SemantiChain: A Trust Retrieval Blockchain based on Semantic Sharding

Zihang Zhen, Xiaoding Wang, Xu Yang, Jiwu Shu, *Fellow, IEEE*, Jia Hu, Hui Lin, and Xun Yi

***Abstract*—Since its inception, blockchain technology has found wide-ranging applications in various fields including agriculture, energy, and so on, owing to its immutable and decentralized nature. However, existing blockchains encounter significant challenges in scenarios that demand efficient retrieval of big data. This is primarily because current blockchains cannot directly store and process diverse types of rich media information. Additionally, the semantic relationships between data within the blockchains are weak, complicating the categorization and retrieval of data and transactions. Moreover, the scalability of current blockchains is limited, with the capacity of full nodes continually increasing. Although some semantic-based blockchain solutions that combine off-chain scalability have been proposed, they are limited in effectiveness and applications. To address these issues, this paper introduces a brand-new blockchain sharding technique called *Semantic Sharding*, which enhances blockchain scalability through a hybrid on/off-chain approach. Building on this, we propose a semantic sharding blockchain architecture, *SemantiChain*, which enables the on-chain storage and retrieval of transaction semantic features. Furthermore, through the *Po2RW* consensus protocol, we balance the scalability and security of SemantiChain. Security analysis proves that SemantiChain can resist security risks such as man-in-the-middle attacks, malicious node attacks and on/off-chain data inconsistency. Experimental results demonstrate that SemantiChain can reduce search time and memory usage by at least 32.29% and 77.97% respectively under the same retrieval performance, compared to mainstream approximate nearest neighbour retrieval algorithms. Furthermore, compared to the SOTA semantic blockchain, SemantiChain achieves a retrieval performance improvement of at least 45.88% and reduces retrieval memory usage by 95.76%.**

***Index Terms*—Semantic Sharding, Blockchain, Information Retrieval, Scalability.**

I. Introduction

IN the era of big data, researchers widely use various types of data for scientific research. To obtain reliable research results, the reliability of the data is their primary concern. At the same time, the storage architecture needs to be able to withstand the risk of central server downtime and data loss. Nowadays, blockchain, due to its characteristics of data immutability and decentralization, has successfully overcome these issues and has been widely applied in fields such as agriculture [1]–[3], energy [4]–[7], and supply chains [8], [9].

However, as a trustworthy database, the data types stored in the blockchain are quite limited. In agricultural scenarios, the blockchain stores key geographical and temporal information about agricultural products from planting to sales in the form of text. Since the average block size of common blockchains is only 20KB to 2MB [10], for rich media such as images, audio, and video, the blockchain can only store them indirectly in the form of hyperlinks or textual identifiers [11]. This means that when farmers and experts need on/off-chain data for analysis and supervision of cultivation and sales processes, additional alignment operations must be performed first.

Meanwhile, after the data is packed into blocks in the form of transactions, these blocks are merely linked in chronological order in the blockchain, resulting in a very weak semantic relationship between transactions. To retrieve a specific type of transaction and its data, one always has to traverse the entire blockchain from the latest block. For Bitcoin and Ethereum, with current block counts exceeding 852K[1] and 20M[2] respectively, the cost of such traversal is extremely high.

While there are already some solutions that propose combining semantic features with blockchain to construct retrieval indexes, certain issues still persist. By modeling blockchain as a relational database [12], the semantic relationships between data are defined in terms of attributes, but this definition heavily relies on personal experience. On the other hand, using statistical methods like TF-IDF to extract semantics and build indexes [13] partially addresses the problem of defining data semantics, but these methods are often limited to text data and overlook the multiple meanings of words in different contexts. At the scale of big data, retrieval performance is constrained by these semantic representation defects.

On the other hand, due to the constraints of the consensus process, blockchain can only sacrifice its scalability in pursuit of security and decentralization. For example, for Bitcoin and Ethereum, their throughput can only reach 10 and 30 transactions per second [14]. This restricts the application of blockchain in high-throughput scenarios. Moreover, due to the immutability of data in the blockchain, continuously growing data increases the storage load of the full node gradually.

The existing solutions mainly enhance the scalability of the blockchain through on/off-chain methods. Off-chain methods do not essentially optimize the blockchain itself, while on-chain methods, especially the state sharding technology, have received widespread attention [15]. It divides the state information of the blockchain into multiple parallel sub-consensus areas, enhancing the scalability of the blockchain while reducing the storage of state information for full nodes. However, due

Zihang Zhen, Xiaoding Wang, Hui Lin are with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China. E-mails: zzihang@foxmail.com, wangdin1982@fjnu.edu.cn, and linhui@fjnu.edu.cn.

Xu Yang and Jiwu Shu are with the College of Computer and Data Science, Minjiang University, Fuzhou 350108, China. Jiwu Shu is also with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: xu.yang@mju.edu.cn, shujw@tsinghua.edu.cn.

Jia Hu is with the Department of Computer Science, University of Exeter, Exeter EX4 4QJ, U.K. E-mail: j.hu@exeter.ac.uk.

Xun Yi is with the School of Computing Technologies, RMIT University, Melbourne 3000, Australia. E-mail: xun.yi@rmit.edu.au.

[1]https://btc.com/btc/blocks

[2]https://etherscan.io/blocks

to the division of the sub-consensus area, the nodes are also divided into different consensus areas, and malicious nodes can aggregate and control the blockchain at a smaller cost. Therefore, it is necessary to build a reconfiguration strategy to periodically reassign nodes [16]. Although reconfiguration can enhance the security of the blockchain, it will also affect its efficiency. Current solutions often struggle to strike a balance between security and scalability.

According to the above analysis, we can conclude that the current blockchain faces the following key challenges in adapting to application scenarios that require accurate and efficient information retrieval:

1) Traditional blockchains cannot store various types of rich media, making it challenging to directly utilize data such as audio, images, and video.
2) The semantic relationships between transactions and data within the blockchain are very weak, making similar transactions and data hard to retrieve.
3) The existing methods involving semantic relationships and blockchain lead to inaccurate semantic representation of data.
4) The scalability bottleneck of blockchain limits its efficiency, and the increasing capacity of full nodes leads to high storage costs.
5) Blockchain sharding technology makes it difficult to strike a balance between security and scalability.

To tackle these challenges, we present in this paper a brand-new blockchain architecture called *SemantiChain*. It is a *Semantic Sharding Blockchain* that supports *Semantic Information Retrieval*. Furthermore, it enhances the security and scalability of the blockchain through specially designed consensus protocols and reconfiguration strategies. Our contributions can be summarized as follows:

- To enhance the semantic representation of data and improve the scalability of blockchain, we proposed, for the first time, a novel blockchain sharding technology called *Semantic Sharding*. This technology stores transactions in the form of semantic features and allocates them to different shards for parallel consensus based on semantic distance. Additionally, to liberate blockchain data storage from specific types, we implemented the *Semantic Sharding Blockchain* (SSB) based on Semantic Sharding, using structures like the *Semantic Tree*.
- To achieve a balance between security and scalability in blockchain, we designed a novel consensus protocol, *Proof of 2-Role Work* (Po2RW), which ensures both security and scalability in sharding blockchain by integrating reconfiguration strategies.
- For efficient retrieval of big data, we proposed a novel distributed *Semantic Information Retrieval* (SIR) structure tailored specifically for semantic sharding blockchain. Additionally, we have introduced an approximate nearest neighbour (ANN) retrieval algorithm aimed at reducing the time and space costs of retrieval, leveraging modules such as *Distribution Product Quantization* (DPQ) and *Coarse Filter*.
- The security analysis of *SemantiChain* demonstrates its effective resistance to man-in-the-middle attacks, malicious node attacks, and on/off-chain data inconsistency. Experimental results show that, compared to mainstream ANN retrieval algorithms, *SemantiChain* reduces memory usage by 77.97% and search time by 32.29% at the very least under the same performance metrics. In comparison to state-of-the-art blockchains, *SemantiChain* increases retrieval performance by at least 45.88% and reduces retrieval memory usage by 95.76%.

TABLE I
RELATED WORK ON BLOCKCHAIN DATABASE

Catalogy	Method	Data Representation	Scalable Technology	Transaction Retrieve
Blockchian System	Bitcoin [17]	Attributes	\	Backtracking
	Ethereum [18]	Attributes, Smart Contracts	Transaction Sharding	Backtracking
	BigchainDB [19]	Attribute, File Identifier	Off-chain Database	MongoDB
Distributed Database	IPFS [20]	File Identifier	\	\
Blockchian Database	SEBDB [12]	Attributes	Off-chain Database	SQL
	MSTDB [13]	Text Keywords	Off-chain Database	Multi-branch Tree
	SemantiChain	Semantic Feature	Semantic Sharding, Off-chain Database	Distributed ANN

The rest of this paper is organized as follows. Section II introduces the relevant work on blockchain databases and sharding blockchains. Section III presents the basic architecture of SemantiChain. Section IV details the semantic sharding technique, the *Po2RW* protocol, and the implementation of the SSB. Section V provides a detailed explanation of the SIR. In Section VI, the security analysis of SemantiChain is conducted. The experimental results and analysis of SemantiChain are presented in Section VII. Finally, Section VIII concludes this paper and outlines our future work.

II. RELATED WORKS

A. Blockchian Database

Due to the immutability of data in blockchain, blockchain is commonly regarded as a trusted distributed database. In 2009, Satoshi Nakamoto introduced Bitcoin [17], marking the beginning of blockchain applications. However, Bitcoin is limited to storing structured data related to transactions. To expand its application scope, Ethereum [18] was proposed, which not only stores transaction-related structured data but also allows developers to deploy decentralized applications through smart contracts. However, these solutions struggle with fast retrieval of on-chain transactions. To address the need for data storage and retrieval, new blockchain technologies like BigchainDB [19] were introduced. They tokenize stored data assets through transactions, utilizing file identifiers to link original data in off-chain databases, enabling storage and retrieval of various abstract and entity objects. Similarly, IPFS [20] as a distributed database offers content-addressed storage, storing diverse media data via file identifiers. However, these methods lack semantic representation of data content, making retrieval based on content correlation challenging. To tackle these issues, the concept of blockchain databases has evolved.

