Cloak: Hiding Retrieval Information in Blockchain Systems via Distributed Query Requests

Jiang Xiao, *Member, IEEE*, Jian Chang, *Student Member, IEEE*, Licheng Lin, *Student Member, IEEE*, Binhong Li, *Student Member, IEEE*, Xiaohai Dai, *Member, IEEE*, Zehui Xiong, *Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Keke Gai, *Senior Member, IEEE*, and Hai Jin, *Fellow, IEEE*

Abstract—The privacy-preserving query is critical for modern blockchain systems, especially when supporting many crucial applications such as finance and healthcare. Recent advances in blockchain query schemes mainly focus on enhancing the traceability efficiency of integrity authentication. Despite these efforts, we argue that the exposure of retrieval information may result in privacy leakage, which inevitably poses an important yet unresolved challenge. In this paper, we introduce *Cloak*, a novel privacy-preserving blockchain query scheme with two notable features. First, it utilizes a two-phase distributed query requests technique, i.e., division and aggregation, to hide retrieval information based on the natural independent characteristic of blockchain. Second, we add noise to the sub-request set to avoid malicious attacks during transmission and adopt smart contract-based asymmetric encryption to guarantee the correctness of query results. Experimental results demonstrate that *Cloak* improves the query performance by up to 4$\times$ and reduces the storage overhead by 50% compared with the state-of-the-art Spiral.

Index Terms—Blockchain, Privacy-preserving, Distributed query, Lightweight client.

1 Introduction

Blockchain systems with traceability are now appealing to facilitate a wide range of applications, including finance, logistics, and healthcare [1]. Its traceability can be realized by the blockchain query mechanism on the immutable on-chain historical data [2]. Traditionally, a blockchain client (i.e., lightweight node) with limited computing and storage resources can only proceed with the query requests on the full nodes that store a complete history of data [3], [4]. However, as the query processing relies on the single full node, the malicious behavior of the full node in an untrusted blockchain environment may cause private information leakage of the client's query request.

Consider a concrete example of a blockchain-based healthcare system. A patient wishes to query for drugs associated with some disease (e.g., *Nucleoside Reverse Transcriptase Inhibitors* (NRTIs) or *Non-Nucleoside Reverse Transcriptase Inhibitors* (NNRTIs) for HIV) through the blockchain-based healthcare system. If the disease' or drug's name forms the query content, then the requested content will indirectly infer that the patient may have such a condition. Hence, potential privacy leakage will occur during the query request process, and a malicious node can infer the client's private health information [5].

There are currently two main approaches to protecting data privacy on the blockchain. One approach is based on the *Trusted Execution Environment* (TEE) to achieve confidentiality (e.g., Ekiden [6], BITE [7], and DeSearch [8]). In TEE, query processing is executed independently and can be performed in parallel to improve query performance. However, the enclave size is limited in TEE, with a maximum capacity of 256M, and cannot be widely used. In addition, it also relies on centralized hardware, which suffers from rollback and other attacks [9]. Another approach is based on data encryption mechanisms, such as homomorphic encryption [10] and attribute-based encryption [11], which leverages the complex ciphertext to resolve the problem of data privacy leakage. Nevertheless, the query processing relies on a single centralized node, and the malicious node can still uncover the user identity from encrypted query requests [12]. It can be observed that the existing methods are still limited to a centralized query framework and do not fully use the characteristics of multiple independent ledgers in blockchain to provide more effective privacy protection.

In this paper, we design Cloak, the first privacy-preserving query scheme to support effective privacy protection and verifiability of distributed requests for blockchain lightweight clients. The key idea is to leverage multiple independent ledgers of blockchain full nodes to realize distributed privacy protection. Our goal is to protect the query request submitted by the query client to the full node and to execute the query without revealing the private retrieval information. In other words, neither the broadcast of query requests nor the malicious attacks will

- *This work was supported by National Key Research and Development Program of China under Grant No. 2021YFB2700700, Key Research and Development Program of Hubei Province No. 2021BEA164. (Corresponding author: Jian Chang.)*
- *Jiang Xiao, Jian Chang, Licheng Lin, Binhong Li, Xiaohai Dai, and Hai Jin are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Laboratory, and the Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: jiangxiao@hust.edu.cn; j_chang@hust.edu.cn; lichenglin@hust.edu.cn; binhong@hust.edu.cn; daixh@hust.edu.cn; hjin@hust.edu.cn).*
- *Zehui Xiong is with the Singapore University of Technology and Design, Singapore (email: zehui_xiong@sutd.edu.sg)*
- *Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, USA (email: raymond.choo@fulbrightmail.org)*
- *Keke Gai is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (email: gaikeke@bit.edu.cn)*

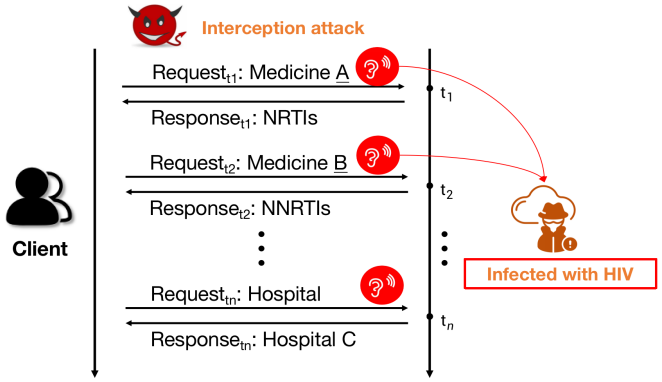
expose information that violates privacy. Specifically, Cloak guarantees a reliable query privacy protection feature in the sense that a malicious node can know which client is sending a request but will not know what the client queried. More importantly, Cloak does not leak the access pattern or correlation between request and results, and clients can still verify results (i.e., ensuring correctness and completeness). We highlight the salient features of the proposed design below.

Distributed privacy-preserving query. Based on the independent ledger feature of the blockchain, we leverage distributed requests to realize the privacy protection of blockchain queries for lightweight clients. The client can submit a distributed set of query requests to full nodes and aggregate the results that are synchronized with the blockchain. In Cloak, we propose a new protocol, named 'n-node' distributed query, which differs from existing works. Generally speaking, we leverage distributed functions to divide a query request among multiple nodes and then aggregate the data from different nodes. In this way, the desired result can be obtained from distributed nodes without revealing the client's private data. In addition, the one-to-one sub-request query process is extended to a one-to-n request process in the distributed query process. Due to the mutual independence between full nodes, any full node that performs sub-request queries cannot obtain the original query content and complete query results. The same sub-request queries different full nodes and verifies the returned results. When the number of consistent results exceeds half of the total fragmented results, the correctness of the query results can be guaranteed.

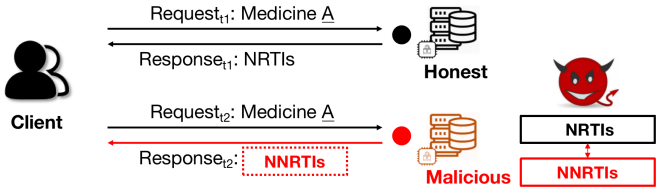
Secure data transmission. To shield the query data during transmission, we add the noises to the requests. In particular, when a client creates random sub-requests from the distributed function, the pseudo-random function will also generate the noises. When a client sends a sub-request query, both the noise and the request are sent, namely *noise-based distributed requests*. An adversary cannot obtain the original query content from any sub-request set with noises embedded. At the same time, *smart contract-based asymmetric encryption* is introduced for the returned result to ensure the security of the requested content during transmission. The client sends sub-requests to multiple full nodes to tolerate the malicious behavior of untrusted nodes during the query process and verify the encrypted results to ensure correctness.

In summary, we make the following contributions:

- We propose Cloak, a distributed privacy-preserving query framework, which leverages the characteristics of the distributed independent ledger of blockchain to protect the privacy of client query requests.
- To protect the privacy of request content for the lightweight client, we leverage a two-phase distributed query requests technique, i.e., division and aggregation, to hide retrieval information.
- We add noise-based distributed requests and smart contract-based asymmetric encryption to guarantee the correctness of query results.
- Experimental results show that Cloak provides significant privacy protection for lightweight clients with sensitive information independent of specific hardware and does not affect overall system performance.



(a) Scenario 1: interception attacks during query transmission process



(b) Scenario 2: malicious attacks on retrieval information

Fig. 1: Two privacy leakage scenarios of distributed query requests in blockchain systems

The rest of this paper is organized as follows: In Section 2, we outline Cloak's research background and motivation before introducing the relevant preliminary materials in Section 3. Sections 4 and 5 describe Cloak's system design and performance evaluation results, respectively. Finally, we review related approaches in Section 6 before concluding this work in Section 7.

2 Motivation and Challenges

Data query is a necessity for clients (i.e., lightweight nodes) to retrieve valuable information from servers (i.e., full nodes) on blockchain [13]. To motivate the design of Cloak, we present the following two scenarios to illustrate the privacy leakage for clients during the request process and from malicious full nodes.

