follow the protocol in the contract.

(1) A single rational SP. It cannot identify which IQUERY to collude with because the user can randomly choose another IQUERY. In order to maximize its payoff, the IQUERY will choose to be honest according to the analysis in §5.4.

(2) A set of colluding rational SPs. These colluders exchange information to see if they have received the same query. Once anyone of them receives a request, it can verify if other colluding fellows have previously received the same request and cheated the user. If other colluding fellows have cheated the user, the SP can return a same query result to cheat the user.

We explain why these colluders still fail to harm users' interest. The prerequisite of a user to select the next SP is that the user **has completed the first query**. Therefore, the first SP does not know who the next SP is and has a high risk of losing rewards if it behaves dishonestly. More specifically, the first colluder will lose rewards, if the user chooses a second SP who is not colluding with them. The first colluder can only cheat the user, when he is sure the user will select a next SP who is colluding with them. However, the randomized SP selection approach coupled with non-ordered asynchronous querying primitive makes it impossible for the first colluder to infer who the user will

choose next. Consequently, the first colluder can never be sure wether he is able to deceive a user through collusion.

Moreover, the **self-interest hypothesis** can easily prove the impracticality of SPs colluding together. Assume that a SP tries to negotiate result duplication (fake results) with other set of SPs. If one of those SPs in the system turns out to be honest and reports the dishonest SP to the IQUERY system, then the dishonest SP risks losing its reputation and incurred rewards. **Given the high-risk outcome of a dishonest behavior**, we believe it is against the self-interest of the SP to behave dishonestly. In other words, given the workings of the system and the secure protocols in place, it is in the self-interest of the SP to behave honestly. We can conclude that there is no well-established reward mechanism possible for behaving dishonestly, except losing reputation and existing financial gains.

(3) Malicious irrational SPs. They are willing to spend time and money to convey questionable results or disrupt the system.

We introduce the reputation mechanism, if a SP is reported to have repeatedly provided incorrect results, it is deleted from the IQUERY pool. As all service orders are registered in the blockchain and IQUERY signs the query results, users can identify adversaries who deliver malicious query services and lose rewards from users. We also offer an alternative way to evaluate the credibility of SPs, i.e., sending SPs some test queries for which the results are already known. The IQUERY system is able to remove these malicious users from the pool.

5.4 Game Theory Analysis

In this section, we provide a formal analysis to explain why the second opinion model can prevent the collusion. We assume that a user requires services from IQUERY, but he/she does not know which IQUERY can meet his/er requirements. The user benefits from the query service a only if it matches his query demand α. If $a = \alpha$, his/er utility U_{user} is v. Otherwise, the utility is zero [34].

$$U_{user} = \begin{cases} v, & \text{if } \alpha = a \text{ where } v > 0 \\ 0, & \text{if } \alpha \neq a \end{cases}$$

5.4.1 User Payoff

We define the user payoff under distinct searching strategies as follows. f is probability that the user stops after the first search and $1 - f$ is the probability of the user searching for the second matching results. The probability of a user selecting an IQUERY with high effort is x.

Definition 2. If the user stops after the first search, the user's payoff is

$$U^{(1)}_{user} = xv - p - d \tag{2}$$

Definition 3. If the user has to search for second matching results, the user's payoff is:

$$U^{(2)}_{user} = v - p - \frac{2d}{x} \tag{3}$$

By comparing the payoff $U^{(1)}_{user}$ in Eq. 2 with $U^{(2)}_{user}$ in Eq. 3, the user chooses the strategy that bringing higher payoff.

Proof 5.1. We prove the user payoff $U^{(2)}_{user}$ in Definition 3. First, we need to prove the search cost of finding two matching results. We assume the query result of the n-th IQUERY is correct and a random sampled IQUERY computes the correct query result with probability x. Therefore, the expected duration of the search for one correct query result is $\frac{1}{x}$. This means that

$$\sum_{i=0}^{n}(1-x)^i = \frac{1-(1-x)^n}{1-(1-x)} = \frac{1}{x} \tag{4}$$

Therefore, the expected duration of the search for two correct query results is $\frac{2}{x}$ and the corresponding expected search cost is $\frac{2d}{x}$. The payoff $U^{(2)}_{user}$ in Eq. 3 equals the utility of a user getting correct results (v) minus the amount that the user rewards IQUERYS who make consistent query results (p) minus the search cost of finding two matching results ($\frac{2d}{x}$).

5.4.2 IQUERY Payoff

We define the IQUERY payoff under distinct computation strategies as follows. First, let probability B denotes the IQUERY's belief conditional on the user accepting the query result.

Definition 4 (IQUERY's belief). B is formalized as:

$$B = \frac{x}{fx + 2(1-f)} \tag{5}$$

We use the following notations to calculate B:

- n: the number of query results received before the sampling of IQUERY k.
- H_k: the set of histories such that k is sampled.
- $P(n|H_k)$: the probability that a user receives n query results before sampling IQUERY k (conditional on k being sampled) [35].
- T: the random stopping time of the search over the set of IQUERYS excluding k.