TABLE II
RELATED WORK ON SHARDING BLOCKCHAIN

Catalogy	Method	Reconfigure	Consensus Protocol	Cross-shard Tx Atomicity	Sharding Number
Transaction Sharding	RSCoin [21]	\	BFT	\	Fixed
	ELASTICO [22]	Random	BFT	No	Dynamic
State Sharding	OmniLedger [23]	Random	BFT	Yes	Dynamic
	Monoxide [24]	\	PoW	Yes	Fixed
	Brokerchain [25]	Graph	BFT	Yes	Fixed
Semantic Sharding	SemantiChain	Shard Activity	PoW	Yes	Dynamic

SEBDB [12], a relational blockchain database, allows users to define tables based on object attributes and use SQL queries to retrieve information. However, this attribute-based indexing heavily relies on personal experience. MSTDB [13] integrates semantic features with blockchain, using TFIDF to extract text keywords for data semantic representation and employing MST-B+ Tree for indexing. Despite overcoming human limitations, this approach is only suitable for textual data and requires re-indexing for other media types.

Therefore, as shown in Table I, for improving blockchain databases, SemantiChain aims not only to store various media data but also to enable content-based retrieval. Moreover, ensuring storage efficiency through on-chain/off-chain scalability technologies is crucial.

B. Sharding Blockchain

Sharding technology is one of the primary means to address the scalability of blockchain. Depending on the divided entities, sharding is categorized into network sharding, transaction sharding, and state sharding. Network sharding serves as the foundation for other sharding technologies. In 2015, RSCoin [21] introduced sharding into decentralized architectures similar to blockchain, enhancing transaction processing efficiency. That same year, addressing blockchain scalability issues, the transaction sharding-based architecture ELASTICO [22] was proposed. It disperses transaction loads across multiple sets for parallel consensus, improving blockchain scalability. However, due to its lack of consideration for atomicity in cross-shard transactions, OmniLedger [23] implemented a state sharding-based blockchain architecture. It divides blockchain states into different shards and ensures atomicity of cross-shard transactions through two-phase consensus. However, such solutions based on BFT-like consensus protocols are challenging to apply to permissionless chains. Monoxide [24], as a state-of-the-art architecture based on state sharding and PoW consensus protocol, balances decentralization, security, and scalability through finality and Chu-ko-nu mining. However, Monoxide's oversight on transaction load balancing leads to potential hot shard issues. To address this, Brokerchain [25], a state sharding blockchain architecture, uses graph policies to reduce cross-shard transactions and dynamically adjusts shard nodes. Nevertheless, these state sharding solutions fix the number of shards as a hyperparameter, making dynamic optimization difficult in changing environments.

Previous approaches have driven the development of blockchain sharding technology. However, as shown in Table II, these solutions still do not apply well to dynamic sharding architectures in permissionless chains. Traditional sharding technologies primarily focus on transactions between nodes, limiting their ability to optimize data storage and retrieval through data semantics in data storage applications. Therefore, this paper proposes semantic sharding technology based on state sharding.

III. SYSTEM MODEL

In this section, we present the architecture of *SemantiChain*. It is built upon the *SSB*, a blockchain architecture based on semantic sharding, and it uses Natural Language Processing techniques to extract semantic features from metadata. *SemantiChain* stores metadata off-chain and saves semantic features in the form of transactions on-chain, collectively improving the scalability of the blockchain through on-chain and off-chain mechanisms. Simultaneously, *SemantiChain* compresses semantic features and builds an index through the *SIR* framework, reducing the time and space costs of retrieval.

A. Architecture Overview

1) Member and Node Types: from a semantic perspective, there exist two types of users in *SemantiChain*, namely data owners and queriers. From the standpoint of blockchain consensus and reconfiguration, *SemantiChain* is composed of semantic nodes, relay nodes, and shard nodes.

- **Data Owners**: Data owners are data semantic nodes holding off-chain metadata. They can upload and update the semantic features of the metadata to the blockchain through transactions.
- **Data Queriers**: Data queriers are user nodes that wish to obtain verified correct query results and metadata. They can be any type of node in the blockchain network.
- **Semantic Nodes**: Semantic nodes (*S-Nodes*) are the basic units for transactions and consensus in SemantiChain, exchanging information through P2P networks. Based on whether they possess metadata, semantic nodes are further divided into data semantic nodes (*DS-Nodes*) and ordinary semantic nodes (*OS-Nodes*).
- **Relay Nodes**: Relay nodes (*R-Nodes*) essentially acts as a semantic node. In SemantiChain, R-Nodes participate in multiple shards simultaneously, possessing semantic feature information from various shards to facilitate cross-shard transactions and message forwarding.
- **Committee Nodes**: To avoid malicious node clustering and reduce cross-shard transaction occurrences, all committee nodes (*C-Nodes*) will form a committee shard (*C-Shard*) to manage and reconfigure the blockchain. C-Nodes will communicate with all other shards by P2P.

2) Basic Concepts and Definitions: Given SemantiChain's uniqueness, we offer new definitions for some fundamental concepts in the blockchain.

- **Toolkit**: The toolkit will be distributed to each node, consisting of (1) Coarse Filter, a tool for filtering the corpus range, which also serves as a selector for classifying transactions, and (2) Embedding Tool, which is the semantic representation model to extract semantic features from metadata such as text and images as semantic features. Only semantic features can be processed as data by SemantiChain.

- **Block**: In general, in SemantiChain, each block corresponds to a semantic feature of the metadata. However, due to the input token limit of embedded tools, when the metadata is too long, it will be segmented for semantic extraction. At this point, each segment of semantic feature will form multiple independent consensus transactions on the blockchain. A block at this time corresponds to a partial semantic feature of the metadata.
- **Transaction Legitimacy**: For SemantiChain, as there's no concept of balance, transaction legitimacy primarily occurs through verifying whether the current transaction's $Data_{id}$ already exists in the Semantic Transaction Tree. If the $Data_{id}$ exists, it's necessary to judge whether the data owner's address in Semantic Transaction Tree conflicts with the address of transaction creation node.

B. Execution Process

The data stored in SemantiChain needs to be uploaded or updated by DS-Node in the form of transactions. When other nodes need to obtain metadata, they can construct query statements to retrieve and obtain it from the SemantiChain.

- **Upload and Update Process**: (1) The data owner first obtains the unique identity of the data by performing a HASH digest on the local metadata, resulting in $Data_{id}$. (2) The data owner extracts the semantic feature v from the metadata using the embedding tool. (3) The corresponding DS-Node broadcasts the message body $M < Type, T_{id}, Addr_A, Data_{id}, v >$ in the form of a transaction to the shard, where Type represents the message type, T_{id} is the unique identity of the transaction, and $Addr_A$ is the address of the DS-Node. (4) Other semantic nodes within the shard determine whether the transaction is a cross-shard transaction using their local Coarse Filter. If it is an intra-shard transaction, it is directly verified and packaged onto the chain. If it is a cross-shard transaction, it is verified and then relayed by the relay node to the corresponding shard, where it is verified by the corresponding shard's semantic nodes and packaged onto the chain. (5) The Coarse Filter is updated at the next epoch.
- **Query Process**: (1) After the data querier sends a query request, the query content Q is first used to extract the semantic feature v_q using the embedding tool. Then, the local Coarse Filter is used to calculate the n nearest shards in terms of semantic distance for v_q. (2) The data querier's node broadcasts the message in the shard in the format of $M_Q < Type,\ IR_{id}, Addr_q, n, top_k, v_q >$, and relays the query message through the relay node and the shard node. Here, IR_{id} represents the retrieval ID, $Addr_q$ represents the node address of the data querier, and top_k represents the top_k query results. (3) Through parallel searches by the semantic nodes within each shard, the top-k retrieval results $O^{ni}_{top_k} < Data_{id}, \Delta, Addr_o >$ are returned to the data querier, where Δ represents the semantic similarity of the query results, ni represents the index of the semantic node in the reply, $Addr_o$ represents the node address of the corresponding data owner. (4) The data querier aggregates all $O^{ni}_{top_k}$ to obtain the top_k results, and obtains metadata in P2P networks using $Data_{id}$ as the file index.

C. System Design Objectives

The SemantiChain is mainly composed of the Semantic Sharding Blockchain architecture (SSB) and the Semantic Information Retrieval architecture (SIR). The security and scalability of its underlying blockchain architecture are the primary objectives.

- **Security**: It refers to the ability of blockchain to protect data from tampering, prevent attacks, and maintain stability.
- **Scalability**: It refers to the ability of blockchain to effectively improve throughput and processing speed while maintaining system performance and stability.

In semantic retrieval, it is essential to ensure the correctness, completeness, and orderliness of the retrieval results, while also controlling the time and space costs of retrieval.

- **Correctness**: Whether the expected retrieval results are returned;
- **Completeness**: Whether all retrieval results are returned;
- **Orderliness**: Whether more high-quality query results are returned in the retrieval results;
- **Time Overhead**: Retrieval response time and the time cost of building indexes;
- **Space Overhead**: Storage overhead of the retrieval index structure.

IV. The Implementation of Semantic Sharding Blockchain

This section primarily discusses the basic processes of the Semantic Shard Blockchain Architecture (SSB), which is inspired by MSTDB and based on our previous work DSSBD [26]. MSTDB is a recently proposed statistical-based blockchain semantic retrieval and storage solution. It uses TFIDF to extract keywords from on-chain data as semantic features, achieving an efficient blockchain database. This has inspired our research on the storage and retrieval of blockchain with semantic features. And DSSBD is a state sharding blockchain architecture, with the working cycle based on epochs. The workflow within each epoch is divided into a Blockchain Consensus stage and a Node Reconfiguration stage. It enhances scalability through parallel consensus and transaction relay, increases security by preventing the aggregation of malicious nodes through committee sharding, and models it as a Markov model to balance scalability and security through deep reinforcement learning.