2.1 Motivating Examples

Malicious full nodes can obtain the query request of a client and infer information such as the identity and needs of the client, leading to the risks of privacy leakage of the client's location and health. There are two aspects of privacy leakage in an untrusted blockchain environment when the lightweight client performs a data query.

Scenario 1: Privacy leakage by interception attacks during the query transmission process. Suppose a lightweight client sends a query request containing medicine A and medicine B to a full node. Full nodes store all medicines and healthcare data in the blockchain medical system. The

hackers can intercept retrieval information during the data transmission process and learn that the medicine is used to treat HIV, as shown in Fig. 1(a). The hacker continues to intercept other client requests and obtains that the client has HIV in hospital C. The hacker infers personal health information, such as specific diseases corresponding to the client, through this process. Worse still, the hackers can manipulate the requests with uninteresting advertising messages or spam. To address this issue, the search engine Google adopts DNT (*Do Not Track*) request service to prevent user information from being tracked [14]. DNT service provides optional privacy control over the search content of a user. It's equivalent to telling the website: please don't follow me, but most browser websites reject and ignore users' requests for DNT service. This service provided by search engines has been in a state of failure for query users, and the user's request content cannot be effectively guaranteed. They cannot know whether their query request information is tracked. Therefore, the client should not wholly trust the network transmission or search engine services and directly send query requests with the plaintext to the full node.

Scenario 2: Privacy leakage by malicious attacks on retrieval information. When the full node receives the query request from the client, it first analyzes the status of the query request. If the request is in plaintext state, the full node can directly query the data through the request [15]. This process completely trusts the full node. However, supposing that the full node is malicious, it can arbitrarily obtain the client's query request and results. In that case, the data that involves clients' private information will be leaked entirely to malicious nodes. More seriously, the query results returned by the full node may also be tampered with by selfish motivation, as shown in Fig. 1(b). If the query request is encrypted, the full node needs to obtain the corresponding data access authority or data decryption key through the client and then perform a data query. In this way, the client's query request can avoid leakage during transmission, and the data query of the full node needs to rely on the consent of the lightweight node, which can partially protect privacy. But in the same way, if the node is malicious, although the data is encrypted, the full node can obtain the approximate content of the query data by analyzing the access pattern of the encrypted data. Therefore, the client relies on a single full node to query sensitive data, leading to privacy leakage during the query process.

2.2 Challenges

In view of the existence of malicious nodes in the blockchain system, when encrypted data is used to protect user confidentiality, the data still needs to be decrypted when relying on full nodes for a query. The original data confidentiality problem solved by these systems is different from the query request protection problem that Cloak tries to solve. Instead, we propose distributed, verifiable privacy-preserving querying over plaintext data generated by lightweight clients. To make the problem more concrete, the data can be viewed as a binary array $D = [d_0, ..., d_{l-1}]$ of length l, and $n \geq 2$ nodes independently store the same ledger data. The client wants to query data d_i but does not want the full node to know that the queried data is with index i. Currently, we concentrate on the entire blockchain data query $Q = (i)$, which can also be extended to time range queries to meet the diverse future needs of blockchain users. To elaborate on time range queries, we provide the following formal specification: The query $Q = (i, < t_{start}, t_{end} >)$ targets data within a defined historical time range, facilitating the retrieval of information from the designated start block with timestamp t_{start} to end block with timestamp t_{end}. This query variant proves particularly valuable for applications necessitating the analysis of blockchain data over specific periods, such as financial audits, transaction trend analysis, and regulatory compliance checks. For historical time range queries, efficient scoping of the query becomes imperative. This entails leveraging blockchain indexing techniques to swiftly identify and retrieve pertinent blocks within the specified time range, thereby enhancing the system query performance and facilitating seamless data retrieval.

When a client (lightweight node) is querying in blockchain, a primary overlooked issue is that requesting data can also lead to privacy leaks. Since the full blockchain nodes handle the query process, the query operation and request data are transparent. However, this is flawed for specific cases dealing with user-sensitive data, such as patents and stock. When the client needs to query the full node for patent information that they are preparing to apply for, they do not want to leak the query request data to avoid being preemptively registered by others. Investors like Warren Buffet want to inquire about specific stock information in the stock market. However, he is unwilling to disclose the stocks that he is interested in to affect the stock price and his preferences.

These cases show that clients' potential privacy will be leaked. Therefore, a novel privacy protection mechanism must be considered to protect potential privacy in the query request process. The challenges of protecting blockchain query requests include two main aspects.

Challenge 1: How to avoid leakage during the broadcast of query requests? The query request is intercepted: hackers can learn the content of the query request and the query result.

Challenge 2: How to prevent leakage by the malicious node? The query operation is exposed: a malicious node can know the query request and tamper with the result.

3 Preliminaries

This section defines the basics involved in privacy-preserving query schemes, which mainly include smart contracts and distributed point functions. We also discuss the advantages of adopting these fundamentals. Table 1 lists the primary notations used in Cloak.

3.1 Smart Contract

Smart contracts were not widely used in the initial era of the blockchain, mainly represented by the Bitcoin system. The emergence of Ethereum opened the prelude to the blockchain 2.0 era defined by smart contracts [16]. Smart contracts make blockchain applications more convenient and expandable. The main advantages are: (1) It has the characteristics of high timeliness and decentralization of

TABLE 1: Summary of primary notations

Notation	Meaning
D	the binary array data stored independently on full node
d_i	the queried data with index i
l	the length of data D
n	the number of full node
$F_{a,b}(x)$	the distributed point function where its value is b at a
k_i	the i-th key
$Gen()$	the algorithm that outputs a pair of keys
$Eval()$	the evaluation algorithm outputs a value
F_k	the denotation of function $Eval(k, \cdot)$
p_i	the denotation of i-th full node
$F_{i,v}(x)$	the distributed point function where its value is v at i
X_i'	the i-th request
$F(X_i')$	the denotation of $Eval(K_i, X_i')$
Y	the result of function $F(X_i')$
R_i	the blockchain index benefit
K_i	the i-th vector
m	the power of matrix rows
u	the power of matrix columns
G	the pseudorandom generator
DPF_1	the vector-based distributed point function
$Gen_1()$	the generation algorithm of DPF_1
$Eval_1()$	the evaluation algorithm of DPF_1
$Dec_1()$	the output decoding function of DPF_1
Q	the query request set n_p
A	the answer set
q_i	the i-th sub-request
a_i	the i-th result
α	the power of entry data set
β	the size of output data
γ'	the row vector of grid
$s_{\gamma',j}$	the j-th random string
cw_i	the i-th random vector
e_δ	the δ-th unit vector
E_n	the set of all arrays with an even number of 1 bits
O_n	the set of all arrays with an odd number of 1 bits
ρ	the proportion of malicious nodes
θ	the probability of at least one honest node
Θ	the probability of more than half of the honest nodes

contract formulation and does not need to rely on the participation of third-party authorities or central institutions. (2) It has a low cost. Smart contracts are based on computer programs and are controlled by pre-established codes, reducing supervision costs. (3) With very high accuracy, no human participation is required, and automatic execution improves the accuracy of the contract.

The smart contract code is deployed on a shared, replicated ledger that maintains its state and responds to incoming external information. The code contains some control conditions that trigger automatic execution, and the relevant results are recorded in the blockchain ledger after the code is executed. When sent to all devices connected to the network, the user can agree on the result of executing the program code and always follow the rules in advance. For example, smart contracts written in the statically typed programming language Solidity can run independently on the *Ethereum Virtual Machine* (EVM). With Solidity, users can program an auto-executing application based on scenario requirements, regarded as an authoritative and never-repentant trading contract. The application scenarios of smart contracts, such as combining with the Internet of Things to enable smart homes, voting, etc., are all application scenarios. In other words, scenarios where machine language can be used to implement established rules, improve efficiency, and prevent users from cheating are the application scenarios of smart contracts.

3.2 Distributed Point Function

Distributed Point Function (DPF) is a privacy protection method that uses distributed features to encrypt data [17]. It divides the shared function into multiple parts for distributed computation. Each process has a piece of information and calculates the result without exposing the original information to any procedure. In fact, in a blockchain network, there may be malicious nodes. To prevent the requested node from being hostile, the client can send query requests to multiple nodes to reduce the connection probability of malicious nodes. The point function is the most fundamental part of the distributed point function. Given any two values a and b one can define a point function $F_{a,b}(x)$ by

$$F_{a,b}(x) = \begin{cases} b & \forall x, \quad x = a \\ 0 & \forall x, \quad x \neq a \end{cases} \tag{1}$$

The function graph of $F_{a,b}(x)$ is a discrete point, for example, if $F_{a,b}(1) = 2, F_{a,b}(2) = 4$, and the rest are 0. Therefore, only when $x \in 1, 2$, $F_{a,b}(x)$ is not 0, which is reflected in the graph, except for the two points of $(1, 2)$ and $(2, 4)$, the other points fall on the x axis. It is zero everywhere except at a, where its value is b.