Proof 5.2. We prove IQUERY belief B of Definition 4. We compute $P(n \mid H_k)$ by decomposing the sampling process over IQUERYS other than k into the disjoint events, T = 1, 2, ... This yields,

$$P(n \mid H_k) = \sum_{m \geq n+1} P(n \mid T = m, H_k) Pr(T = m \mid H_k) \tag{6}$$

Notice that,

$$P(n \mid T = m, H_k) = \begin{cases} \frac{1}{m} & if\ 0 \leq n \leq m-1 \\ 0 & otherwise \end{cases} \tag{7}$$

Furthermore,

$$\begin{aligned} Pr(T = m \mid H_k) &= \frac{Pr(H_k \mid T = m) Pr(T = m)}{\sum_{n \geq 1} Pr(H_k \mid T = n) Pr(T = n)} \\ &= \frac{m \times Pr(T = m)}{\sum_{n \geq 1} n \times Pr(T = n)} \end{aligned} \tag{8}$$

where the last equality uses the fact that

$$\frac{Pr(H_k \mid T = m)}{Pr(H_k \mid T = n)} = \frac{m}{n} \tag{9}$$

To compute the likelihood ratio of two zero probability events we take the limit of shrinking neighbourhoods of positive probability. Hence,

$$\begin{aligned} \frac{Pr(H_k \mid T = m)}{Pr(H_k \mid T = n)} &= \lim_{\epsilon \to 0} \frac{Pr(\{H_{\tilde{k}} \mid \tilde{k} \in [k, k+\epsilon]\} \mid T = m)}{Pr(\{H_{\tilde{k}} \mid \tilde{k} \in [k, k+\epsilon]\} \mid T = n)} \\ &= \lim_{\epsilon \to 0} \frac{1-(1-\epsilon)^m}{1-(1-\epsilon)^n} = \frac{m}{n} \end{aligned} \tag{10}$$

and therefore equation Eq.9 follows. Substituting Eq.8 and Eq.7 into Eq.6 yields

$$B = P(0 \mid H_k) = \frac{\sum_{m\geq 1} Pr(T=m)}{\sum_{n\geq 1} n \times Pr(T=n)} = \frac{1}{f + (1-f)\frac{2}{x}} \tag{11}$$

where the last equality follows since $\sum_{m\geq 1} Pr(T=m) = 1$ and $\sum_{n\geq 1} n \times Pr(T=n) = f + (1-f)\frac{2}{x}$.

Definition 5. With high effort, IQUERY's payoff is:

$$U^{high}_{\text{IQUERY}} = d + pfB + (1-fB)\frac{p}{2} - c \tag{12}$$

Definition 6. With low effort, IQUERY's payoff is:

$$U^{low}_{\text{IQUERY}} = d + pfB \tag{13}$$

By comparing U^{high}_{IQUERY} in Eq. 12 with U^{low}_{IQUERY} in Eq. 13, the IQUERY chooses the strategy that bringing higher payoff.

Proof 5.3. In Eq. 12, fB is the probability that the user has never sampled before and stops after the first query and $(1-fB)$ is the probability that the user searches for a matching query results. In the latter case, IQUERY gains a reword with probability $\frac{1}{2}$. This follows because the user samples IQUERYS in random order and the quality of the service provided by the IQUERY following the $\mathcal{U}(0,1)$ uniform distribution according to the assumptions in §5.4. On the other hand, if IQUERY does not incur the cost c, it will give an incorrect query result. The expected profit U^{low}_{IQUERY} in this case is $d + pfB$ in Eq. 13.

Equilibrium. The profile (d, p, x, f) is a *fixed price equilibrium* if the user's search strategy, f, is optimal given (d, p, x) and the IQUERYS's effort decision $x \in [0,1]$ is optimal given (d, p, x, f) and the belief B. (d, p, x, f) is *non-degenerate* if IQUERYS choose high effort with positive probability, i.e. if $x > 0$.

5.4.3 Incentive Parameter Control

Based on the definitions of payoff functions, we can derive theorem 1 and theorem 2 to calculate the conditions that need to be satisfied between monetary variables. Users control incentive monetary parameters d and p in the smart contract, so that $d \leq \overline{d} \equiv \frac{v}{2\sqrt{2}+3}$ (according to Theorem 1) and $p > 2c$ (according to Theorem 2) are satisfied. We motivate the rational user to ask a second opinion and IQUERYS to return the correct query results with high effort to maximize their payoffs. Therefore, we can minimize the number of query SPs while guaranteeing the trustworthiness of the query results.

Theorem 1 (Conditions for a user to search a second opinion). In a non-degenerate fixed price equilibrium, if $d \leq \overline{d} \equiv \frac{v}{2\sqrt{2}+3}$ is satisfied in the smart contract, the user weakly prefers to ask a second opinion and search for two matching query results where

$$U^{(2)}_{user} \geq U^{(1)}_{user}, \ i.e.,$$
$$v - p - 2(\frac{d}{x}) \geq xv - p - d \tag{14}$$

Users can quantify the value of v based on custom methods, e.g., the analytic hierarchy process (AHP) [36]. Then users are able to calculate the acceptable d to pay to IQUERY for computing the query task, and set the monetary parameters in the smart contract.