Built on DSSBD, SSB also uses the epoch as its working cycle. However, as each block in SSB contains only one

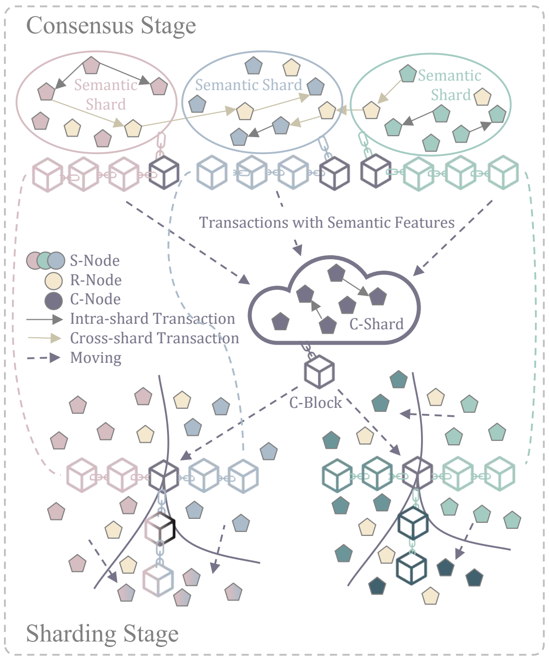

Fig. 1. The main execution stages and architecture of SSB

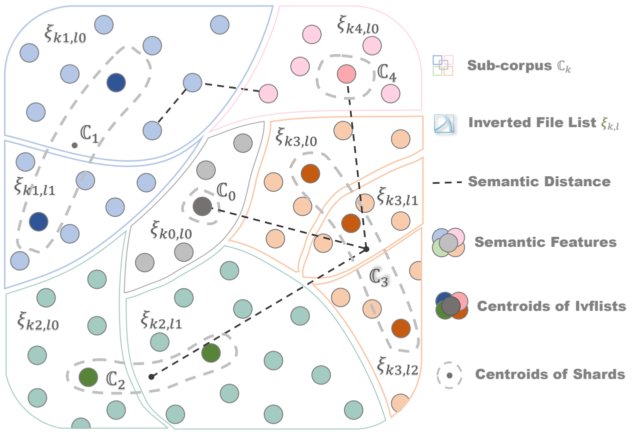

Fig. 2. Strategies for the Splitting and Merging of Semantic Shards

transaction, its workflow includes multiple alternating consensus and sharding stages. Since semantic features support data retrieval at a large data scale, in order to integrate semantic features, SSB enhances execution strategies, modifies data structures, and balances scalability and security through a new consensus protocol, realizing a blockchain architecture that uses semantic features as the basis for sharding.

As shown in Figure 1, within the Consensus Stage, multiple consensus blocks are formed within each shard. Simultaneously, C-Nodes of C-Shard connect to the subnets of each shard through a P2P network, listening to transactions of the longest valid chain in each shard. C-Nodes split and merge shards based on sharding strategies and reassign node's shard membership according to reconfiguration strategies and the results of reconfiguration. Consensus blocks, C-Blocks, are then formed in C-Shard based on the results and broadcasted to all nodes.

During the Sharding Stage, each shard merges and splits based on C-Blocks. Blockchain nodes enter new shards based on C-Blocks and reconstruct the sub-databases and indexes within the current shard based on the consensus blocks.

A. *Strategy for Sharding Stage*

1) Initial Sharding Strategy: To prevent Sybil attacks [27], any new node joining the SSB needs to solve a hash puzzle to calculate its index for joining the shard. Once a node has joined the shard, data synchronization is carried out through the P2P network.

For the corpus, as shown in Figure 2, the Coarse Filter initially divides the current corpus into k semantically distinct sub-corpora $\mathbb{C}_k$, which correspond to K semantic shards S_k. To accelerate retrieval speed, the semantic features in $\mathbb{C}_k$ are divided into L_k invert file lists $\xi_{k,l}$, also known as $ivflist$. During retrieval, an $ivflist$ $\xi_{k,l}$ is selected in the form of key-value pair $< k, \xi_{k,l} >$, and then traversed to obtain the retrieval results. Therefore, the value of k is generally several orders of magnitude lower than l. In the initial state, the current corpus is directly clustered and divided.

2) Corpus Sharding Strategy: In the SSB, semantic shards are arranged based on semantic distance, with shard activity serving as the criterion for shard splitting and merging. The activity of semantic shards is reflected through the sub-throughput τ_k and the number of semantic features ε_k in each shard. The SSB dynamically splits and merges the corpus using C-Shard and Coarse Filter. Based on shard activity, we define the basic concepts of active shards and dormant shards as follows.

- **Active Shard**: In each epoch, if the sub-throughput of a semantic shard $\tau_k > \tau_{max} \times \eta_{upper}$ or the number of semantic features $\varepsilon_k > \varepsilon \times \eta_{max}$, where τ_{max} is the theoretical maximum throughput of each shard in the current epoch, ε is the current corpus data quantity, and η_{upper} and η_{max} are known as the splitting factor and overload factor, it indicates that the semantic shard has a high frequency of legitimate transactions in the current epoch, or the $\mathbb{C}_k$ of this shard contains a large amount of corpus data. Such semantic shards are referred to as active shards.
- **Silent Shard**: In each epoch, if the sub-throughput of a semantic shard $\tau_K < \tau_{max} \times \eta_{lower}$, where η_{lower} is referred to as the merging factor, it indicates that the frequency of transactions in the shard has become slow, and the shard is no longer sufficiently active.

Specifically, the values of τ_{max} and ε are calculated and statistically determined based on consensus results. $\tau_{max} = \frac{n_{tx}}{K \times T_{epoch}}$, where n_{tx} is the theoretical maximum number of transactions in this epoch. Since each block in SSB only records one transaction, n_{tx} is equivalent to the theoretical maximum number of blocks produced in this epoch, that is, the number of blocks produced that each shard can continuously reach consensus within the epoch; K is the number of shards in this epoch, and T_{epoch} is the execution time of this epoch.

η_{upper}, η_{lower}, and η_{max} are adjustable hyperparameters, with values ranging from 0 to 1. If frequent shard splitting or merging occurs during an epoch due to excessively high or low transaction throughput, η_{upper} should be increased or η_{lower} should be decreased accordingly to reduce the corresponding operations. If frequent shard splitting occurs due to uneven data distribution, η_{max} should be appropriately increased to improve the shards' tolerance to the degree of uneven distribution.

As shown in Figure 1, at the end of each round of the consensus stage, C-Shard obtains information about active or dormant shards. Based on this, it generates a committee block, C-Block, to guide the shard stage of the current round.

If a shard becomes an active shard, it needs to be split into two new semantic shards based on semantic similarity during the shard stage. The transactions within the original shard and the cached transactions in the corpus transaction pool will be allocated to the new shards based on semantic similarity, and the Coarse Filter will be updated. C-Shard forms a consensus result, C-Block, for the shards that need to be split during the consensus stage. The C-Block links the blocks in the original shard, and in the next round's epoch consensus stage, the blocks of the new shard consensus will continue to be linked from the C-Block.

If a shard becomes a dormant shard, it means that the number of transactions and blocks in this shard during the current epoch is far below the theoretical value. This will lead to a reduction in the incentive for block generation within the shard, and a decrease in the willingness of consensus nodes to participate in this shard. At this point, these inactive dormant shards need to be merged. When merging dormant shards, C-Shard considers the semantic distance between the centroids of the shards and selects the semantic shards with the closest semantic distance to merge, in order to ensure the similarity of semantics in the new shard. The result is also formed into a consensus C-Block in the form of transactions, and added to the new shard. The forward pointer of the C-Block points to the two shard chains before the merge, linking historical transactions.

After the new shards are constructed, although the transactions from the old shards are linked to the new shards in the form of blocks, the new shards need to filter historical transactions based on the C-Block. The semantic features, i.e., transactions, allocated to the current shard are used to locally construct retrieval indexes on the nodes, generating $ivflist$s and the centroids of each $ivflist$. C-Shard updates the Coarse Filter based on the centroids of each shard and broadcasts it to the shards.

3) Node Reconfiguration Strategy: In blockchain, when a transaction is initiated through intra-shard transactions, the transaction can directly achieve consensus and be added to the chain within the current shard, resulting in the lowest communication cost for the transaction process. Otherwise, the transaction needs to be forwarded to the destination shard via relay nodes, increasing communication costs. Additionally, for sharded blockchains, honest nodes are typically evenly distributed among the shards, while malicious nodes tend to aggregate in specific shards to influence the blockchain by affecting the shards. In order to reduce the communication cost of data being added to the chain and ensure the security of the blockchain, it is necessary to reassign nodes to the shards within the blockchain network in each epoch.

To ensure the security of the blockchain, in each epoch, all nodes reconfigure the shards by solving HASH puzzles and selecting the remainder of the results. However, DS-Nodes, which are the initiators of transactions, often frequently modify files within a short period or focus on a specific domain for a certain period of time, resulting in semantic similarity among the transaction data uploaded within a specific time frame. To reduce the number of cross-shard transactions, $p_{reconfig}$ is set as a threshold to consider whether the DS-Node is subject to its tendency when reconfiguring. If not, the result of solving the HASH problem is still used as the basis for reconfiguration. For other nodes, it is only necessary to use the reconfiguration strategy to evenly distribute the nodes in the blockchain.

During the SSB Sharding Stage, C-Shard will read the on-chain transactions of each shard within an epoch, and record the initiator node address and the shard to which each transaction belongs. In each round of reconfiguration, nodes select the shard with the most transactions as the trending shard S^{I}_{trend} for the DS-Node, where I represents the I of the reconfiguring node. For DS-Nodes that did not initiate any transactions in the current epoch, the sharding committee will continue to use the previous trending shard as their S^{I}_{trend}. During reconfiguration, S^{I}_{trend} is used as the candidate for the node's reconfigured shard, and with probability $p_{reconfig}$, it is selected as the reconfiguration result S^{I}_{reconf}, i.e., $S^{I}_{reconf} = S^{I}_{trend}$. With a probability of $(1-preconfig)$, a shard is randomly selected as the reconfigured shard, i.e., $S^{I}_{reconf} = random(k)$.

Finally, C-Shard modifies the belonging shards of these nodes in the semantic state tree and broadcasts the semantic state tree mapped to the C-Block to all shards.

B. Strategy for Consensus Stage

1) Consensus Protocol: In SSB, DS-Nodes exhibit a tendency to choose shards, while DS-Nodes are evenly distributed across the shards. For sharded architectures, in order to prevent malicious nodes from masquerading as DS-Nodes by aggregating their tendencies towards specific shards, and thereby compromising security, we propose a Proof of 2 Role-based Work consensus protocol based on the PoW consensus protocol.