Simply, a distributed point function can be implemented with a point function $F_{a,b}(x)$ with two keys k_1 and k_2. The privacy protection of query content can be realized by leveraging the distributed point function. DPF allows the client to divide a point function into multiple sub-function fragments. Any proper subset of sub-function fragments does not leak any information about the original function. The full node calculates the value of the sub-function according to x and returns it to the client, and the client can aggregate the result. $Gen()$ generates keys k_1, k_2: $Gen(a, b) \longrightarrow (k_1, k_2)$ that allow the full node union to compute the point function calculated as b at the input a. Each key hides a and b separately, but there is a valid algorithm $Eval$ for each a.

$$Eval(k_1, x) \oplus Eval(k_2, x) = F_{a,b}(x) \tag{2}$$

Letting F_k represent the function $Eval(k, x)$, and the functions F_{k_1} and F_{k_2} can be viewed as an additive secret sharing of $F_{a,b}(x)$. When the client sends query information to node1 and node2, they return the corresponding partial query results according to the keys in the distributed point function. The client obtains the result by XOR of two sub-query results, which can hide the requested information.

4 System Design

In this section, we elaborate on the system design of Cloak, which realizes the privacy protection of query requests in the face of challenges pinpointed in Section 2.2. We first present the design principles and overview of Cloak. Then, to meet the system requirements, we introduce the crucial components separately, including secure data transmission and a distributed query process.

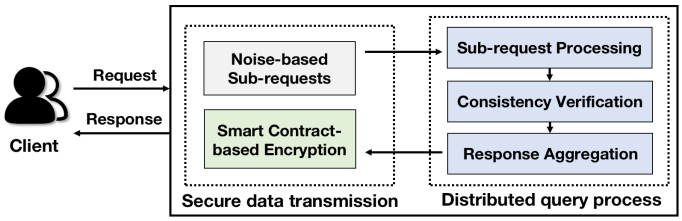


Fig. 2: The system overview of Cloak

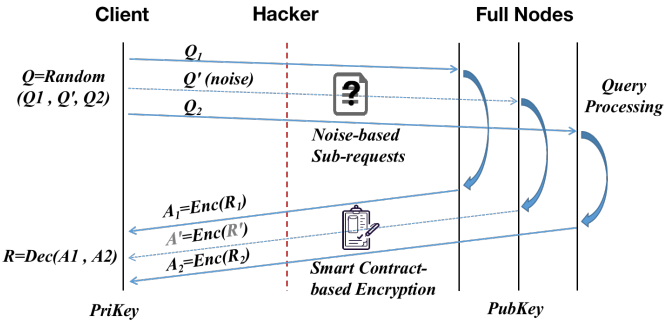


Fig. 3: The secure data transmission of lightweight client requests to full nodes

4.1 Design Principles

The goal is to devise a scheme that supports more efficient privacy-preserving queries for lightweight clients. All Bitcoin and other popular public blockchain systems transactions are transparent: every node can synchronize and verify the on-chain data. Clients can generate pseudonyms using one-off encryption keys for addresses, but the processing details of the user's anonymous address transactions and the relationship between the addresses are still transparent. More specifically, our resolution can meet the following requirements:

- **R1:** Lightweight clients can query what they need without revealing the requested content and query results to the potentially untrusted transaction broadcast.
- **R2:** A full node cannot learn the requested content and query result from the lightweight client to generate additional information about the client's potential identity or behavior.

4.2 Cloak Overview

In general, the Cloak system works similarly as presented in Section 2, where a client sends sub-query requests to full nodes and waits for the responses. However, to prevent malicious actions taken by the hackers or full nodes, we propose a secure data transmission mechanism and distributed query process mechanism in Cloak, respectively, as shown in Fig. 2.

Concretely speaking, during data transmission, we add noise to the requests and implement asymmetric encryption based on smart contracts, thus protecting data privacy. On the full node's side, we present a distributed query scheme based on the blockchain's independent ledger feature. Unlike searchable encryption of high-overhead ciphertext retrieval or a trusted execution environment that relies on additional hardware, we leverage distributed point functions to decompose the query requests and then aggregate the responses to hide retrieval information.

4.3 Secure Data Transmission

To ensure network communication security, it is necessary to encrypt the data transmitted on the network. The current primary method is to rely on the *Transport Layer Security* (TLS) to secure communications. The client uses asymmetric encryption to communicate with the full node to achieve identity verification and negotiate the key used for symmetric encryption. Then, the symmetric encryption algorithm uses the negotiated key to encrypt the information and the information digest. Different nodes use symmetric keys to ensure that only the communicating parties can obtain the data.

However, this method has man-in-the-middle attacks during transmission. The attacker creates independent connections with both ends of the communication and exchanges the data they receive, making the two ends mistakenly believe they are directly talking to each other through a private connection. Still, the whole session is entirely controlled by the attacker. Moreover, the *Certificate Authentication* (CA) center of TLS constitutes a valuable target for malicious attackers, resulting in severe data breaches and leakage. In Cloak, we advocate a client-oriented query transmission design in that the query request will be split and encrypted by the client to deal with *Challenge 1* and meet requirement **R1**. In this way, it gives more control back to the client and can guarantee privacy. In particular, we propose two mechanisms to achieve secure data transmission: noise-based distributed requests and smart contract-based asymmetric encryption, whose schematic diagram is shown in Fig. 3.

Noise-based Distributed Requests. The query request is split through a distributed function and sent to different full nodes. In this way, hackers intercepting any subset of the set of sub-requests cannot acquire the original request content. However, this method also brings a new problem: how to keep the hacker from getting real query content if it possesses all the sub-requests? When a hacker intercepts all sub-requests, although s/he cannot recover the original request, s/he can send all sub-requests to full nodes for the query to obtain the corresponding query results. To deal with this problem, we add noises to the sub-request set, which ensures that any combination of sub-requests cannot recover query results. In addition, while generating sub-requests by distributed functions, corresponding noise requests are also generated. Instead of adding noises to the result in differential privacy, we add noises to the query sub-requests. The advantage is that our approach is more preventive and request-oriented in the pre-query stage, whereas differential privacy is result-oriented in the post-query stage. The noise and standard requests are sent simultaneously when the client sends a sub-request query. In this way, even if the hacker intercepts all the sub-requests, the query result cannot be obtained, thus protecting the privacy and security of the request during the sending

process.

Smart Contract-based Asymmetric Encryption. To ensure the security of the transmission process, the asymmetric encryption method is used to encrypt the query results. However, there is also an important problem: how to prevent the hacker from collecting returned query results with the asymmetric public key? In other words, it can be difficult for a full node to prove that the transmitted public key is announced by the client rather than tuned by the attacker. Traditionally, we can use the public key certificate issued by the CA and its related agencies to solve this problem. However, CA turns out to be a third-party organization that needs trust from both the client and the full node. To overcome this bottleneck, we recur to the smart contract to realize the decentralized trust without relying on digital CA. A smart contract-based asymmetric encryption scheme is used to protect the privacy of returned data, which does not depend on the TLS protocol at the network transport layer. In this regard, the public key is releasable for any full node to use, and the client owns the private key for decryption.

4.4 Distributed Query Process

In this section, we first introduce a strawman design with some problems therein to deal with *Challenge 2* and meet requirement **R2**. To solve these problems, the vector-based and one-to-n request mechanism of Cloak is proposed. Recall the scenario we mentioned earlier, a client wants to query an element with index i (i.e., d_i) from data $D = [d_0, ..., d_{l-1}]$, where l represents the total length of D. In the untrusted environment, all the independent full nodes hold D, and more than half of them are honest nodes. The client wants to get the data d_i without revealing the index i to any full node. Generally speaking, we achieve this privacy-preserving query scheme based on the blockchain's inherent distributed independent ledger feature.

4.4.1 *Strawman*

To aid the presentation of the distributed query process in Cloak, we first introduce a two-node strawman design, which can work roughly but suffers from a new challenge. The strawman is designed for a blockchain system consisting of two full nodes (p_1 and p_2) and a client, where p_1 and p_2 store the identical data D. Let $F_{i,v}(x)$ be a function to return the value according to the value of x, where i and v represent the index and corresponding value in D. If $x = i$, $F_{i,v}(x) = v$; otherwise, namely $x \neq i$, $F_{i,v}(x) = 0$. A client wants to query the D for information with the entry i. Two query requests, X_1' and X_2', are delivered to node1 and node2, respectively. Node1 and node2 calculate independently $F(X_1') = Eval(K_1, X_1')$, $F(X_2') = Eval(K_2, X_2')$, and sent $F(X_1')$, $F(X_2')$ to the client respectively. Then, the client gathers these two answers and only needs to compute $Y = F(X_1') \oplus F(X_2')$. If the query object x does exist, then the result is $Y = F(X_1') \oplus F(X_2') = F_{i,v}(x = i) = v$, if not there is the result $Y = F(X_1') \oplus F(X_2') = F_{i,v}(x \neq i) = 0$.