Proof 5.4. Eq. 14 can be solved for $x \in [0,1]$ when $d \leq \overline{d} \equiv \frac{v}{2\sqrt{2}+3}$, that is, when the service fee are not too large. In that case, Eq. 14 is satisfied for all $x \in [\underline{x}(d), \overline{x}(d)]$ where $\underline{x}(d) < 1$ and $\overline{x}(d) < 1$ are the two roots of the quadratic equation implied by Eq. 14 when it holds with equality. Eq. 14 implies that the user can only be induced to search for a matching query result if $d \leq \overline{d} \equiv \frac{v}{2\sqrt{2}+3}$. Hence, non-degenerate equilibria can exist only when $d \leq \overline{d}$.

Theorem 2 (Conditions for IQUERY to be honest). In a non-degenerate fixed price equilibrium, if $p > 2c$ is satisfied in the smart contract, IQUERY has a greater incentive to make a high effort to return a correct result where

$$U^{high}_{\text{IQUERY}} > U^{low}_{\text{IQUERY}}, \ i.e.,$$
$$d + pfB + (1-fB)\frac{p}{2} - c > d + pfB \tag{15}$$

c, IQUERY's cost for computing the query request, can be quantified from history off-chain computation cost. Then users are able to set an acceptable p in the smart contract that is greater than $2c$.

Proof 5.5. We need to prove with any probability f, U^{high}_{IQUERY} is always greater than U^{low}_{IQUERY}. If when $f = 0$, $(1-fB)\frac{p}{2} > c$ is satisfied, then for any $f \in [0,1]$, $(1-fB)\frac{p}{2} > c$ will be satisfied. So we have the monetary variable relationship $p > 2c$ that must be satisfied.

5.5 Query Frequency Analysis

Existing blockchain query layer is able to apply the replication-based approach to guarantee the trustworthiness of query results. More specifically, users send query requests to multiple SPs and cross-check the query results. Hence, we analyze how many SPs a user has to query by using the replication-based approach, i.e., the **query frequency** to ensure result trustworthiness with a formal proof.

We make the following assumptions to define the probability of getting trustworthy query results.

- Let n be the number of SPs in the network, and E is the query procedure, i.e., a user sends a query request to a single SP. There are only two possible outcomes in E: A and $\overline{A}$, A = {SP returns correct results}, $\overline{A}$ = {SP returns wrong results}. Then, we have $P(A) = p, P(\overline{A}) = 1 - p = q$ $(0 < p < 1)$.
- If the probability of event A occurring in each E is p, in a Bernoulli process [37], the probability of event A happens k times is $P(A = k) = C^k_n p^k q^{n-k}$, $k = 0, 1, 2, \ldots n$.
- A user can cross-check query results retrieved from different SPs to validate the correctness of query results. A user gets verifiable query results only when the number of correct results is greater than the number of wrong results.

The number of correct results when a user selects any y SPs for query, conforms to the Binomial distribution, i.e., $A \sim Binomial(y, p)$. When y is large, x is not close to 0 or 1, then $Binomial(y, p) \approx Normal(yp, yp(1-p))$.

Definition 7. The probability that a user cannot get verifiable query results is $P(Binomial(y,p) \leq \frac{y}{2})$, i.e., the number of correct results are less than or equal to $\frac{y}{2}$:

$$P(Binomial(y,p) \leq \frac{y}{2}) \approx P(Normal(yp, yp(1-p)) \leq \frac{y}{2}) = pnorm(\frac{y}{2}, yp, yp(1-p)) \quad (16)$$

TABLE 3
Minimum Query Frequency with Given p_{trust} and p_{honest}

p_{trust}	p_{honest}	y
95%	78%	5
95%	80%	4

Minimum Query Frequency Calculation. We assume that there are enough SPs in the network. If a user initiates a query request to an arbitrary SP in the network, p equals the proportion of honest SPs p_{honest} in the network. A user sets a query credibility threshold p_{trust}, i.e., the probability that the user gets a verifiable query result is greater than or equal to p_{trust}. Moreover, we have $p_{trust} = 1 - P(Binomial(y,p) \leq \frac{y}{2})$. With given p_{honest} and Eq. 16, we can compute the **minimum query frequency** y to achieve p_{trust} that satisfies user needs.

Table 3 lists the y with corresponding p_{trust} and p_{honest}. For example, when a user wants to achieve query confidence of 95%, and the proportion of honest SPs in the network is 78%, the user has to query 5 SPs at least. Notably, with the help of the second opinion smart contract, a user only needs to query two SPs to achieve 100% query credibility.

6 Evaluation

In this section, we provide a set of comprehensive experiments. Our second opinion mechanism only needs to compare *two* copies of query results to ensure the trustworthiness (tackle **R1**), while the query frequency is 2 ∼ 134 × less than the state-of-the-art systems. Moreover, our proposed system offers a set of rich query semantics to support different types of queries, e.g., conditional, analytical, and multi-origin query methods (tackle **R2**). In terms of these query semantics, IQUERY is 2 ×, 7 ×, and 1.5 × faster than advanced blockchain and blockchain databases (tackle **R3**).

6.1 Experimental Setup

Cluster Setup. We perform all experiments on a 7 node cluster interconnected with 1Gbps networking. Each node has an equal configuration with 32 cores (2.40GHz Intel Xeon E5 v3 CPU), 256 GB RAM, 2 TB hard drive, and Ubuntu 14.04 Trusty operation system. The prototype is implemented by Node.js, interacting with Hyperledger Fabric v1.4 or Geth v1.10.23 (popular Go implementation of Ethereum) as the underlying blockchain technology, and CouchDB 2.3.1 is used to store off-chain data.