Similar to the PoW protocol, Po2RW is also based on proof of work and has the same consensus process. In theory, it has a similar communication cost to PoW. However, the difference is that for different types of block-producing nodes, the difficulty requirements of Po2RW when producing blocks are inconsistent.To achieve this, in addition to the difficulty coefficient $p_{difficult}$, the block header in Po2RW needs to include a role difficulty coefficient p_{role} to adjust the difficulty of proof-of-work for different roles. Then the result judgment of the consensus solver HASH in Po2RW is corrected to $p_{role} \times p_{difficult}$. The role difficulty coefficient of role A defaults to 1, and the coefficient of role B refers to the block's p_{role} field. Therefore, the computational overhead of role A is the same as in the PoW protocol, while the computational overhead of role B is $1/p_{role}$ times that of the PoW protocol.

In SSB, we set the default role difficulty coefficient of DS-Nodes to 1, while the role difficulty coefficient of DS-Nodes is set to p_{role}. This is because during node reconfiguration, OS-Nodes are always randomly assigned to shards, while DS-Nodes have the potential to cluster into specific shards due to shard inclination. In such shards, an excessive number of clustered DS-Nodes can control the shard consensus results. To mitigate potential risks, SSB weakens the computing power of DS-Nodes by setting p_{role} to a value less than 1, thereby enhancing the influence of OS-Nodes within the shards and balancing the advantage brought to DS-Nodes by shard inclination.

In SSB, the Po2RW consensus protocol is closely linked with the Node Reconfiguration Strategy. By continuously adjusting p_{role} and $p_{reconfig}$, it achieves a dynamic balance between security and efficiency. Derived from PoW, the Po2RW consensus protocol also demonstrates resistance to common security attacks in blockchain systems, as detailed in VI.

2) Semantic Tree: In order to determine the shard of each node in the blockchain when validating transactions and receiving information, as well as to combine Po2RW to verify the semantic node type during consensus, we modify the state tree to a Semantic State Tree. In comparison to the state tree, the leaf nodes of the Semantic State Tree will include an additional field *Shard#* of length k^{max} bits and a 1-bit role field *isData*. Here, k^{max} represents the maximum number of shards in the SSB, indicating the shard to which the node belongs, while the isData field is used to distinguish between data semantic and DS-Nodes. Notably, as the R-Node retains the information of multiple shards, multiple bit positions of Shard# are set to 1, and the isData field maintains consistency across all shards based on whether the node possesses data.

The local semantic state tree will retain the public key θ and shard information of the nodes in the same shard, and update their node type based on the consensus within the shard. However, for nodes not in the same shard, the Semantic State Tree only retains their shard information and uses a regular pointer to point to the account, without serving as a validation node for the Merkle tree. For relay nodes located in the same shard, the Semantic State Tree similarly retains all their information and updates accordingly.

Meanwhile, to rapidly verify the legitimacy of transactions and obtain metadata from nodes, we construct a Semantic Transaction Tree in the SSB. The semantic transaction tree reads transactions from the blocks of the current shard, forming a practical Merkle tree with $Data_{id}$. Its leaf nodes correspond to semantic features, and the content of the leaf nodes mainly consists of the data owner's address for that semantic feature, as well as the shard in which the semantic node resides.

3) Cross Shard Transaction: Since the entire corpus is divided into different shards, different shards cannot communicate directly. Cross-shard transactions need to be divided into two related intra-shard transactions, and consensus is reached in each shard based on Po2R2W. As illustrated in Figure 3, this is a typical example of a cross-shard transaction.

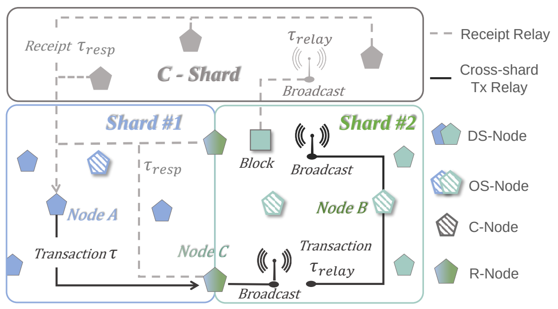

Fig. 3. Example of Cross-Shard Transaction Process

First, DS-Node A in Shard #1 creates an original transaction $\daleth$. Node A needs to extract the semantic feature of the raw data, validate the off-chain original file, and sign the original transaction. To facilitate the upload of the transaction by light nodes, node A directly broadcasts the original transaction $\daleth$ within Shard #1 in the following format:

$$\daleth = << M >, \sigma_A > \tag{1}$$

where σ_A represents the digital signature of node A using the private key for the transaction content M.

Upon receiving the original transaction $\daleth$ via the P2P network within Shard #1, the semantic node first uses a Coarse Filter to determine whether $\daleth$ is a cross-shard transaction. For in-shard transactions, the semantic node directly validates the transaction using node A's public key in the local Semantic Transaction Tree and then packages it into a block. For cross-shard transactions, only relay nodes with functionality in both Shard #1 and Shard #2 have sufficient information to process them. Upon receiving $\daleth$, such a R-Node C creates a relay transaction $\daleth_{relay}$ in Shard #2 and broadcasts it, represented as follows:

$$\daleth_{relay} = << T'_{id}, Type, Addr_C, \daleth >, \sigma_C > \tag{2}$$

where T'_{id} represents the transaction ID in Shard #2, and σ_C represents the digital signature of C on the transaction.

Upon receiving the relay transaction $\daleth_{relay}$ via the P2P network within Shard #2, the semantic nodes utilize the public keys from the Semantic State Tree and the Semantic Transaction Tree to validate the transaction and include it in the blockchain. At this point in Shard #2, T_{id} is incremented, and the Semantic Transaction Tree is updated.

After the successful inclusion of the transaction in the blockchain in Shard #2, both the shard nodes in the committee shard and the relay node will send confirmation receipts to the DS-Node A. The format of the confirmation receipt $\daleth_{resp}$ is as follows:

$$\daleth_{resp} = << T_{id}, Type, Addr, \daleth >, \sigma > \tag{3}$$

Node A needs to obtain confirmation receipts from both the C-Shard and the R-Nodes in order to trust that the transaction has been successfully included in the blockchain. However, due to information loss, node A can still choose not to trust the success of the cross-shard transaction and continue to send cross-shard transactions. However, since in Shard #2, the raw data in the transaction has already been updated in the Semantic Transaction Tree, the new transaction will not be successfully validated, and node A will not receive a confirmation receipt.

V. The Implementation of Semantic Information Retrieval

This section mainly introduces index construction and semantic feature retrieval in the Semantic Information Retrieval (SIR) architecture. Inspired by FAISS [28], we model the retrieval of semantic information as an approximate nearest neighbour search problem and reduce the cost of index construction in a distributed manner.

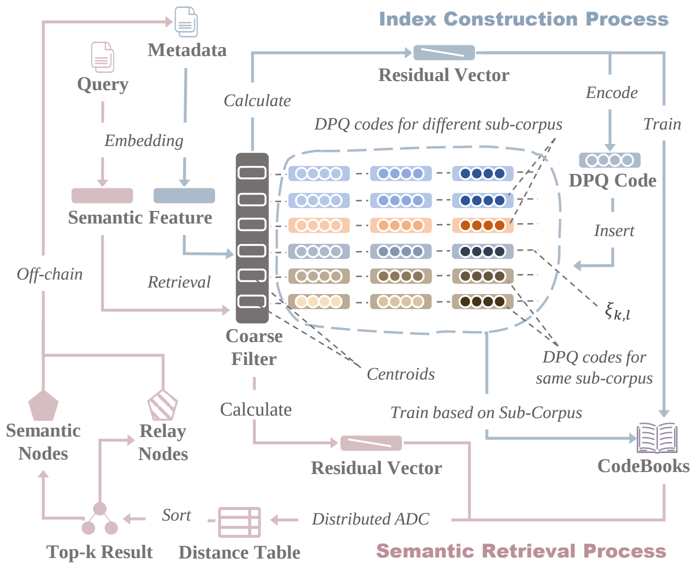

Fig. 4. The Main Execution Stages and Architecture of SIR

The indexing construction process, as shown in Figure 4, begins with the data owner embedding the metadata to obtain its semantic features. Based on the current Coarse Filter, the semantic features are categorized into the corresponding $ivflists$ $\xi_{k,l}$ and sub-corpora $\mathbb{C}_k$, along with the centroids $\xi^c_{k,l}$ corresponding to $\xi_{k,l}$. The Coarse Filter is updated in each epoch based on the current data of each sub-corpora. The residual vector is obtained with $\xi^c_{k,l}$ as the reference for the current sub-corpora, and the residual vector undergoes DPQ for the current sub-corpora to obtain the DPQ code of the semantic feature. The $Data_{id}$ and DPQ code of the semantic feature are is inserted at the end of $\xi_{k,l}$.

The semantic retrieval process, as depicted in Figure 4, begins with the data querier embedding to obtain its semantic features based on the current Coarse Filter, determining the $ivflists$ $\xi_{k,l}$ to which the semantic features belong. Specifically, to improve retrieval performance, multiple $\xi_{k,l}$ are obtained during retrieval, and results are searched for in multiple $ivflist$s. For any $ivflist$, the semantic feature of the Query first needs to compute the residual vector with $\xi^c_{k,l}$. The residual vector is used to calculate the Distance Table with the codebook. Each shard sorts the retrieval results based on the Distance Table and ultimately returns the top-k retrieval results to the data querier. Finally, the data querier obtains metadata from the data owner's off-chain database through P2P networks.

A. Space Compression

Regardless of whether it is for images or text, the semantic features extracted through embedding are often high-dimensional sparse vectors, and the storage cost of constructing a corpus with such vectors is enormous. From the perspective of blockchain, this also imposes excessively high memory requirements on nodes.

1) Quantization: The goal of vector quantization is to encode source data, which has a probability density function, into as few bits as possible (i.e., low rate), allowing the data to be recovered from the bit representation with the highest possible quality (i.e., small average distortion). This is a destructive process that causes information loss.