A properly distributed point function needs to be constructed. The simple way is to construct a truth table according to all possible input x as follows:

$$Y = \begin{cases} v, & x = i \\ 0, & x \neq i \end{cases} \tag{3}$$

Based on a random function G to generate R_i, let $F(X_1') = R_i$ and $F(X_2') = R_i \oplus Y_i$ $(i = 1, 2, ..., 2^{|x|})$, as shown in Table 2. When the client sends query requests to full nodes, they return the answer according to the truth table. However, this comes with a new problem: the size of the truth table grows exponentially with the length of the query data, making it difficult to be practical.

TABLE 2: The truth table of distributed point function

X'	Y	$F(X_1')$	$F(X_2')$
X_1'	Y_1	R_1	$R_1 \oplus Y_1$
X_2'	Y_2	R_2	$R_2 \oplus Y_2$
X_3'	Y_3	R_3	$R_3 \oplus Y_3$
...	...	...	...
$X_{2^{\|x\|}}'$	$Y_{2^{\|x\|}}$	$R_{2^{\|x\|}}$	$R_{2^{\|x\|}} \oplus Y_{2^{\|x\|}}$

In the distributed query process, the lightweight client splits the query request into multiple sub-requests and sends them to the full node for the query. However, there are untrusted full nodes on the blockchain. The connected full nodes may have malicious behaviors, so it is necessary to tolerate malicious nodes and verify the query results returned by the full nodes. In other words, there is a new problem: How to overcome the obstacle of malicious nodes appearing on connected nodes in the query process? That is, an untrusted node can produce malicious behaviors, such as denial of service, and return inconsistent results. If a sub-request connects to a malicious node one-to-one, the query result must be invalid.

4.4.2 ***Cloak***

As mentioned above, the size of the truth table is too large in section 4.4.1 (Strawman). To solve this problem, an advanced vector-based $DPF_1 = (Gen_1, Eval_1)$ is constructed. We also propose a practical and scalable n-node solution (Cloak), and convert the one-to-one sub-request request process into a one-to-n batching query process for malicious behaviors, as shown in Fig. **??**.

Consider the input of $F_{i,v}(x)$, and its total input space is of size $2^{|x|}$. If all possible inputs x are listed in matrix form, the matrix will contain 2^m rows and 2^u columns, where $m + u = |x|$. Each element in the matrix corresponds to a possible input of x, and each x can be converted in the format of coordinates $(row, column)$ in the matrix. The random $0-1$ bit group builder $G : \{0,1\}^\lambda \longrightarrow \{0,1\}^{2^u}$ takes the bit vector of length λ as input and returns a random bit vector of size 2^u as output. Gen_1 generates two vectors of size $2^m + 1$ with only one bit being different. The two vectors will be assigned to K_1 and K_2, respectively. In other words, we have $K_1 = \langle s_1, ..., s_i^0, ..., s_{2^m} \rangle$ and $K_2 = \langle s_1, ..., s_i^1, ..., s_{2^m} \rangle$.

Given input x', where $x' = (i', j')$. The pseudo-random number generator G calculates $G(K[i'])$ and generates a bit vector of size 2^u, where $K[i']$ means the value with the index i' in K, namely $s_{i'}$. Therefore, $Eval_1(K, x')$ will return the j_{th} vector among 2^u vectors, namely $G(K[i'])[j']$.

Fig. 4: Privacy-preserving N-node Cloak protocol during distributed query process in the case of malicious nodes

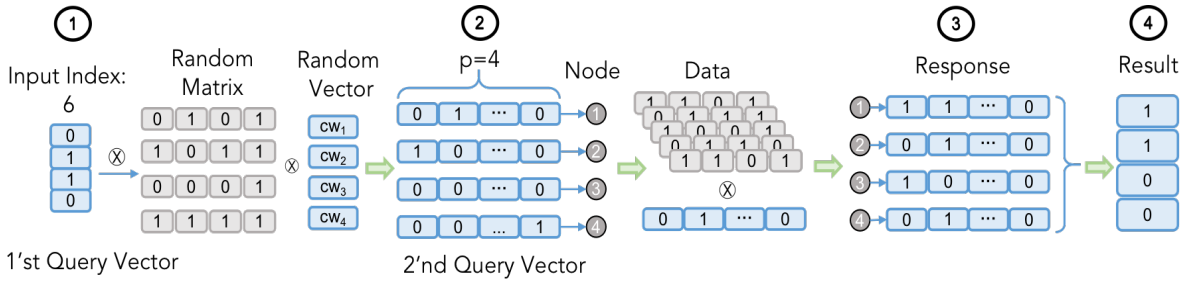


Fig. 5: The distributed vector computation process of Cloak

For each $x' = (i', j')$, we have the following equation:

$$Eval_1(K_1, x') \oplus Eval_1(K_2, x') = \begin{cases} G(s^0_{i'})[j'] + G(s^1_{i'})[j'], & i' = i \\ G(s_{i'})[j'] + G(s_{i'})[j'] = 0, & i' \neq i \end{cases} \quad (4)$$

We now present a scalable n-node Cloak based on DPF_1, which prevents any alliance of up to n-1 malicious nodes. An n-node Cloak protocol involves n full nodes $p_1, ..., p_n$, storing the identical data D, and a client who wants to retrieve the bit d_i of D. The protocol $P = (Q, A, Gen_1, Eval_1, Dec_1)$ consists of a random probability distribution query request set $Q = \{q_1, ..., q_n\}$, an answer set $A = \{a_1, ..., a_n\}$, a random request generating function Gen_1, an answering function $Eval_1$, and a reconstruction function Dec_1. To start the protocol, the client picks a random seed and computes a query $q_j = Gen_1(j, r)$. The query is then sent to each node p_j. Each node responds with an answer $a_j = Eval_1(q_j, D)$. Finally, the client computes the bit d_i by applying the reconstruction algorithm $Dec_1(a_1, ..., a_n)$.

The 2^α-entry data set of the distributed point function $F_{i,v}(x) : \{0,1\}^\alpha \to \{0,1\}^\beta$ can be regarded as a $2^{\alpha/2} \times 2^{\alpha/2}$ grid. Gen_1 generates random requests. For the row $\gamma' \in \{0,1\}^{\alpha/2}$, it consists of 2^{n-1} random λ-bit strings $s_{\gamma',1}, ..., s_{\gamma',2^{n-1}} \in \{0,1\}^\lambda$. Moreover, it also produces 2^{n-1} vectors $cw_1, ..., cw_{2^{n-1}} \in (\{0,1\}^\beta)^{2^{\alpha/2}}$, which satisfies the following constraint: $\bigoplus_{j=1}^{2^{n-1}} (cw_j \oplus G(s_{\gamma',j})) = e_\delta \cdot v$, where e_δ is the δ-th unit vector. $Eval_1$ outputs the sub-query result. First, given an input x, it parses $x = (\gamma', \delta') \in \{0,1\}^{\alpha/2} \times \{0,1\}^{\alpha/2}$. Next, it expands the row γ' via pseudorandom generator G to a vector $(\{0,1\}^\beta)^{2^{\alpha/2}}$. Then, it takes the exclusive OR of the data set with the sub-query request and outputs the result.

Given $n \in N$, E_n and O_n are binary arrays with the length of $n \times 2^{n-1}$. E_n is the array with an even number of 1 bits. O_n is the array with an odd number of 1 bits. The computation process is shown in Fig. 5. The client inputs the index values to get the first query vector ①. Random matrices and random vectors are selected to operate with the first query vector to generate multiple second query vectors ②. The client sends the second query vector to multiple nodes. Each node calculates the data and query vector separately and sends the obtained response to the client ③. The client receives the response returned by multiple nodes, calculates the retrieval value through the aggregation algorithm, and then obtains the desired retrieval results ④.

Since the query result returned by the malicious node may be empty or inconsistent with the correct result, the client will have inconsistent sub-request results when collecting the results returned by multiple full nodes. Therefore, it is necessary to conduct a statistical analysis of the results generated in the one-to-n query process to extract the correct results. In other words, a problem posed by malicious nodes is how to separate the correct results from invalid ones in the returned answer set. To deal with it, we count the query results of sub-requests from different full nodes. Assuming that the proportion of malicious nodes is less than 50%, we can reconstruct the results by aggregation and determine them as correct with more than half of consistent results. The result set size corresponds to the number of connected full nodes. Furthermore, when the proportion of malicious nodes is constant, the more connected full nodes, the more accurate obtaining correct results is, but the higher the system overhead cost. We evaluate the cost in the experimental section. To describe the distributed privacy-preserving query more clearly, we depict the query mechanism with pseudocode, as shown by Algorithm 1.

4.5 Security Analysis

In this section, we analyze the security of Cloak, including the result correctness and privacy, which is described as Theorem 1.