Datasets. The datasets consist of four types of transactions as shown in Table 2 of §3.2, which is simulated by Caliper [38]. The size of each record is 300 Byte, and an example record is in the form of {commodityID: #A1_pizza, Txid:121$e2_{64bit}$, timestamp: t, price:30, ...}. We use the same configurations as the SEBDB experiment [8]. Each block contains 200 transactions. We use SHA-256 for VO generation.

TABLE 4
Query Operation Benchmark

q_1	$\langle [\$eq(organization:?), db_{initUser}\rangle$
q_2	$\langle [b_{start}, b_{end}], \$eq(ProductType:clothes), db_{initCommodity}\rangle$
q_3	$\langle \$in(price:[num_s, num_e]), db_{rechargeToken}\rangle$
q_4	$\langle \$eq(Commodity_{id}:?), db_{ConsumptionJoinCommodity}\rangle$
q_5	$GetTotal(\$eq(User_{id}:?), db_{rechargeToken})$
q_6	$AnanlyzeUniqueCommodity(db_{ConsumptionJoinCommodity})$
q_7	$\langle \text{IQUERY}_n, q\rangle, n = 1, 2, 3...$

Workloads. The workloads include seven queries listed in Table 4. q_1 tracks all $tx_{initUser}$ from the indexed $db_{initUser}$, where the user identities belong to a certain organization. q_2 tracks all $tx_{initCommodity}$ in a time window, i.e., $[b_{start}, b_{end}]$, where the product type is clothes. q_3 tracks all $tx_{rechargeToken}$ where the recharge token is within an amount window, i.e., $[num_s, num_e]$. q_4 tracks the complete information of a consumption payment which contains $Commodity_{id}$. It selects from $db_{ConsumptionJoinCommodity}$ which joins $tx_{commodityConsumption}$ and commodity details. q_5 and q_6 are analytical query operations. q_5 analyzes the total user recharge of $User_{id}$. q_6 tracks unique commodities in all consumption payment transactions. q_7 delivers q to multiple origin SPs (identified by IQUERY_n) to evaluate the performance of multi-origin query operations. Note that q is a user-defined rich semantic query.

Metrics. *Latency* is measured as the average response time per query operation. We evaluate the multi-origin query with the following metrics: (i) VO construction cost in terms of IQUERY server CPU time, (ii) result verification cost in terms of user CPU time, and (iii) size of the VO.

6.2 IQUERY Efficiency and Trustworthiness Evaluation

In this subsection, we first demonstrate efficient implementation of our proposed system by performing q_1 in Fabric, Ethereum, SEBDB, and IQUERY in a single node. Next, we turn on the second opinion method to show off the advantage of IQUERY that ensures the trustworthiness of query results.

Efficiency evaluation in single node. We first raise the block number from 500 to 2500 and fix the returned result size to 10,000. Fig. 6 shows that the proposed system outperforms the state-of-the-art systems in terms of latency. It is because IQUERY applies a record-level B-tree, a more efficient index for retrieving blockchain data than the block-level B-tree used by SEBDB and the Merkle Patricia tree used by Ethereum. Moreover, Fabric utilizes the smart contract APIs to query the world state database where the indexes are not optimized.

Secondly, we fix the block number to set the number of transactions in the blockchain database and adjust the result size from 2,000 to 1,250,000. Fig. 7 shows IQUERY achieves

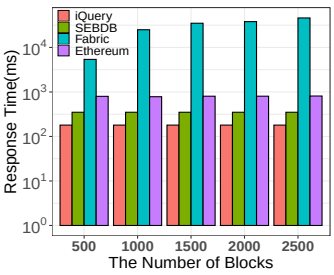


Fig. 6. Varying Block Number in Conditional Query.

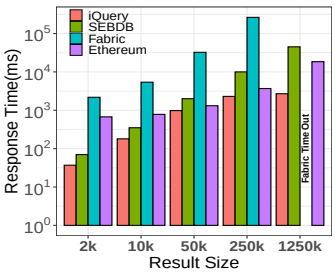


Fig. 7. Varying Result Size in Conditional Query.

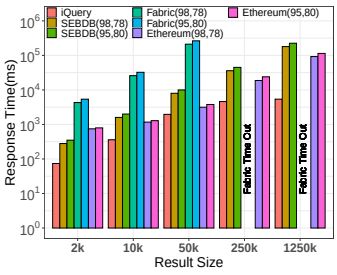


Fig. 8. Impact of the Second Opinion Smart Contract

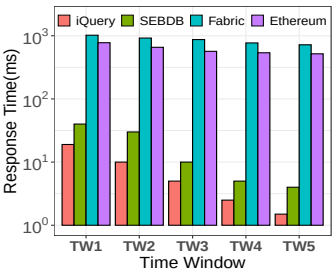


Fig. 9. Varying Time Window.