To balance the quantization rate and average distortion, the Product Quantizer (PQ) [29] was proposed as an optimization algorithm. It truncates all source vectors x in the corpus into j groups of sub-vectors x_j. Each group of sub-vectors is clustered separately to form PQ codes, which are used to represent x in the corpus, achieving a high quantization rate. Furthermore, a codebook is utilized to store the relationship between the cluster centroids and PQ codes, thereby maintaining a low average distortion.

However, on a large data scale, performing PQ on the entire corpus is extremely costly and unfavourable for distributed retrieval. Based on the Product Quantizer, we propose a Distributed Product Quantizer (DPQ) scheme for SIR applications. As illustrated in Figures 4, the data in the corpus is first pre-classified by a Coarse Filter based on semantic similarity into K subsets, e.g., K semantically similar sub-corpora $\mathbb{C}_k$. Then, each sub-corpus $\mathbb{C}_k$ is locally quantified in parallel using the Product Quantizer to form DPQ codes, and local codebooks are trained to store the centroid vectors of each cluster.

In SIR, the process of forming DPQ codes is primarily achieved through K-means. This is because the simplicity and efficiency of K-means make it suitable for large-scale data. Additionally, K-means clusters data by minimizing the Euclidean distance between them, which aligns with the goal of DPQ to minimize the distance between the corpus source data and the centroid vectors in the codebook. In DPQ, the number of cluster centroids for each subvector x_j is 2^{nbit}, and the length of the subvector x_j is $nsegment$.

2) Decode: To determine the semantic similarity between the query data v and the target data x, the retrieval process requires decoding DPQ encoding. In the Semantic Information Retrieval (SIR) framework, v is first filtered through the Coarse Filter to select multiple similar sub-corpora $\mathbb{C}_k$. Since DPQ quantization essentially approximates all local source data to the vector coordinates of cluster centroids, and the codebook stores the vector coordinates of these local cluster centroids, v is computed in parallel with the codebook in each $\mathbb{C}_k$ to obtain a local distance table. By comparing the distance tables, the x with the closest semantic relationship can be quickly found.

B. Index Construction

1) Inverted File System: The inverted file system organizes all corpus data in accordance with predefined indexing rules, thereby streamlining the retrieval process, and making it a popular choice in retrieval systems. The Semantic Information Retrieval (SIR) system also employs an inverted file system structure, primarily comprising the inverted index and inverted list substructures. In SemantiChain, each sub-corpus is segmented into multiple $ivflist$s based on semantic distance. Consequently, in SIR, the inverted list consists of $ivflist$s, while the inverted index is formed by the centroids of each $ivflist$.

In order to reduce the cost of retrieval, the inverted index is structured as a dictionary, with the key being the centroids of each $ivflist$ and the value being the $ivflist$s and its corresponding shard. The inverted list is formed by linking the entries of the $ivflist$s in the form of a linked list. In SIR, each entry corresponds to a semantic feature, so each entry is composed of the unique $Data_{id}$ and DPQ code for that semantic feature. Based on the judgment result of the Coarse Filter, each entry is inserted into the corresponding inverted list to build the inverted file system.

During retrieval, the inverted index is obtained based on the judgment result of the Coarse Filter for the query vector. The corresponding inverted list is traversed, and the semantic distance is calculated for each inverted list entry to obtain the top_k results.

2) Coarse Filter: To achieve non-exhaustive semantic retrieval, we have constructed a Coarse Filter to filter the retrieval scope of the corpus. The Coarse Filter is primarily based on the Inverted file system. As semantic information is a high-dimensional sparse vector, the main task of the Coarse Filter is to use clustering algorithms, such as K-means, to form l clusters from the current corpus, which are referred to as $ivflist$s in SIR. The index numbers of the clustering results serve as the inverted index in the Inverted file system, and the inverted list corresponds to the $ivflist$s formed by the clustering.

In SIR, in order to accurately cluster semantic features, the local Coarse Filter needs to be updated periodically with an epoch. Transactions are stored in the form of blocks in each shard based on semantic distance, allowing each shard to update the Coarse Filter in parallel. At the end of an epoch, all transactions of the round have been recorded on the chain, and each shard reads transactions from the blocks. Within each shard, transactions are allocated to new $ivflist$s in a clustered manner, and centroids for the clusters are obtained. Subsequently, each shard sends its $ivflist$ centroids to the C-Shard via a P2P network. Nodes in the C-Shard aggregate the $ivflist$ centroids from different shards, update the inverted index of the Coarse Filter, and distribute the updated Coarse Filter to each shard as a toolkit, initiating a new round of epoch cycles. The results of the Coarse Filter update directly impact the consensus of transactions, and its security is ensured by blockchain consensus. However, this clustering method may misjudge the positions of semantic features on the cluster edges during retrieval. Therefore, this paper employs the cell-probe method to simultaneously search for multiple $ivflist$ near the target cluster, aiming to improve retrieval accuracy.

TABLE III
NODE TYPE OF SECURITY ANALYSIS OF THE CONSENSUS PROTOCOL

Abbreviation	Meaning
n_{all}	Total number of semantic.
n_o	Number of honest OS-Nodes.
n_d	Number of honest DS-Nodes.
n_m	Number of malicious nodes.
n_{mo}	Number of malicious OS-Nodes.
n_{md}	Number of malicious DS-Nodes.

VI. Security Analysis

In this section, we will discuss the risks posed by semantic scalability blockchain from a security perspective, including man-in-the-middle attacks, on/off-chain data inconsistency, and the security of the Po2RW consensus protocol. Details are as follows.

A. Security Analysis of the Consensus Protocol

To demonstrate the security of the Po2RW consensus, we assume that each node has equal computational power and that the distribution of honest nodes is uniform. The definitions of relevant terms are provided in Table III.

1) Security Analysis of Sharding Blockchain in PoW Consensus Protocol: In a blockchain using the PoW consensus mechanism, security is compromised when malicious nodes control more than half of the total computational power, known as a 51% attack. Given uniform computational power among nodes, the number of malicious nodes needed to control the blockchain is:

$$n_m > n_o + n_d \tag{4}$$

In sharded systems, consensus is achieved within each shard. Hence, malicious nodes only need to control a specific shard to compromise the entire blockchain. The number of malicious nodes required in this scenario is:

$$n_m > \frac{n_o + n_d}{K} \tag{5}$$

where k is the number of shards.

To mitigate security risks in sharded blockchains, nodes are randomly reconfigured after each epoch by solving a hash, thereby increasing the number of malicious nodes to the limit defined by Eq. (4). Consequently, for a sharded blockchain with a PoW consensus protocol, Eq. (4) and Eq. (5) represent the upper and lower bounds on the number of blocks controlled by malicious nodes.

2) Security Analysis of Sharding Blockchain in Po2RW Consensus Protocol: In SSB's node reconfiguration strategy, DS-Nodes undergo random reconfiguration, while DS-Nodes tend to select shards based on their owned data. Malicious nodes can exploit this by aggregating in specific shards. However, by adjusting the values of $p_{reconfig}$ and p_{role}, we can achieve a balance between the security and efficiency of the blockchain.

For malicious nodes, if they are all DS-Nodes, their distribution across shards is uniform during node reconfiguration. due to the random allocation of DS-Nodes. Furthermore, since the proof-of-work difficulty for DS-Nodes is p_{role} times that of DS-Nodes, the number of malicious nodes required to control the blockchain is given by:

$$n_m > n_o + p_{role} \times n_d \tag{6}$$

Since $p_{role} \leq 1$, by adjusting p_{role} to 1 in extreme cases, we can sacrifice efficiency to achieve security equivalent to Eq. (4), thereby causing the Po2RW consensus protocol to degrade to a PoW consensus protocol.

Additionally, malicious nodes can only aggregate in shards after becoming DS-Nodes. If all malicious nodes are DS-Nodes and honest nodes are uniformly distributed, the number of malicious nodes required to control the blockchain is given by:

$$n_m > \frac{n_o + p_{role} \times n_d}{p_{reconfig} \times p_{role} \times K} \tag{7}$$

Since $p_{reconfig} \leq 1$, in an ideal scenario, by adjusting p_{role} to 1 and $p_{reconfig}$ to $1/k$, all consensus nodes undergo random reconfiguration, achieving security equivalent to Eq. (4).

From the perspective of malicious nodes, the optimal and most common approach is to have $n_m = n_{mo} + n_{md}$. In this case, the number of malicious nodes required to control the blockchain is given by:

$$n_{mo} + p_{reconfig} \times p_{role} \times K \times n_{md} > n_s + p_{role} \times n_d \tag{8}$$

Similarly, adjusting $p_{reconfig}$ and p_{role} can achieve a balance between security and efficiency.

B. Security Analysis of Man-in-the-Middle Attacks

Due to the fact that nodes within each shard in SemantiChain are connected through P2P networks, nodes from different shards do not communicate directly. Therefore, SemantiChain implements cross-shard transactions through relaying, which introduces the possibility of a man-in-the-middle attack.

1) Security of Cross-shard Transaction Relay: As shown in Figure 3, in SemantiChain, when node A initiates a legitimate transaction $\tau << M >, \sigma_A >$, and node C obtains semantic feature information M from τ. If node C is a malicious node and does not relay τ to form τ_{relay}, but instead initiates a new transaction $\tau_{new} << M >, \sigma_C >$, as the owner of M is A, τ_{new} is not considered a legitimate transaction in SemantiChain. However, τ_{new} may still be considered a legitimate transaction and added to the blockchain of shard #2.

However, since SemantiChain has multiple R-Nodes between the two shards, it is not solely dependent on node C. In a blockchain architecture, honest shard nodes always constitute the majority. Therefore, after obtaining τ, the nodes that relay τ_{relay} in shard #2 still constitute the majority, and the efficiency of broadcasting τ_{relay} in shard #2 far exceeds that of τ_{new}. Consequently, the majority of nodes in shard #2 also validate the legitimacy of τ_{relay}, ultimately ensuring the security of cross-shard transactions relaying through node consensus.

2) Security of Transaction Confirmation Receipt Propagation: After the cross-shard transaction is added to the blockchain, the R-Nodes will send a confirmation receipt to node A to inform it of the transaction's outcome. However, due to the lack of direct communication between shard #1 and shard #2, A cannot directly confirm whether the transaction has been genuinely validated in shard #2.