Theorem 1. *The client can retrieve d_i correctly with the result set, and full nodes cannot get any information about i.*

Proof. During the process of query, $\forall t$ $(1 \leq t \leq m)$, t can be expressed as a pair of values, namely $t = (\gamma_t, \delta_t)$. If $\gamma_t \neq \gamma$, then $A_{\gamma_t} \in E_n$. Each term $cw_j \oplus G(s_{\gamma_t,j})$ in $\bigoplus_{j=1}^{n} d_{jt}$ is an even number of times, which offsets each other (i.e., $\bigoplus_{j=1}^{n} d_{jt} = 0^u$). Besides, $\bigoplus_{j=1}^{n} d_{jt}[\delta_t] \cdot d_t = 0^h$ and d_t counteracts each other. On the contrary, if $\gamma_t = \gamma$, then $\bigoplus_{j=1}^{n} d_{jt} = \bigoplus_{j=1}^{2^{n-1}} (cw_j \oplus G(s_{\gamma_t,j})) = e_\delta$. If $\delta_t \neq \delta$ (i.e., $t \neq i$), then $\bigoplus_{j=1}^{n} d_{jt}[\delta_t] \cdot d_t = 0^h$ (i.e., d_t counteracts each other). If $\delta_t = \delta$ (i.e., $t = i$), then $\bigoplus_{j=1}^{n} d_{jt}[\delta_t] = 1$ (i.e.,

Algorithm 1 Distributed Privacy-preserving Query

Input: the query index $i \in \{0, ..., l-1\}$
Output: return data d_i

Let $G : \{0,1\}^{\lambda} \longrightarrow \{0,1\}^{2^u|y|}$ be a random $0-1$ number builder
Let $u \longleftarrow \{0,1\}^{2^u|y|}$ be a random $0-1$ number builder
Let $F_{i,1}(x)$ be the index function on the data D
Let $k_i = (\epsilon_i||cw_1||...||cw_{2^{n-1}})$ for $1 <= i <= n$
Generate n query requests $q_1, ..., q_n$ and noise randomly and independently from Gen
for $j \in \{1, ..., n\}$ **do**
    The client sends request q_j to full node p_j
    if node p_j is honest **then**
        $a_j \longleftarrow Eval(q_j, D)$
    else
        a_j is illegal
The client receives and verifies a_j from all nodes
while a_j is valid **do**
    $y \longleftarrow Dec(a_1, a_2, ..., a_n)$
return $d_i = y$

$\bigoplus_{j=1}^{n} d_{jt}[\delta_t] \cdot d_t = d_t$). Since $i \in \{1, 2, ..., m\}$, there exists a value of t such that $t = i$, thereby $\bigoplus_{j=1}^{n} d_{jt}[\delta_t] \cdot d_t = d_t$. For the other value of t ($t \neq i$), we have $\bigoplus_{j=1}^{n} d_{jt}[\delta_t] \cdot d_t = 0^h$ and thus $\bigoplus_{j=1}^{n} y_i = d_i$. In other words, the user can definitely recover d_i. Therefore, this protocol is correct. The distributed point function obtains the key set $(k_1, k_2, ..., k_n)$ from $Gen(1^{\lambda}, F_{i,v}(x))$ and aggregates all results. Each subset (at most $n-1$ keys) of k_i includes $n-1$ strings δ_i. Therefore, any full node can get neither information of the point function $F_{i,v}(x)$ nor information of i. □

Assume that the proportion of malicious nodes in the blockchain network is less than ρ. The value of ρ can be different under different application scenarios, for example, $\frac{1}{3}$ in the Byzantine Fault Tolerance (BFT) algorithm [18]. The client splits the request into k sub-requests and sends sub-requests to k full nodes. If all the connected nodes are malicious, the result returned by the query may be tampered with and cannot be verified; otherwise, the query result can be judged whether it is true or not. Considering that there are at most m malicious nodes, the probability of at least one honest node is θ, which can be calculated as follows:

$$\begin{aligned}\theta &= C_k^1 \times \rho^{k-1} \times (1-\rho) + ... + C_k^k \times \rho^0 \times (1-\rho)^k \\ &= \sum_{i=1}^{k} C_k^i \times \rho^{k-i} \times (1-\rho)^i \\ &= \begin{cases} 1-\rho^k & (k \leq m) \\ 1 & (k > m) \end{cases}\end{aligned} \tag{5}$$

When the number of connected nodes exceeds malicious nodes, the probability that the query result will not be tampered with is 1. The client can satisfy different confidence intervals by adjusting the number k of requesting nodes. When there are malicious nodes, the number of consistent sub-query results is calculated by statistics to verify the correctness of the final result. The probability of connecting more than half the number of honest nodes is Θ, which can be figured out as Equation 6.

$$\Theta = \sum_{i=k/2}^{k} C_k^i \times \rho^{k-i} \times (1-\rho)^i \tag{6}$$

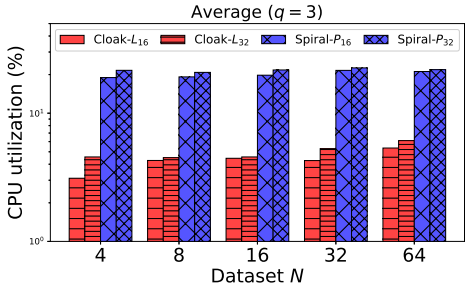
5 Evaluation

This paper introduces a general privacy-preserving query scheme Cloak for blockchain lightweight clients, and in this section, we present the system setup and performance evaluation. We analyze the system's throughput and query latency under different conditions. In addition, the system's average CPU usage rate and memory usage during run-time is also evaluated.

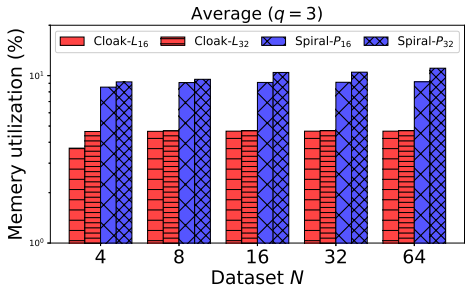
5.1 Experimental Setup

We perform experiments on the server Intel Xeon (Skylake) Platinum 8163 with 2.5GHz CPUs, running CentOS 7.9 with four cores and 16 GB Memory. The Cloak system is programmed in Rust language, an open-source framework for creating blockchain applications. Integrating Cloak with platforms such as Ethereum and Hyperledger Fabric has the potential to significantly enhance their privacy-preserving capabilities. However, this integration has two main challenges. The primary challenge arises from the diverse architectures of Ethereum and Fabric, each characterized by distinct design paradigms and consensus mechanisms. To successfully integrate Cloak, it is imperative to adapt its distributed query framework to operate seamlessly across these varying architectures. This can be achieved by developing platform-specific modules or plugins for Cloak, custom-tailored to the unique architectural nuances and functionalities of Ethereum and Fabric. Such an approach requires a comprehensive understanding of each platform's intricacies and meticulous optimization of Cloak's components. Another critical challenge is to ensure compatibility with smart contracts, which serve as the backbone of the Cloak query framework for managing distributed queries. Seamless integration hinges on ensuring compatibility with Ethereum's Solidity language and Fabric's chaincode environment. One effective strategy is to provide standardized interfaces and blockchain APIs, facilitating smooth interaction between Cloak's query framework and the smart contracts employed by Ethereum and Fabric. Additionally, the development of middleware may be necessary to bridge communication gaps and facilitate seamless integration with the underlying blockchain infrastructure.

In this paper, we analyze the performance of the Cloak and compare it with state-of-the-art Spiral, which is a private information retrieval scheme based on homomorphic encryption, and we implement it on the blockchain. The focus is on four metrics: response time, the throughput of query processing, computation utilization, and storage requirements. Response time, namely latency, includes the query content submitted from the clients to the full node and the time needed to receive the query answer. The throughput of query processing is evaluated through the



(a) CPU utilization



(b) Memory usage

Fig. 6: The system cost of computation and storage

operations per second of request, and computation usage is considered through average CPU utilization. Finally, we evaluate the system memory footprint for operation.

5.2 System Cost

Computation. The average CPU usage is smaller in Cloak than in Spiral (Fig. 6(a)). Cloak sends client requests to multiple full nodes through a distributed function, generating less computational overhead than the Spiral. The CPU usage in Cloak is slow linear ascent, with dataset N from 4 to 64. In contrast, Spiral sends the encrypted client request to the full node through a homomorphic encryption function, and this process generates a large computational overhead, which is why Spiral-P_{16} and Spiral-P_{32} have much higher CPU usage than Cloak-L_{16} and Cloak-L_{32}. The CPU usage in Spiral changes slightly as the data set increases and depends on the different lengths of P. The parameter P is responsible for a significant part of the raw data length in Spiral, which is why Spiral-P_{32} has higher CPU usage than Spiral-P_{16}.

The difference in CPU usage between Spiral and Cloak is due to the query process method. Unlike homomorphic encryption queries, the CPU usage for Cloak depends on the number of connected full nodes and is still directly proportional to the request size. We also explore the CPU usage to process requests for different fragments of the distributed point function in Cloak. As these fragments are under 3, the length of raw data is about 16 and 32, Cloak-L_{16} can effectively process the request and has the lowest CPU utilization. The usage rate of CPU does not change much with the dataset, and the overall usage rate of Cloak is less than 70% compared with Spiral.