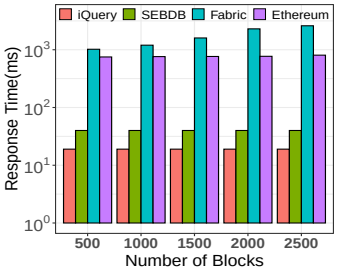


Fig. 10. Vary Range Data Size

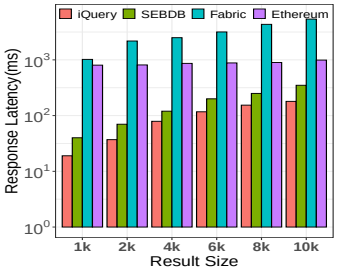


Fig. 11. Vary Range Result Size

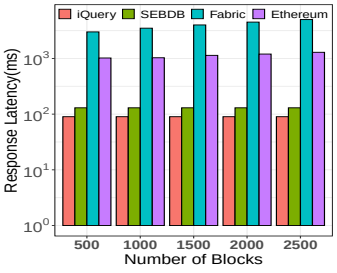


Fig. 12. Vary Join Data Size

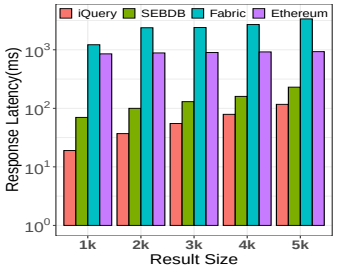


Fig. 13. Vary Join Result Size

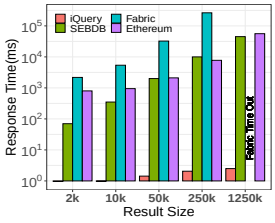


Fig. 14. Vary Record Data Size in q_5

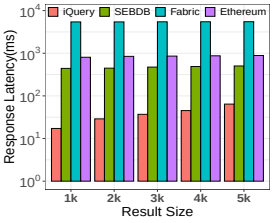


Fig. 15. Vary Record Data Size in q_6

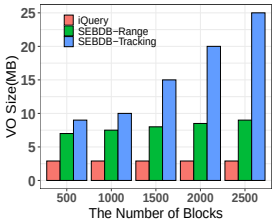


Fig. 16. VO Size

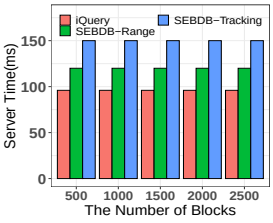


Fig. 17. Running Time at Server Side

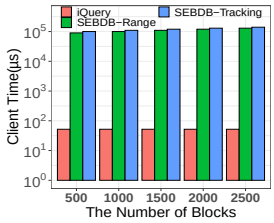


Fig. 18. Running Time at Client Side

the smallest latency. Note that, Fabric runs out of memory when the result size is 1,250,000.

The efficiency of achieving trustworthiness. Our second opinion mechanism only needs to compare *two* copies of query results to ensure the trustworthiness which is 2.5 or 2 × less than SEBDB, Fabric and Ethereum with different trustworthiness configurations. The fewer copies of query results, the fewer resulting transactions will be queried. Consequently, IQUERY reduces the query latency significantly.

To define trustworthiness, we denote the query configurations in the form of C/P, where C is the confidence of query results, and P is the percentage of honest SPs in the blockchain query layer. We intend to compare the query efficiency of three systems under different trustworthiness configurations. Fig. 8 shows that the latency of IQUERY is 7 ∼ 270 × less than other three systems for executing q_1 when the result size rises.

By employing the second opinion smart contract, a user only needs to query two origins to ensure that rational IQUERY will return correct results. SEBDB, Fabric and Ethereum support query ledger data from local databases, e.g., the state and history databases. According to the analysis in Table 3 of §5.5, when the confidence is 98%, and the percentage of honest SPs in the blockchain query layer is 78%, a user needs to query four origin SPs in SEBDB(98,78). When C is 95%, and P is 80%, a user needs to query five origin SPs in SEBDB(95,80). Consequently, with fixed single query result size, SEBDB/Fabric/Ethereum(98,78) and SEBDB/Fabric/Ethereum(95,80) need to return four × and five × the amount of data a single q_1 request requires, respectively.

6.3 Evaluation of IQUERY Flexibility

In this subsection, we evaluate the rich query APIs offered by IQUERY to demonstrate its flexibility while guaranteeing its efficiency.

Range query. q_2 is a temporal query, which is required to provide a pair of time window (TW) parameters ($start, end$). Let $start = 0$, and change end block number to $1000/2^{i-1}$ for TW_i. We set blockchain to contain 10,000 $tx_{initCommodity}$ and there exist 1,000 resulting transactions. Fig. 9 shows that IQUERY outperforms the other three systems for executing q_2.

Fig. 10 and 11 show that IQUERY achieves the best performance, while Fabric has the highest latency for executing q_3. In Fig. 10, the number of resulting transactions is fixed to 1,000. The resulting transactions are the $tx_{rechargeToken}$ whose price attribute falls into the price window. In Fig. 11, the number of resulting transactions is changed from 1,000 to 10,000. The latency of IQUERY, SEBDB and Ethereum remains stable. It is because IQUERY applies the layered index in CouchDB, while SEBDB and Ethereum employ a customized layered index in the contract layer.

Join query. Fig. 12 and 13 demonstrate that IQUERY achieves the best performance, compared with two other systems for performing join query q_4. Fig. 12 shows that the latency of IQUERY and SEBDB remains the same when we increase block number (blockchain database size) and fix the

result size to 10,000. It is because IQUERY applies a joined table $db_{ConsumptionJoinCommodity}$, whereas SEBDB employs a layered index.