Similarly, as shown in Figure 3, after node A initiates a legitimate transaction $\tau << M >, \sigma_A >$, node C can immediately forge a transaction confirmation receipt $\urcorner_{resp}$ to instruct A to stop sending τ. However, in SemantiChain, for the same reasons, multiple R-Nodes exist between the two shards, and $\urcorner_{resp}$ is also forwarded to node A through C-Node. Therefore, despite node C's attempt to stop the sending of τ, node A can still determine whether it is necessary to continue sending τ based on the number of received $\urcorner_{resp}$. This ensures the security and reliability of transaction confirmation receipts.

C. Security Analysis of On/Off-chain Data Inconsistency Attacks

The blockchain system SemantiChain improves its scalability through a combination of on-chain and off-chain methods. Consequently, the fundamental issue faced by data queriers is whether the semantic features stored in the blockchain correspond to the metadata in the off-chain database.

In SemantiChain, data queriers first obtain the metadata corresponding to $Data_{id}$ through P2P networks. The data queriers perform a hash verification on the metadata to obtain H_{meta}, and compares it with $Data_{id}$ for consistency. Since the queriers possess the address of the data owner, if the metadata is obtained from the data owner's off-chain database, weak validation of the metadata can be obtained at this point. When the data owner initiates a transaction, the correspondence between $Data_{id}$ and the semantic features of the metadata is not verified in order to reduce the cost of node consensus. Therefore, weak validation can only resist data inconsistency attacks when the data owner is trustworthy.

To fully resist data inconsistency attacks, data queriers can also extract semantic features from the metadata locally and calculate the semantic similarity with the query vector. If the semantic similarity difference in the query results is within an extremely small range, strong validation of the metadata is achieved. Metadata that passes strong validation can prove its correspondence with the semantic features on the blockchain.

D. Security Analysis of Other Attacks

The unique resistance to man-in-the-middle attacks arising from sharding technology, resistance to Sybil attacks and 51% attacks brought by the Po2RW consensus protocol, and the resistance to data inconsistency attacks resulting from the on-chain and off-chain scalability enhancement techniques have been discussed above. Additionally, SemantiChain functions as a data storage architecture without the concept of balance, rendering economically motivated attacks like double spending meaningless. As SemantiChain remains a blockchain-based architecture, its security is inherently ensured by the blockchain itself, providing natural resistance against risks such as data tampering and replay attacks.

VII. Experiment

A. Basic Settings

1) Environment: We constructed a SemantiChain simulator, which runs on a 64-bit Linux server (Ubuntu 20.04) equipped with an Intel Core(TM) i9-12900 CPU, NVIDIA RTX A5000, and 64GB of memory. Additionally, we set the block interval time for the SemantiChain simulator to be 0.1s, the number of blocks per epoch to be 1000, and the transaction broadcast rate to be 1000tx/s.

2) Datasets and Models: This experiment selected QUORA [30], MSMARCO [31], and Stanford Online Products(SOP) [32] as experimental datasets. QUORA is based on real data from the QUORA Question Answering site, with a corpus of approximately 520K pieces of text and 10K questions. MSMARCO, proposed by Microsoft and based on Bing, contains approximately 8M text and 1K questions. To simplify the experiment, we extracted the first 10% of the corpus from MSMARCO as the experimental corpus. Stanford Online Products includes 120K product images and 22K categories. We selected one product image from each category as the corresponding question, with other images serving as the experimental corpus. Since the process of semantic feature extraction is not the focus of this paper, we used pre-trained models all-mpnet-base-v2 [33] and clip-ViT-L-14 [34] to map the data into the semantic space.

3) Benchmark: For the retrieval experiments, we selected some commonly used ANN retrieval algorithms. The brute force algorithm is the most naive approach, without index construction and vector quantization. HNSW [35] is a graph retrieval scheme based on the theory of six-dimensional space, with index construction. The PQ [29] scheme does not involve index construction, only vector quantization. IVFPQ [29] involves both index construction and vector quantization through an inverted file list. The FAISS scheme combines the advantages of HNSW and IVFPQ, optimizing both time and space, and is an advanced ANN solution.

Furthermore, from the perspective of blockchain, MSTDB is the SOTA semantic blockchain, which uses TD-IDF and semantic Merkle trees to store and retrieve semantic data, and Monoxide is the SOTA state sharding blockchain. We use them as the baseline for the experiment.

4) Evaluation Metrics:

- **Recall**: The ratio of the number of retrieved relevant semantic features to the actual number of relevant semantic features reflects the ability of the retrieval system to find relevant documents.
- **Precision**: The ratio of the number of retrieved relevant semantic features to the total number of retrieved features reflects the performance of the retrieval system.
- **NDCG**: Normalized Discounted Cumulative Gain, represents the normalized cumulative benefit of the top-k positions with the inclusion of position information.
- **MAP**: Mean Average Precision measures the system's accuracy and order of returning relevant documents by calculating the average precision of all retrieval queries.
- **Time**: The Average Search Time (ms) refers to the duration required to retrieve. The Index Construction Time (ms) denotes the duration needed for index creation by retrieval systems, also measured in milliseconds.
- **Memory**: The memory usage during retrieval by different retrieval algorithms, is measured in megabytes.

B. Experiment of Malicious Node Resistance

1) Experimental Setup: We conducted experiments to verify the security of the Po2RW protocol under different values of $p_{reconfig}$, p_{role}, r_o, and r_d. Here, $p_{reconfig} = \{0, 0.05, 0.1, 0.5, 1.0\}$, $p_{role} = \{0, 0.01, 0.1, 1.0\}$, $r_o = \{0, 0.1, 0.2, ..., 1.5\}$ represents the ratio of malicious OS-Nodes to honest ones, and $r_d = \{0, 0.1, 0.2, ..., 1.5\}$ represents the ratio of malicious DS-Nodes to honest ones.

We fixed the number of shards in the blockchain at 32, with 10,000 honest OS-Nodes and 10,000 honest DS-Nodes. Initially, all nodes were evenly distributed among the shards. Honest nodes followed the reconfiguration rules, while malicious nodes attempted to concentrate in the same shard as much as possible. If the computational power of malicious nodes within a shard exceeded that of honest nodes, the blockchain was considered to be under the control of malicious nodes. For each parameter combination, we conducted 1000 reconfigurations to calculate the probability of the blockchain successfully resisting control by malicious nodes under various parameter combinations.

2) Experimental Evaluation: The experimental results are shown in the left part of Figure 5. For each row, p_{role} remains unchanged, and for each column,$p_{reconfig}$ also remains the same. It can be observed that as $p_{reconfig}$ increases, the tolerance of SSB towards malicious nodes gradually decreases, primarily towards malicious DS-Nodes. This is because, during the node reconfiguration phase, a higher $p_{reconfig}$ makes it more likely for the SSB to follow the shard tendency of the DS-Node, while the reconfiguration of the OS-Node is always random, maintaining a uniform distribution. This tendency makes it easier for malicious DS-Nodes to cluster together, thereby reducing SSB's resistance to malicious nodes.

For each column, as p_{role} increases, although SSB's resistance to DS-Nodes gradually decreases, its resistance to OS-Nodes actually increases. This is because Po2RW indirectly adjusts their computational power by controlling the mining difficulty coefficient of block production for these two types of consensus nodes. In SSB, the mining difficulty coefficient for OS-Nodes is 1, and p_{role} represents the coefficient for DS-Nodes. When p_{role} approaches 1, the computational power of DS-Nodes and OS-Nodes becomes more balanced. Due to the shard tendency of DS-Nodes, a small number is also needed for malicious DS-Nodes to successfully cluster and control a specific shard, leading to lower tolerance by SSB towards them. Since OS-Nodes and honest DS-Nodes are uniformly distributed, the increasing computational power of honest DS-Nodes can tolerate more malicious OS-Nodes.

When p_{role} is 0, at this point, the tolerance of SSB to the number of malicious DS-Nodes reaches infinity, while the tolerance to the number of malicious OS-Nodes remains constant. This is because at this point, DS-Nodes no longer participate in node consensus, and their computing power cannot affect the

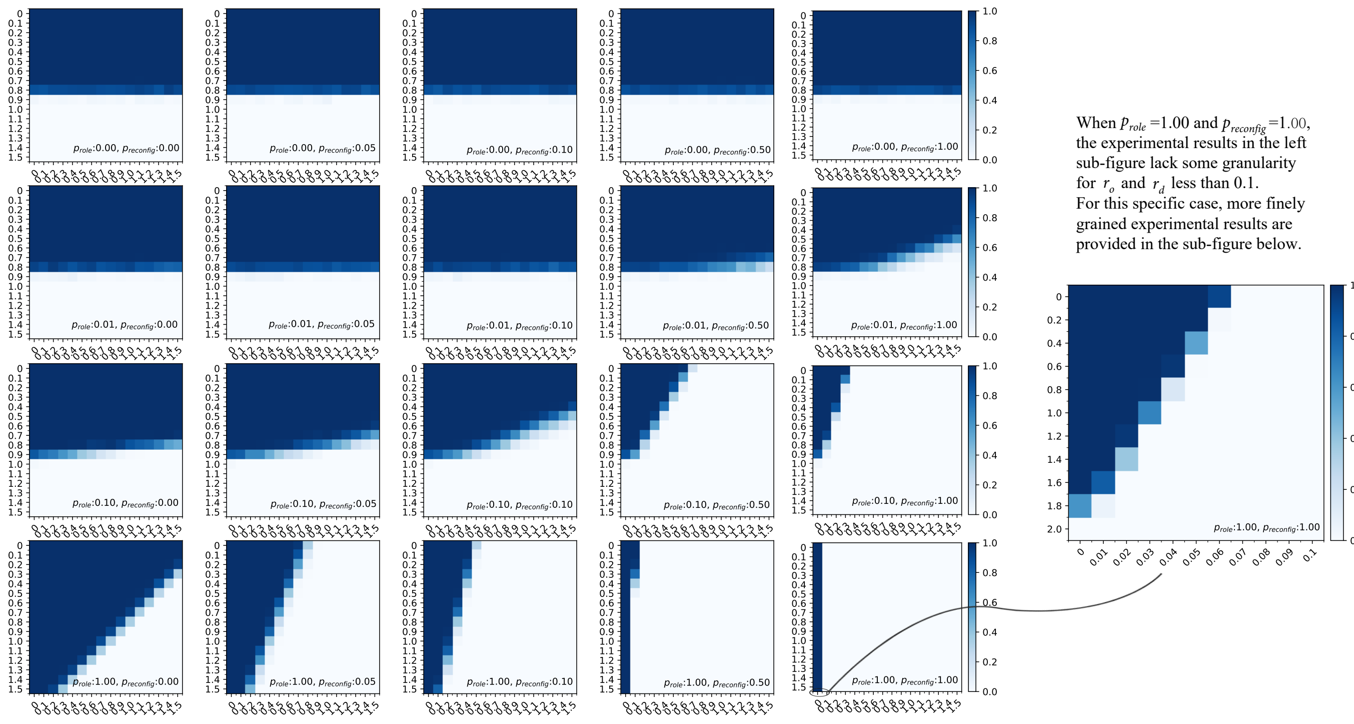
When p_{role} =1.00 and $p_{reconfig}$=1.00, the experimental results in the left sub-figure lack some granularity for r_o and r_d less than 0.1.
For this specific case, more finely grained experimental results are provided in the sub-figure below.