Storage. Fig. 6(b) shows Cloak uses nearly equal memory storage space for the different dataset N. As the dataset increases, the memory utilization of Cloak-L_{16} and Cloak-L_{32} remains almost constant at around 4.5% when the dataset N is more than 8. However, for Spiral-P_{32} and Spiral-P_{32}, the average memory utilization rises slowly with the increase of dataset N. We can observe that Spiral's memory usage is, on average, more than twice as high as Cloak's for different data sets.

Cloak mainly processes the query's request data fragment. For the same number of query fragments, the client can choose a small number of connected full nodes to minimize memory usage. Cloak's memory usage is almost unaffected by the transaction data volume size. However, Spiral needs to deal with the encrypted data of both requests and raw transaction data. As the dataset grows, the volume of transactions increases and more memory is required to load ciphertext data. Therefore, the Spiral's memory utilization keeps increasing with the dataset in a small range.

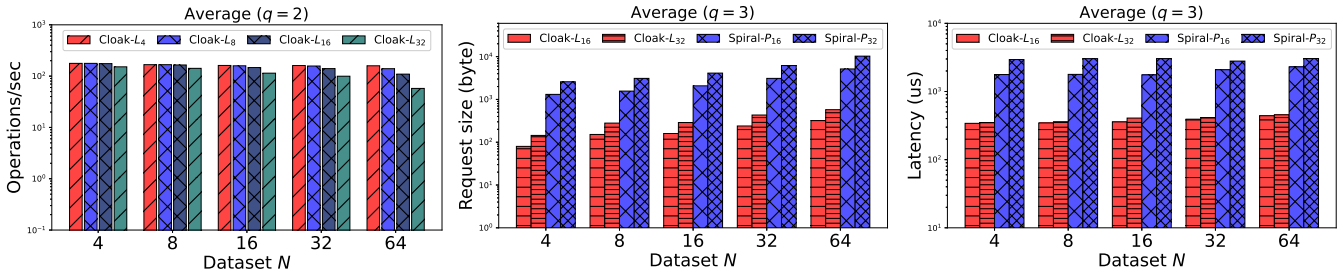
5.3 Query Performance Analysis

Throughput. Fig. 7(a) shows the throughput comparison under different workloads for Cloak. Note that the measurements have no regard for the network malfunction. When a query fragment fails during the entire operation, the request is resubmitted, ensuring that all requests are completed successfully. Our Cloak uses distributed processing of requests, which can be sent simultaneously in parallel compared to Spiral with single serial processing. The throughput of distributed requests in Cloak-L_4, Cloak-L_8, and Cloak-L_{16} can be maintained above 100 operations per second in the different workloads, which improves by more than 40% compared with Cloak-L_{32} in dataset 64. With the dataset increase, there is a flattening off and then a slight decline, which is insignificantly small since the request processing capability does not continue to grow.

When processing different data lengths, we observe that the throughput of Cloak-L_4 and Cloak-L_8 remains stable as the dataset within 64. The reason is that the shorter data length can be quickly calculated by the full node, and the request size is smaller than Cloak-L_{16} and Cloak-L_{32}. In Spiral, the request operation is generated by homomorphic encryption, which is processed in a single full node and needs more computation resources. Therefore, to be fair, we only compare distributed multi-node Cloak with different parameters, not centralized single-node Spiral.

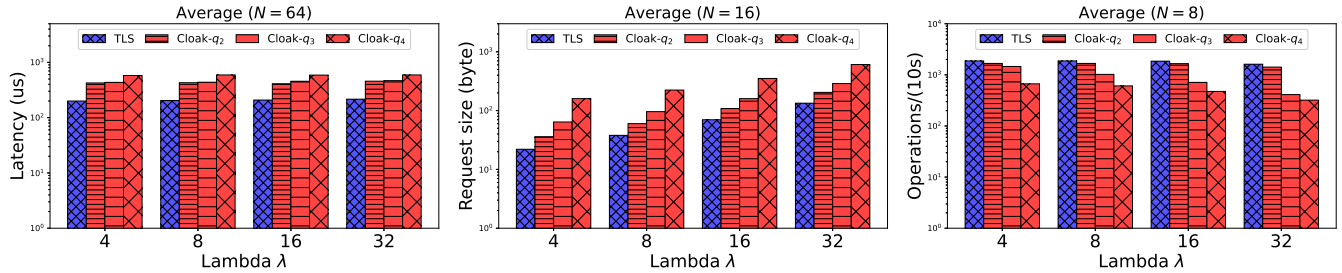
Request size. Fig. 7(b) shows the request size of Cloak and Spiral during the dataset from 4 to 64 and the data length from 16 to 32. The client sends requests to full nodes, and the request size stands for the amount of network overhead occupied. Cloak-L_{16} and Cloak-L_{32} generate smaller request data compared with Spiral-L_{16} and Spiral-L_{32} under the dataset from 4 to 64. Cloak keeps the request size within 0.5 KB as the dataset increases.

For the Spiral with homomorphic encryption, we evaluate Spiral-L_{16} and Spiral-L_{32}, with the same raw data length compared to the Cloak. They have more than 15x and 20x request sizes compared with Cloak-L_{16}. When processing the dataset within 64, we observe that Spiral-L_{16} and Spiral-L_{32} greatly exceed Cloak-L_{16} and Cloak-L_{32} all the time. The reason is that the ciphertext data generated by homomorphic encryption is much larger than the query fragment



(a) Throughput (b) Request size (c) Latency

Fig. 7: The comparison of query performance



(a) Latency (b) Request size (c) Throughput

Fig. 8: The impact of lambda with different query fragments

generated by the distributed point function. The request size also increases with the length of data, which causes Cloak-L_{32} to be more than 1.5x compared with Cloak-L_{32}.

Latency. Fig. 7(c) shows that the average query latency increases slightly with the number of workloads growth, which is the same with the different system parameters L and P for Cloak and Spiral. The average latency requires a total query time to divide the number of requests in Cloak. In contrast, Spiral query focuses on a single query process, which obtains a most common round trip. The average latency reflects how long a client waits for response results during a search. There is little difference between the latency of Cloak-L_{16} and Cloak-L_{32}, with the average latency less than 0.5 ms. But for Spiral-L_{16} and Spiral-L_{32}, the latency is much higher than Cloak, requiring more than 2 ms to process dataset 64.

We can conclude that the size of requests in Spiral is larger than the distributed query fragments generated by Cloak. As shown in Fig. 7(c), when the dataset is within 64, the latency of Spiral-L_{16} and Spiral-L_{32} is more than four times that of Cloak-L_{16} and Cloak-L_{32}. With the increase in workloads, the query latency of Cloak and Spiral under different parameters approaches gradually. The gap between Cloak-L_{16} and Cloak-L_{32} is only 3 us for dataset 64. Besides, the gap between Spiral-L_{16} and Spiral-L_{32} decreases from 1.2 ms to 0.7 ms when the dataset increases from 4 to 64.

5.4 Impact of System Parameter

In the following, we evaluate the impact of various system parameters on Cloak's latency, request size, and throughput performance in Fig.8. Note that the TLS query method means a solution that relies on network transport layer security with plaintext requests. The lightweight client submits the query request serially without extra camouflage and protection. We set lambda values from 4 to 32 to test the scalability of Cloak. Fig.8(a) and Fig.8(b) reveal the results of query latency and request size with query fragments from 2 to 4. From these figures, we can draw a conclusion that the query latency of plaintext remains almost unchanged as the lambda enlarges, and the request size of plaintext and Cloak-q_2, Cloak-q_3, and Cloak-q_4 increases linearly with the growth of lambda. Cloak has a slower response time than Plaintext under different lambada since the number of sub-requests is at least 2. It is less effective to process the request distribution and result aggregation. However, the gap between them is not obvious, only 0.4 ms at most. When the value of lambda is increased, the overall request size grows simultaneously and more than 20 bytes. The reason is that the request size is positively correlated with the data length (i.e., the value of λ).

Next, we assess the impact of lambda on throughput. Fig.8(c) demonstrates the query operations within 10 seconds under different query fragments, and the lambda changes from 4 to 32 with the average of dataset N at 8. We can notice that the query operations of plaintext and Cloak-q_2 are more than 1400, which declined slightly with the lambda. When the value of lambda is more than 16, the operations of Cloak-q_3 and Cloak-q_4 decrease noticeably since more computations are needed with a large number of query fragments (i.e., the value of q.) The plaintext query with no-privacy protections performs relatively more effectively than Cloak-q_2, Cloak-q_3, and Cloak-q_4 as it is centralized processing of requests with no additional

computational overhead. In Cloak, the distributed query fragments are sent to multiple nodes for processing, and the throughput depends not only on the number of requested nodes but also on the request allocation and result aggregation. In a plaintext query, the query result is directly obtained from the connected full node, and other nodes do not participate. Hence, the throughput performance does not rely on the number of nodes but on the processing time of full nodes.