Fig. 12 shows that Fabric has the most significant latency. Because in Fabric and Ethereum queries, we need to add user details, price, and commodity information fields for $tx_{commodityConsumption}$ in the smart contract. It significantly increases the read and write time of smart contracts. Furthermore, in order to implement q_4 operations, SEBDB needs to query $tx_{commodityConsumption}$ table and commodity information table with a layered index for the join operation. The runtime join operations introduce an additional operation cost. IQUERY applies a joined table $db_{ConsumptionJoinCommodity}$ built in the block information extracting stage in advance, thus reducing query latency. Fig. 13 shows that IQUERY maintains its performance advantages when the result size rises from 1,000 to 5,000.

Analytical query. The analytical query is a unique feature offered by IQUERY, which allows users to reconstruct meta records and analyze blockchain data (see §4.2). We extend Fabric, Ethereum and SEBDB by first querying blockchain data and then analyzing data on the client side. Fig. 14 illustrates that the latency of IQUERY is $10^2 \sim 10^5 \times$ less than other three systems for executing q_5 with the record size from 2,000 to 1,250,000. The record size is the amount of blockchain transaction records that need to be analyzed. IQUERY maximizes the parallel processing for designing the analytical query to achieve outstanding performance. Note that, Fabric is unable to respond to the query requests when the record size is 1,250,000. Fig. 15 shows that IQUERY's latency is $25 \sim 7 \times$ smaller than SEBDB, $315 \sim 85 \times$ less than Fabric, and $46 \sim 13 \times$ less than Ethereum when performing q_6 with the record size from 1,000 to 5,000.

6.4 Evaluation of the Multi-Origin Query

The multi-origin query enables users to efficiently verify the correctness of query results from distinct IQUERY. Because Fabric and Ethereum do not implement the VO mechanism, in this subsection, we benchmark multi-origin query q_7 in IQUERY and SEBDB. As mentioned in §4.3, the multi-origin query is an authenticated query method that can increase the trustworthiness of any query semantics. Note that it is not query semantics but **resulting transactions** that affects the performance of the multi-origin query. On the other hand, authenticated range query and authenticated tracking query have different performances in SEBDB. Consequently, we denote SEBDB-Tracking and SEBDB-Range as the performance results of authenticated tracking query q_1 and authenticated range query q_2 in SEBDB. Besides, we only use iQuery to denote the performance of authenticated q_1 and q_2 in IQUERY.

VO size. The resulting transactions are set as 10,000. Fig. 16 shows that the VO size of IQUERY is always smaller than the other two methods of SEBDB. The smaller the VO size, the smaller the network bandwidth will be consumed in large-scale application query scenarios. SEBDB adds sibling nodes to VO, thus increasing VO size. Fig. 16 illustrates that the VO size of IQUERY stays constant, and the VO size of SEBDB increases when the block size increases. Because the VO size of SEBDB depends on the number of blocks that contain query results [8] and the VO size of IQUERY only depends on the size of resulting transactions.

VO construction cost. We compare the VO construction cost in terms of CPU time on the server side. Fig. 17 shows that the VO construction cost of IQUERY is always smaller than the other methods because SEBDB applies Merkle B-tree (MB-tree) to generate VO and IQUERY employs Merkle Bucket Tree (MBT). With the same number of indexed nodes, the height of the MBT is smaller, thus the VO construction cost is smaller.

Result verification cost. We compare the result verification cost in terms of user CPU time on the client side. Fig. 18 shows that the result verification cost of IQUERY is $10^3\times$ less than SEBDB-Tracking and SEBDB-Range. Because the multi-origin query method in IQUERY builds a MBT based on query results, the client only needs to check the root hash to verify the correctness of query results. On the other hand, the client needs to reconstruct every MB-tree root, where the MB-tree contains query results in SEBDB.

7 Conclusion and Future Work

We propose IQUERY, a novel blockchain analytics system which is scalable and decentralized. A user pays the SP for every query request and applies the replication-based approach to guarantee the trustworthiness of query results. However, some SPs may collude and deliver fake results to save costs and maximize their benefits. Therefore, we propose a query protocol based on game theory that can destroy collusion's economic foundation.

In real-world scenarios, a trusted third party is usually in charge of enforcing protocols. Fortunately, blockchain provides a trusted execution environment, and smart contracts guarantee the protocol execution according to predefined rules. Consequently, we design the second opinion smart contract, incentivizing SPs to deliver honest query services and minimizing the user's monetary cost. Moreover, the trustworthiness, flexibility and effectiveness of our system have been proved in a real blockchain system. Extensive experiments demonstrate that the performance of our system significantly surpasses state-of-the-art systems.

For future work, we plan to build a decentralized reputation mechanism for IQUERY. We may further propose a reputation blockchain, a side chain recording the query orders between IQUERY and users in the blockchain query layer. The consensus protocol in reputation blockchain utilizes reputation rank as the measurement of IQUERY 's trust and the probability of a leader-elected consensus process. Users tend to select high reputation IQUERYS for query services, thus increasing the revenue of these IQUERYS.