Fig. 5. The Probability of SSB Successfully Resisting Malicious Nodes under Different p_{role} and $p_{reconfig}$, with x-axis as r_d and y-axis as r_o

security of the blockchain. From the perspective of consensus, only OS-Nodes are left to participate in consensus in SSB, and at this point, the Po2RW consensus protocol degrades to POW. On the other hand, since DS-Nodes no longer participate in block consensus, $p_{reconfig}$ no longer has performance and security implications. From the perspective of reconfiguration, there is only one type of node in the blockchain at this point, and the reconfiguration strategy also degrades to a random reconfiguration strategy.

In particular, when a subgraph with $p_{role} = 1$ and $p_{reconfig} = 0$, it can be observed that the tolerance of SSB to the number of malicious nodes is always in direct proportion. From the perspective of computing power, when $p_{role} = 1$, the two types of semantic nodes are no longer distinguishable. From the perspective of the reconfiguration strategy, when $p_{reconfig} = 0$, DS-Nodes also adopt the same random reconfiguration strategy. At this point, SSB can resist malicious nodes to a level of approximately 50%.

For the right part of Figure 5, it can be seen that when $p_{role} = 1$ and $p_{reconfig} = 1$, SSB only exhibits resistance when r_d is close to 0. This is because when $p_{role} = 1$, DS-Nodes and OS-Nodes have the same computing power. At the same time, with $p_{reconfig} = 1$, malicious DS-Nodes can arbitrarily cluster in the same shard. Since we have divided the blockchain into 32 shards, the maximum tolerance of malicious DS-Nodes has decreased to 1/32 honest nodes.

C. Experiment on the Split-Merge Coefficient of SemantiChain

1) Experimental Setup: We set SemantiChain's $\eta_{upper} = \{0.65, 0.80, 0.95\}$ and $\eta_{lower} = \{0.2, 0.4, 0.6\}$, while keeping η_{max} fixed at 0.1. In each epoch, we adjusted the number of shards for SemantiChain based on these parameters to obtain experimental data.

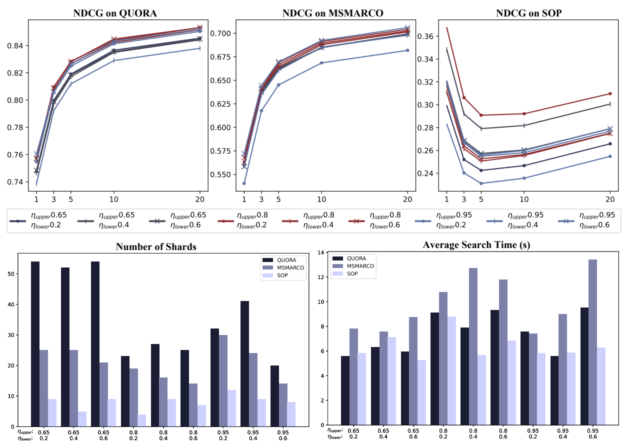

Fig. 6. Semantichain's Top-k Retrieval Results based on Different Datasets and Different Coefficients

2) Experimental Evaluation: From the lower part of Figure 6, it can be observed that, generally speaking, schemes with higher η_{lower} or η_{upper} closer to the middle tend to have higher NDCG. Particularly, the scheme with $\eta_{upper} = 0.8$ and $\eta_{lower} = 0.6$ achieves excellent performance across all three datasets. This is because a higher η_{lower} makes it easier for the throughput of each shard to reach the threshold, leading to more frequent merging between shards and thus achieving data reintegration. The re-integrated sub-corpus contains more information, making the subsequent division of the lvflist more reasonable. As for η_{upper}, a value that is too low will cause excessive fragmentation of the shards, resulting in overly dispersed data, which is not conducive to information retrieval; a value that is too high will reduce the probability of SemantiChain generating new shards. However, this stable sharding method does not increase the probability of data reintegration between shards. Therefore, we can also see that schemes with lower η_{upper} values tend to have lower NDCG, while higher η_{upper} do not necessarily yield better results.

Additionally, it can be seen that there is an inverse relationship between the number of shards and average retrieval time. Schemes with more shards require longer average retrieval times. This is because SemantiChain constructs semantic shards and performs parallel retrieval; the more shards there are, the less content each shard contains, making it easier to obtain results with less retrieval time. This relationship is precisely a manifestation of SemantiChain's distributed advantage.

D. Experiment of Retrieval of SemantiChain

1) Experimental Setup: Regarding the ANN algorithms, the number of inverted lists ($nlist$), and the quantity of nprobe ($nprobe$) will affect the performance of retrieval metrics. In order to achieve optimal retrieval performance of these algorithms, according to the empirical formula which is partly based on the GitHub community [3] and our own experience, we set $nlist$ to be the arithmetic square root of the total data volume and set $nprobe$ to be 10% of $nlist$. For algorithms using PQ, the sub-vector length $nsegments$ and the number of centroids affect quantization level and retrieval performance in a similar manner. Therefore, we fixed $nsegments$ as 8 and only vary the number of centroids to be 2^7,2^8,2^9, i.e., $nbit$ being 7, 8, 9, to illustrate their influences. Additionally, we simulated MSTDB using TF-IDF and binary tree data structures in Sklearn. For SemantiChain, based on the experimental results in VII-C, we selected the optimal parameters $\eta_{upper} = 0.8, \eta_{lower} = 0.6, \eta_{max} = 0.1$ for comparison.

2) Experimental Evaluation: The performance of the brute force algorithm demonstrates superior metrics in terms of NDCG, MAP, Recall, and Precision for all datasets, as shown in Figure 7a. However, Figure 7b reveals that the brute force algorithm also exhibits the highest average search time, with search times of 700.25s for QUORA, 732.27s for MSMARCO, and 700.25s for SOP at different top_k values. Due to the lack of quantization steps, brute force needs to load the corpus into memory during retrieval, resulting in memory usage reaching 1532.02MB, 2364.91MB and 285.41 MB for QUORA, MSMARCO and SOP, respectively.

In contrast, SemantiChain achieves significantly lower average search times of only 9.31s, 11.73s and 5.78s under the same conditions, with memory footprints of only 10.01MB, 14.11MB and 5.05 MB for QUORA, MSMARCO and SOP, respectively. From Figure 7a and 7b, it can be seen that SemantiChain only sacrifices a small amount of performance to construct the index, resulting in a significant reduction in search time and memory usage. For example, when the top-k is equal to 10, compared to the brute force algorithm, SemantiChain's average NDCG, MAP, Recall and Precision decreased by about 5.09%, 4.92%, 4.33% and 0.76 respectively, but its average search time and space occupancy is only 1.38% and 0.65% of the cost of the brute force algorithm.

[3]https://github.com/facebookresearch/faiss

The brute force algorithm represents a naive approach, and its performance metrics for NDCG, MAP, Recall, and Precision serve as the theoretical upper limits within the current semantic space mapping. In contrast, SemantiChain significantly reduces time and space usage at the cost of a certain level of precision loss, achieved through index construction, distributed quantization, and distributed retrieval. Notably, SemantiChain can control the degree of distortion by adjusting the number of shards.

Additionally, while the HNSW algorithm demonstrates no significant difference in NDCG, MAP, Recall, and Precision performance compared to SemantiChain, it achieves an average search time of only 1.72s for three datasets. However, this comes at the expense of higher memory usage, with HNSW consuming 1667.74MB, 2574.4MB and 310.7MB of memory. Although the average search time of SemantiChain is higher than that of HNSW, the average space occupancy is only 0.61% of it.

This is because HNSW also does not have a quantization step, and the entire corpus needs to be loaded into memory for retrieval. At the same time, since HNSW also needs to build a retrieval index with a graph structure to significantly reduce the time, its memory usage needs to be even greater than the brute force algorithm. Since the construction of HNSW's graph index structure will affect the retrieval performance, it will also lose retrieval accuracy. SemantiChain, with the advantage of distribution, can significantly reduce space usage while ensuring similar retrieval performance.

For other common ANN algorithms such as PQ, IVFPQ, and Faiss, they all apply product quantization to the corpus, and varying degrees of quantization have different impacts on retrieval performance. It is observed that, at the same $top - k$ value, an increase in the quantization index $nbit$ leads to an increase in NDCG, MAP, Recall, and Precision. Similarly, in Figure 7a, memory usage and index construction time also increase with $nbit$. However, the average search time is at its lowest only when $nbit$ is 8. Specifically, the average search time of the 8-bit quantized IVFPQ algorithm is the shortest at 12.21s, but SemantiChain is still 9.71s faster than it. Similarly, the memory usage of the 7-bit quantized PQ algorithm is the least at 42.27MB, and SemantiChain reduces it by 76.09% compared to it. Overall, SemantiChain demonstrates no significant differences in NDCG, MAP, Recall, and Precision performance compared to these algorithms. However, leveraging the advantages of distribution, SemantiChain exhibits substantial improvements in average search time, index construction time, and memory usage.