6 Related Work

This paper's related work mainly includes privacy-preserving queries in a distributed database and data queries on the blockchain. The comparison of Cloak with existing query methods is shown in Table 3.

6.1 Privacy-preserving Query in Distributed Database

Many user files are currently stored on remote servers and retrieved as needed. Clients encrypt the raw data and then upload it to the remote server to protect the file content. This data encryption method can protect remote data and prevent the leakage of server-side data. The privacy protection technology of distributed databases is mainly data encryption [19].

In [20], Song et al. present the first study of searchable encryption, which is a query method in view of ciphertext to resolve data privacy leakage. In this method, the basic technology of cryptography is utilized to protect the security of client privacy data. Aiming at the shortcomings of searchable encryption schemes, Liu et al. [21] propose a novel searchable encryption method SE-EPOM. It can realize multi-keyword queries, hide the query mode, and resist keyword guessing attacks. However, there is access pattern leakage when searching encrypted files. To accomplish timely and accurate truth discovery while protecting individual privacy, Pang et al. [23] propose a personalized privacy-preserving truth discovery (PPPTD) framework for crowdsourcing data streams. To preserve the privacy of sensitive data, Wang et al. [24] propose a novel ownership-enhanced attribute-based encryption query method AESM^2 for multi-owner and multi-user systems. DORY [22] uses a *Private Information Retrieval* (PIR) protocol that hides the indexed columns where the data is located on multiple servers during the query process to preserve search access patterns. The request is shared through different replica nodes during the search to ensure that the access pattern is not leaked. DORY uses batching to amortize costs instead of executing a two-phase commit between the master and replicas for every update. Although distributed trust with the multiple-copy node is adopted, there is a premise that the master node must be a trusted node with centralized dependency.

These works ignore the privacy protection of request content when clients initiate a query since the assumption of an honest but curious centralized server. The distributed database still relies on the centralized authorization server, and the privacy of the requested content depends on the server's security. The inherent problem with the blockchain is that the nodes are decentralized and independent. The malicious nodes are unknown, which may cause the information leakage of query requests.

6.2 Data Query on Blockchain

Data queries on the blockchain have gained increasing attention since they can provide tamper-resistant consensus data. Some efforts provide verifiable query mechanisms to guarantee the reliability of query results from malicious full nodes.

In [26], a vChain solution is proposed by adding verifiable data structures built based on accumulators to the construction of the block. vChain proposes two authenticated indexing structures: one for intra-block data and another for inter-block data, to enable batch verification. AttDigest supports batch verification of multiple objects both within and across blocks. This method adopts a multiset accumulator to make efficient batch validation of Boolean and range query results. However, the public key size will be much larger relative to the data size, and the larger SkipList will increase the service provider's CPU time. LineageChain [27] has proposed a novel skip list index to achieve an efficient on-chain provenance query to speed up the location. This method uses the skip list hash table in the block header to check whether the block is authentic and valid. However, it cannot support integrity verification of query results, as it only supports version-based queries against specific state IDs. In the hybrid-storage blockchain, Zhang et al. [28] present a new data verifiable structure, called GEM^2-Tree, by constructing a two-level index to reduce gas consumption where smart contracts update data structures on the chain and make the query results verifiable. In the GEM^2-Tree, new objects can be consistently inserted into the smaller SMB-trees, which is more gas-efficient. Objects indexed by the SMB-trees can be batch-merged into the MB-tree to optimize update costs. Wu et al. [29] present a verifiable query layer (VQL), which is a middleware layer to ensure efficient and verifiable query service for blockchain systems.

However, these query efforts do not consider another important query aspect, namely, privacy protection. Medical and health data privacy protection is an important application field of blockchain technology in medical treatment [30]. To protect the original data, Azaria et al. [31] propose MedRec, a novel type of multiple replica data maintenance method that uses blockchain technology to process medical and health data. MedRec uses identity verification mechanisms and shared data management to ensure medical data privacy. In [32], a blockchain-based privacy protection quality control mechanism is designed to prevent data from being tampered with and ensure fair distribution of rewards during the execution of mobile crowd-sensing (MCS). Yue et al. [33] propose a *Healthcare Data Gateways* (HGD) model based on blockchain, enabling users to quickly and securely control and share their medical data. HGD offers a purpose-centric access control model to protect user health data privacy. The existing data protection mechanism mainly focuses on controlling user authentication functions, and data access through identity authentication realizes the privacy protection of personal information [34]. However, these works ignore the privacy protection of client query requests in the data query process.

TABLE 3: Comparison of Cloak with existing query methods

Category	Approach	Privacy-preserving		Batching	Verifiable
		Transmission	Processing		
Distributed Database	SE-EPOM [21]	✔	✖	✖	✔
	DORY [22]	✖	✔	✔	✔
	Spiral [25]	✔	✔	✖	✔
Blockchain	vChain [26]	✖	✖	✔	✔
	LineageChain [27]	✖	✖	✔	✔
	GEM^2-Tree [28]	✖	✖	✔	✔
	MedRec [31]	✔	✖	✖	✔
	HGD [33]	✖	✔	✖	✖
	BITE [7]	✔	✔	✖	✖
	Cloak	✔	✔	✔	✔

NOTE: ✔ : support ✖ : poor support

Matetic et al. [7] presented a lightweight client's privacy protection (BITE) for Bitcoin by leveraging a trusted execution environment. BITE leverages SGX to provide all the necessary data for lightweight clients to verify and create transactions, enabling privacy protection for lightweight Bitcoin clients. However, the enclave size is limited in SGX and cannot be widely used. In addition, the data is not encrypted and may be leaked during transmission under a man-in-the-middle attack. BITE can only defend against side-channel attacks, but SGX will also be subject to rollback and other attacks [6], [9], [35]. Nonetheless, this method's restriction is that it is merely applicable to the node equipped with trusted hardware, such as SGX, whose limitation is that it relies on centralized hardware. In contrast, we propose Cloak, which adopts a distributed query framework to preserve the privacy of client requests. In Cloak, the batching requests are distributed to multiple full nodes across the network to tolerate malicious behaviors. Full nodes receiving batching requests process each query and generate a response, which is then returned to the requester. This approach leverages the independent ledger characteristic of blockchain itself, without the need for additional hardware environments.

7 Conclusion

Supporting privacy-preserving queries, particularly for lightweight clients, is crucial but challenging to achieve in blockchain systems. This paper presents Cloak, a privacy-preserving query framework, for the lightweight client using the distributed query. Cloak decomposes a client's request content into multiple sub-requests and sends them to numerous independent full nodes. Then, it aggregates the query results to serve privacy-preserving queries for lightweight clients. The experimental results demonstrate that our query method can protect the privacy of lightweight clients without introducing additional hardware support. At the same time, the query performance is improved by up to $4\times$, and the storage overhead is reduced by 50% compared with the state-of-the-art Spiral.

As blockchain systems continue to evolve rapidly, the demand for querying and processing data grows in tandem. While Cloak effectively streamlines query processing, its performance may diminish with the increasing size and complexity of blockchain data. While Cloak can support vector data queries, it may encounter challenges with complex keyword queries. Exploring advanced indexing and data retrieval techniques is crucial for managing large-scale blockchain data, a critical step towards enhancing Cloak's scalability and performance. Additionally, applying machine learning algorithms for predictive analysis and query execution optimization could further enhance Cloak's capabilities.

Acknowledgments

This work was supported by National Key Research and Development Program of China under Grant No. 2021YFB2700700, Key Research and Development Program of Hubei Province No. 2021BEA164.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. https://bitcoin.org/bitcoin.pdf
[2] R. Han, J. Xiao, X. Dai, S. Zhang, Y. Sun, B. Li, and H. Jin, "Vassago: Efficient and Authenticated Provenance Query on Multiple Blockchains," in *Proceedings of the 40th International Symposium on Reliable Distributed Systems (SRDS 21)*, 2021, pp. 132-142.
[3] X. Dai, J. Xiao, W. Yang, C. Wang, J. Chang, R. Han, and H. Jin, "LVQ: A Lightweight Verifiable Query Approach for Transaction History in Bitcoin," in *Proceedings of the 40th International Conference on Distributed Computing Systems (ICDCS 20)*, 2020, pp. 1020-1030.
[4] H. Jin and J. Xiao, "Towards Trustworthy Blockchain Systems in the Era of "Internet of Value": Development, Challenges, and Future Trends," *Science China Information Sciences (SCIS)*, vol. 65, no. 153101, pp. 1-11, 2022.
[5] S. Jiang, J. Liu, J. Chen, Y. Liu, L. Wang, and Y. Zhou, "Query integrity meets blockchain: A privacy-preserving verification framework for outsourced encrypted data," *IEEE Transactions on Services Computing (TSC)*, vol. 16, no. 3, pp. 2100-2113, 2023.
[6] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P 19)*, 2019, pp. 185-200.
[7] S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, and S. Capkun, "BITE: Bitcoin lightweight client privacy using trusted execution," in *Proceedings of the 28th USENIX Security Symposium*, 2019, pp. 783-800.
[8] M. Li, J. Zhu, T. Zhang, C. Tan, Y. Xia, S. Angel, and H. Chen, "Bringing Decentralized Search to Decentralized Services," in *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation*, 2021, pp. 331-347.
[9] Y. Ren, J. Li, Z. Yang, P.P. Lee, and X.Zhang, "Accelerating Encrypted Deduplication via SGX," in *Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 957-971.
[10] Y. Wang, S.F. Sun, J. Wang, J.K. Liu, and X. Chen, "Achieving searchable encryption scheme with search pattern hidden," *IEEE Transactions on Services Computing (TSC)*, vol. 15, no. 2, pp. 1012-1025, 2020.