Acknowledgments

This paper is financially supported by the National Key R&D Program of China (2021YFB2700500, 2021YFB2700501) and Key R&D Program of Zhejiang Province (2022C01086). Ye Yuan is supported by the National Key Research and Development Program of China (Grant No. 2022YFB2702100) and the NSFC (Grant Nos. 61932004, 62225203, U21A20516). We thank Shizhou Yang for proof in query frequency, Hanning Ruan for improving paper writing, and Jianting He for helping to build the Ethereum experiments.

REFERENCES

[1] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, "Combining graph neural networks with expert knowledge for smart contract vulnerability detection," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pp. 1–1, 2021.

[2] C. Xu, C. Zhang, and J. Xu, "vchain: Enabling verifiable boolean range queries over blockchain databases," in *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)*, 2019, pp. 141–158.

[3] M. Zhaofeng, W. Lingyun, W. Xiaochang, W. Zhen, and Z. Weizhe, "Blockchain-enabled decentralized trust management and secure usage control of iot big data," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4000–4015, 2019.

[4] L. Lu, J. Chen, Z. Tian, Q. He, B. Huang, Y. Xiang, and Z. Liu, "Educoin: a secure and efficient payment solution for mooc environment," in *Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain 2019)*. IEEE, 2019, pp. 490–495.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the 13th European Conference on Computer Systems Conference (EuroSys 2018)*. ACM, 2018, pp. 1–15.

[6] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang, "Fine-grained, secure and efficient data provenance on blockchain systems," vol. 12, no. 9. VLDB Endowment, 2019, pp. 975–988.

[7] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, "Blockchain meets database: design and implementation of a blockchain relational database," vol. 12, no. 11. VLDB Endowment, 2019, pp. 1539–1552.

[8] Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, "Sebdb: Semantics empowered blockchain database," in *Proceedings of the 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1820–1831.

[9] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[10] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proceedings of the 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2018, pp. 264–276.

[11] C. Cai, L. Xu, A. Zhou, and C. Wang, "Toward a secure, rich, and fair query service for light clients on public blockchains," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2021.

[12] J. Chen, Z. Zhan, K. He, R. Du, D. Wang, and F. Liu, "Xauth: Efficient privacy-preserving cross-domain authentication," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2021.

[13] L. Yin, J. Xu, and Q. Tang, "Sidechains with fast cross-chain transfers," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2021.

[14] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, "Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*, 2017, pp. 211–227.

[15] Y. Guo, J. Tong, and C. Feng, "A measurement study of bitcoin lightning network," in *Proceedings of the 2nd IEEE International Conference on Blockchain (Blockchain 2019)*. IEEE, 2019, pp. 202–211.

[16] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[17] "Fisco-bcos," https://github.com/FISCO-BCOS, 2021.

[18] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "Falcondb: Blockchain-based collaborative database," in *Proceedings of the 2020 International Conference on Management of Data (SIGMOD)*, 2020, pp. 637–652.

[19] "Etherscan: The ethereum block explorer." https://etherscan.io/, 2022.

[20] S. Wu, L. Wu, Y. Zhou, R. Li, Z. Wang, X. Luo, C. Wang, and K. Ren, "Time-travel investigation: Toward building a scalable attack detection framework on ethereum," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–33, 2022.

[21] H. Kalodner, M. Möser, K. Lee, S. Goldfeder, M. Plattner, A. Chator, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," in *Proceedings of the 29th USENIX Security Symposium*, 2020, pp. 2721–2738.

[22] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "Etherql: a query layer for blockchain system," in *Proceedings of the 22nd International Conference on Database Systems for Advanced Applications (DASFAA)*. Springer, 2017, pp. 556–567.

[23] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "Gemˆ2-tree: A gas-efficient structure for authenticated range queries in blockchain," in *Proceedings of the 35th international conference on data engineering (ICDE)*. IEEE, 2019, pp. 842–853.

[24] H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "Vql: efficient and verifiable cloud query services for blockchain systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 33, no. 6, pp. 1393–1406, 2021.

[25] A. Küpçü, "Incentivized outsourced computation resistant to malicious contractors," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 14, no. 6, pp. 633–649, 2015.

[26] E. Fernandes, A. Rahmati, K. Eykholt, and A. Prakash, "Internet of things security research: A rehash of old ideas or new intellectual challenges?" *IEEE Security & Privacy*, vol. 15, no. 4, pp. 79–84, 2017.

[27] "Hyperledger fabric sdk for node.js," https://hyperledger.github.io/fabric-sdk-node/, 2021.

[28] Z. Huang, Y. Huang, P. Qian, J. Chen, and Q. He, "Demystifying bitcoin address behavior via graph neural networks," in *Proceedings of the 39th International Conference on Data Engineering (ICDE)*, 2023.

[29] U. Demirbaga and G. S. Aujla, "Mapchain: A blockchain-based verifiable healthcare service management in iot-based big data ecosystem," *IEEE Transactions on Network and Service Management (TNSM)*, 2022.

[30] M. Zhang, Z. Xie, C. Yue, and Z. Zhong, "Spitz: a verifiable database system," *Proceedings of the 46th International Conference on Very Large Data Bases (VLDB)*, vol. 13, no. 12, pp. 3449–3460, 2020.

[31] "Read-write set semantics," https://hyperledger-fabric.readthedocs.io/en/latest/readwrite.html, 2021.

[32] "Introduction to views," https://docs.couchdb.org/en/latest/ddocs/views/intro.html, 2021.