[11] M. Li, Y. Chen, C. Lal, M. Conti, M. Alazab, and D. Hu, "Eunomia: Anonymous and secure vehicular digital forensics based on blockchain," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 225-241, 2023.
[12] N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck, "ORCAS: 20 million clicked query-document pairs for analyzing search," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2983-2989.
[13] C. Zhang, C. Xu, H. Wang, J. Xu, and B. Choi, "Authenticated keyword search in scalable hybrid-storage blockchains," in *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE 21)*, 2021, pp. 996-1007.
[14] M. Mehrnezhad, K. Coopamootoo, and E. Toreini, "How Can and Would People Protect From Online Tracking?" in *Proceedings of the 2022 Privacy Enhancing Technologies*, 2022, pp.105-125.
[15] A. EL Azzaoui, H. Chen, S.H. Kim, Y. Pan, and J.H. Park, "Blockchain-Based Distributed Information Hiding Framework for Data Privacy Preserving in Medical Supply Chain Systems," *Sensors*, vol. 22, no. 4, pp.1371-1387, 2022.
[16] M. Fang, Z. Zhang, C. Jin, and A. Zhou, "High-performance smart contracts concurrent execution for permissioned blockchain using SGX," in *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE 21)*, 2021, pp. 1907-1912.
[17] N. Gilboa and Y. Ishai, "Distributed point functions and their applications," in *Proceedings of the 2014 Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2014, pp. 640-658.
[18] C. Stathakopoulou, S. Rüsch, M. Brandenburger, and M. Vukolić, "Practical byzantine fault tolerance," In *Proceedings of the 40th International Symposium on Reliable Distributed Systems (SRDS 21)*, 2021, pp. 34-45.
[19] M. Bailleu, D. Giantsidi, V. Gavrielatos, V. Nagarajan, and P. Bhatotia, "Avocado: A Secure In-Memory Distributed Storage System," in *Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 65-79.
[20] X.D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P)*, 2000, pp. 44-55.
[21] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, and J. Shen, "Privacy-Preserving Multi-Keyword Searchable Encryption for Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 561-574, 2021.
[22] E. Dauterman, E. Feng, E. Luo, R.A. Popa, and I. Stoica, "DORY: An Encrypted Search System with Distributed Trust," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 1101-1119.
[23] X. Pang, Z. Wang, D. Liu, J.C. Lui, Q. Wang, and J. Ren, "Towards personalized privacy-preserving truth discovery over crowdsourced data streams," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 327-340, 2021.
[24] M. Wang, Y. Miao, Y. Guo, H. Huang, C. Wang, and X. Jia, "AESM^2 Attribute-Based Encrypted Search for Multi-Owner and Multi-User Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol.34, no.1, pp.92-107, 2022.
[25] S.J. Menon and D.J. Wu, "Spiral: Fast, high-rate single-server PIR via FHE composition," in *Proceedings of the 2022 IEEE Symposium on Security and Privacy (S&P)*, 2022, pp. 930-947.
[26] C. Xu, C. Zhang, and J.L. Xu, "vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases," in *Proceedings of the 2019 International Conference on Management of Data (SIGMOD 19)*, 2019, pp. 141-158.
[27] P.C. Ruan, G. Chen, T.T.A. Dinh, Q. Lin, B.C. Ooi, and M.H. Zhang, "Fine-Grained, Secure and Efficient Data Provenance on Blockchain Systems," in *Proceedings of the 2019 VLDB Endowment*, 2019, pp. 975-988.
[28] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "GEM^2-Tree: A Gas-Efficient Structure for Authenticated Range Queries in Blockchain," in *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE 19)*, 2019, pp. 842-853.
[29] 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*, vol. 33, no. 6, pp. 1393-1406, 2021.
[30] X. Tang, C. Guo, K.K.R. Choo, Y. Liu, and L. Li, "A secure and trustworthy medical record sharing scheme based on searchable encryption and blockchain," *Computer Networks*, vol. 200, no. 1, pp. 108540, 2021.
[31] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Proceedings of the 2nd International Conference on Open and Big Data (OBD)*, 2016, pp. 25-30.
[32] J. An, Z. Wang, X. He, X. Gui, J. Cheng, and R. Gui, "PPQC: A blockchain-based privacy-preserving quality control mechanism in crowdsensing applications," *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1352-1367, 2022.
[33] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, pp.1-8, 2016.
[34] N. Waheed, X. He, M. Ikram, M. Usman, S.S. Hashmi, and M. Usman, "Security and privacy in IoT using machine learning and blockchain: Threats and countermeasures," *ACM Computing Surveys (CSUR 20)*, vol. 53, no. 6, pp. 1-37, 2020.
[35] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y.T. Hou, "PrivacyGuard: Enforcing private data usage control with blockchain and attested off-chain contract execution," in *Proceedings of the 2020 European Symposium on Research in Computer Security (ESORICS 20)*, 2020, pp. 610-629.

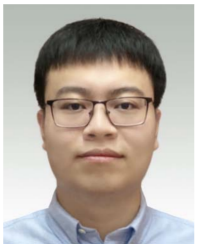
Jiang Xiao (Member, IEEE) is currently a professor in School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China. Jiang received the B.Sc. degree from HUST in 2009 and the Ph.D. degree from Hong Kong University of Science and Technology (HKUST) in 2014. She has been engaged in research on blockchain, distributed computing, big data analysis and management, and wireless indoor localization. Awards include Hubei Dawnlight Program 2018, CCF-Intel Young Faculty Research Program 2017, and Best Paper Awards from IEEE ICPADS/GLOBECOM 2012.

Jian Chang received the M.S. degree in School of Software from Central South University (CSU), Changsha, China, in 2019. He is currently working toward the Ph.D. degree in the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China. His current research interests include blockchain and distributed systems.

Licheng Lin received the B.Sc. degree from Wuhan University of Technology (WHUT), Wuhan, China, in 2022. He is currently pursuing the M.S. degree in the School of Computer Science and Technology from HUST. His current research interests include blockchain and distributed systems.

Binhong Li received the B.Sc. degree from Central South University (CSU), Changsha, China, in 2021. He is currently pursuing the M.S. degree in the School of Computer Science and Technology from HUST. His current research interests include blockchain and distributed systems.

Xiaohai Dai (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2021. He is currently a Post-Doctoral Researcher with the School of Computer Science and Technology, HUST. His current research interests include blockchain and distributed systems. His awards include the Outstanding Creative Award in 2018 FISCO BCOS Blockchain Application Contest and the Top Ten in FinTechathon 2019.



Zehui Xiong (Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore. He is currently an Assistant Professor with the Singapore University of Technology and Design, and an Honorary Adjunct Senior Research Scientist with Alibaba-NTU Singapore Joint Research Institute, Singapore. He was a Visiting Scholar with Princeton University, Princeton, NJ, USA, and the University of Waterloo, Waterloo, ON, Canada. He has authored or coauthored more than 150 research papers in leading journals and flagship conferences and many of them are ESI Highly Cited Papers. His research interests include wireless communications, Internet of Things, blockchain, edge intelligence, and Metaverse.

Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio. He was a recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher) and the IEEE Systems, Man, and Cybernetics Technical Committee on Homeland Security (TCHS) Research and Innovation Award in 2022. He is the Founding Co-Editor-in-Chief of ACM Distributed Ledger Technologies: Research and Practice and the Founding Chair of the IEEE TEMS Technical Committee on Blockchain and Distributed Ledger Technologies.

Keke Gai (Senior Member, IEEE) received the Ph.D. degree in computer science from Pace University, New York, NY, USA. He is currently a Professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. He has published 4 books and more than 150 peer-reviewed journal/conference papers. He serves as Associate Editors for a number of decent journals, including IEEE Transactions on Dependable and Secure Computing, Journal of Parallel and Distributed Computing, etc. He also serves as a co-chair of IEEE Technology and Engineering Management Society's Technical Committee on Blockchain and Distributed Ledger Technologies, a Standing Committee Member at China Computer Federation-Blockchain Committee, a Secretary-General at IEEE Special Technical Community in Smart Computing. His research interests include cyber security, blockchain, privacy-preserving computation, decentralized identity.

Hai Jin (Fellow, IEEE) is a Cheung Kung Scholars Chair Professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. Jin received his Ph.D. in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is Fellow of IEEE, Fellow of CCF, and a member of the ACM. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security.