[33] C. Yue, Z. Xie, M. Zhang, G. Chen, B. C. Ooi, S. Wang, and X. Xiao, "Analysis of indexing structures for immutable data," in *Proceedings of the 2020 International Conference on Management of Data (SIGMOD)*, 2020.

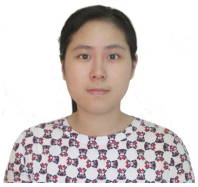
[34] W. Pesendorfer and A. Wolinsky, "Second opinions and price competition: Inefficiency in the market for expert advice," *The Review of Economic Studies*, vol. 70, no. 2, pp. 417–437, 2003.

[35] W. Mimra, A. Rasch, and C. Waibel, "Second opinions in markets for expert services: Experimental evidence," *Journal of Economic Behavior & Organization*, vol. 131, pp. 106–125, 2016.

[36] T. L. Saaty, "What is the analytic hierarchy process?" in *Mathematical models for decision support*. Springer, 1988, pp. 109–121.

[37] R. J. Larsen and M. L. Marx, *Introduction to Mathematical Statistics and Its Applications: Pearson New International Edition PDF eBook*. Pearson Higher Ed, 2013.

[38] "Hyperledger caliper," https://github.com/hyperledger/caliper/, 2020.



Lingling Lu is currently pursuing the Ph.D. degree with the College of Computer Science and Technology , Zhejiang University, Hangzhou, China. She engages in blockchain security and optimization research work and finished two blockchain application projects. Her research interests include blockchain, database managements and big data processing.

Zhenyu Wen (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Newcastle University, Newcastle Upon Tyne, U.K., in 2011 and 2016, respectively. He is currently a Professor with the Institute of Cyberspace Security and college of Information Engineering, Zhejiang University of Technology, China. His current research interests include IoT, crowd sources, AI system, and cloud computing. For his contributions to the area of scalable data management for the internet-of-things, he was awarded the the IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researchers) in 2020.

Ye Yuan received his BS, MS and PhD degrees in Computer Science from Northeastern University, China in 2004, 2007 and 2011, respectively. He is now a Professor at the College of Information Science and Engineering at Beijing Institute of Technology. His research interests include graph databases, probabilistic databases, data privacy-preserving and cloud computing.

Binru Dai is currently pursuing the Ph.D. degree with School of Economics, Zhejiang University. Her research interests include game theory and price competition.

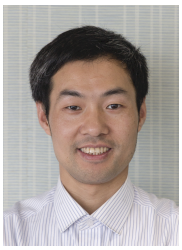
Peng Qian respectively received his B.S. and M.S. degrees from Yangtze University and Zhejiang Gongshang University, in 2018 and 2021. He is currently pursuing his Ph.D. degree with the College of Computer Science and Technology, Zhejiang University. His research interests include blockchain security and deep learning.

Changting Lin received the Ph.D's degree in computer science from the Zhejiang University in 2018. His is currently a post-doctoral of Big data statistics method and application at Zhejiang Gongshang University, China. His research interests include IoT, Blockchain and SDN.

Qinming He (Member, IEEE) received the BS, MS, and PhD degrees in computer science from Zhejiang University, Hangzhou, P. R. China, in 1985, 1988, and 2000, respectively. He is currently a professor with the College of Computer Science & Technology, Zhejiang University. His research interests include data mining and blockchain system security.

Zhenguang Liu is currently a professor of Zhejiang University. He had been a research fellow in National University of Singapore, and A*STAR (Agency for Science, Technology and Research, Singapore) for three years. He respectively received his Ph.D. and B.E. degrees from Zhejiang University and Shandong University, China, in 2010 and 2015.

His research interests include blockchain security and multimedia data analysis. Various parts of his work have been published in first-tier venues including CVPR, TKDE, TIP, AAAI, ACM MM, INFOCOM, IJCAI. Dr. Liu has served as technical program committee member for conferences such as ACM MM, ICME, and MMM, session chair of ICGIP, local chair of KSEM, and reviewer for IEEE Transactions on visualization and computer graphics, IEEE Transactions on Dependable and Secure Computing, ACM MM, ACM Transactions on Multimedia Computing Communications and Applications, etc.



Jianhai Chen (Member, IEEE) received the MS and PhD degrees in computer science and technology from Zhejiang University (ZJU), Hangzhou, China. He is currently an associate professor of the College of Computer Science and Technology, ZJU. He is also the director of ZJU Super Computing Team, and the director of ZJU Intelligent Computing Innovation and Entrepreneurship Laboratory (ICE-lab). His research interests include blockchain system security, cloud computing scheduling algorithms and game theory, super computing application optimization, and data mining. He is a member of the CCF and ACM.

Rajiv Ranjan (Senior Member, IEEE) is a Full professor in Computing Science at Newcastle University, United Kingdom. Before moving to Newcastle University, he was Julius Fellow (2013-2015), Senior Research Scientist and Project Leader in the Digital Productivity and Services Flagship of Commonwealth Scientific and Industrial Research Organization (CSIRO C Australian Government's Premier Research Agency). Prior to that he was a Senior Research Associate (Lecturer level B) in the School of Computer Science and Engineering, University of New South Wales (UNSW). Dr. Ranjan has a PhD (2009) from the department of Computer Science and Software Engineering, the University of Melbourne.