3) Statistical Analysis: In this section, we analyzed the performance comparison of different retrieval schemes and SemantiChain on the datasets QUORA, MSMARCO, SOP. We selected some metrics of each scheme at $Top_k = 10$, including NDCG, Average Search Time, and Memory for analysis. We used Two-way analysis of variance (ANOVA) to test whether there is a significant difference ($p < 0.05$) in the data means between different schemes and SemantiChain

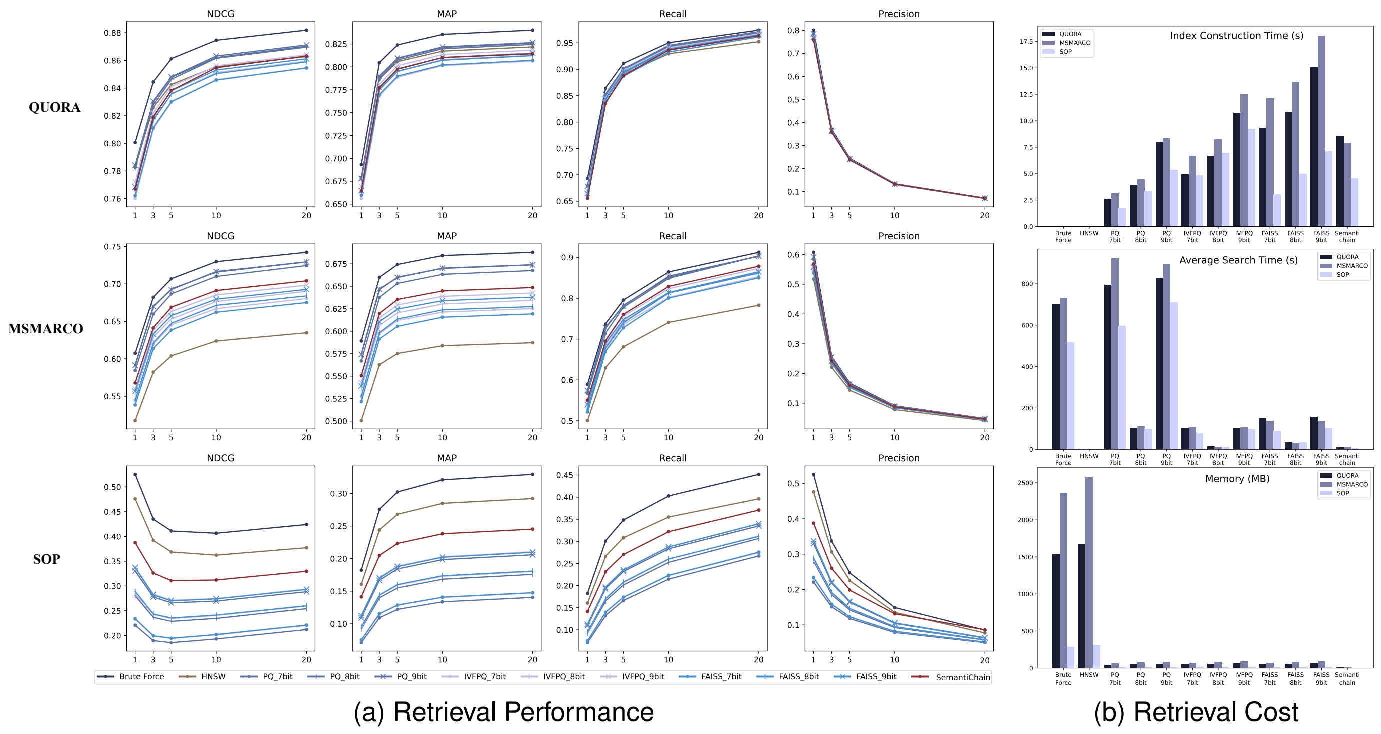

Fig. 7. Top-k Retrieval Results of Different Various Search Schemes based on Different Datasets

under the same metrics of the same dataset, and calculated the corrected significant difference ($p_a dj < 0.05$) and related details through Tukey HSD, and put them into Table IV.

It can be seen that when comparing the NDCG, except for Brute Force, SemantiChain has a significant difference with most methods on the MSMARCO and SOP datasets, and is slightly stronger than these methods. On the QUORA dataset, SemantiChain has no significant difference with most schemes. For the average search time, on these three datasets, except for HNSW and 8-bit quantized IVFPQ, SemantiChain is significantly less than all other schemes. Especially for PQ 7bit, PQ 9bit and Brute Force, the difference is large, and SemantiChain only consumes 1.1%-1.4% of their time. Compared with IVFPQ 8bit and HNSW, in most cases, there is no significant difference in the average search time of SemantiChain. In addition, due to the advantages of distribution, SemantiChain's memory occupancy is significantly lower than all algorithms, especially compared with HNSW and Brute Force, SemantiChain's memory occupancy on the three datasets is reduced to an average of 0.96% and 0.97%.

The results of Statistical Analysis are consistent with the experimental results. Most of the time, compared with other schemes, SemantiChain can achieve optimization in time and space without reducing the retrieval performance through this distributed ANN algorithm.

E. Experiment of Blockchian Performance

1) Experimental Setup: We used TF-IDF and binary tree data structure in sklearn to simulate MSTDB. Since MSTDB does not support storing image data, this experiment is not applicable to the SOP dataset. To emphasize the performance differences between various blockchain methods, we set the transaction broadcast rate for each blockchain to 1000 tx/s, initial shard number to 50, block interval to 0.1s, and simulate transaction data for throughput and latency testing.

2) Experimental Evaluation: From Table V, it can be seen that compared to MSTDB, SemantiChain has similar index building and search times on datasets around 1M in size, but its memory usage is significantly lower than MSTDB. By analysis, MSTDB constructs an index by traversing each data's keywords to build a multi-branch tree. Since the number of keywords for each data is a constant, the time and space complexity of its index building is simplified to $O(N)$, and the search time complexity is about $O(logN)$, where N represents the number of data points in the dataset.

SemantiChain computes the arithmetic square root of the data to form $\sqrt{N}$ inverted file lists, with each shard obtaining a constant number of inverted file lists for distributed product quantization. Therefore, the index building time complexity for each shard is about $O(\sqrt{N} \times N_l \times logN_k)$, where N_l denotes the vector dimension and N_k represents the number of quantization clusters. Since these two parameters are also constants, the index building time and space complexity are simplified to $O(\sqrt{N})$, and the search time complexity is $O(\sqrt{N})$. Thus, leveraging the advantages of its distribution, SemantiChain achieves greater efficiency in terms of index-building time and space complexity.

Additionally, Table V also demonstrates the throughput and latency of various state-of-the-art blockchain methods. Due to MSTDB lacking on-chain scalability techniques, its throughput and transaction latency metrics are significantly weaker compared to SemantiChain and Monoxide. Because

TABLE IV
PERFORMANCE ANALYSIS OF DIFFERENT SEARCH SCHEMES COMPARED TO SEMIANTICHAIN

		QUORA				MSMARCO				SOP			
Metric	**Scheme**	**MeanDiff**	**P-adj**	**CI-Low**	**CI-Upr**	**MeanDiff**	**P-adj**	**CI-Low**	**CI-Upr**	**MeanDiff**	**P-adj**	**CI-Low**	**CI-Upr**
NDCG	Brute Force	-0.0205	0.0000	-0.0270	-0.0141	-0.0416	0.0000	-0.0466	-0.0367	-0.1054	0.0000	-0.1072	-0.1036
	FAISS 7bit	0.0100	0.0002	0.0035	0.0164	0.0277	0.0000	0.0227	0.0326	0.0996	0.0000	0.0978	0.1014
	FAISS 8bit	\	\	\	\	0.0191	0.0000	0.0141	0.0241	0.0608	0.0000	0.0590	0.0626
	FAISS 9bit	\	\	\	\	0.0094	0.0000	0.0044	0.0144	0.0284	0.0000	0.0266	0.0302
	HNSW	\	\	\	\	0.0638	0.0000	0.0589	0.0688	-0.0591	0.0000	-0.0609	-0.0573
	IVFPQ 7bit	0.0098	0.0002	0.0034	0.0163	0.0213	0.0000	0.0163	0.0263	0.1001	0.0000	0.0983	0.1019
	IVFPQ 8bit	\	\	\	\	0.0124	0.0000	0.0074	0.0174	0.0614	0.0000	0.0596	0.0632
	IVFPQ 9bit	\	\	\	\	0.0057	0.0130	0.0007	0.0107	0.0271	0.0000	0.0253	0.0289
	PQ 7bit	\	\	\	\	-0.0216	0.0000	-0.0266	-0.0166	0.1082	0.0000	0.1064	0.1100
	PQ 8bit	\	\	\	\	-0.0258	0.0000	-0.0308	-0.0208	0.0677	0.0000	0.0660	0.0695
	PQ 9bit	\	\	\	\	-0.0283	0.0000	-0.0333	-0.0233	0.0325	0.0000	0.0307	0.0343
Average Search Time	Brute Force	-708.1812	0.0000	-721.2732	-695.0892	-706.6395	0.0000	-716.4305	-696.8486	-495.3442	0.0000	-508.4265	-482.2618
	FAISS 7bit	-137.9753	0.0000	-151.0673	-124.8833	-122.5792	0.0000	-132.3701	-112.7882	-84.2514	0.0000	-97.3337	-71.1690
	FAISS 8bit	-25.3399	0.0000	-38.4319	-12.2478	-16.3927	0.0000	-26.1836	-6.6018	-28.9070	0.0000	-41.9894	-15.8247
	FAISS 9bit	-145.7374	0.0000	-158.8294	-132.6453	-127.3465	0.0000	-137.1375	-117.5556	-92.0900	0.0000	-105.1723	-79.0076
	HNSW	\	\	\	\	10.3468	0.0299	0.5559	20.1377	\	\	\	\
	IVFPQ 7bit	-88.2890	0.0000	-101.3810	-75.1969	-96.9887	0.0000	-106.7796	-87.1977	-72.5812	0.0000	-85.6635	-59.4988
	IVFPQ 8bit	\	\	\	\	\	\	\	\	\	\	\	\
	IVFPQ 9bit	-94.7456	0.0000	-107.8376	-81.6535	-94.3519	0.0000	-104.1428	-84.5610	-94.6620	0.0000	-107.7443	-81.5796
	PQ 7bit	-804.4442	0.0000	-817.5362	-791.3522	-913.6001	0.0000	-923.3910	-903.8092	-565.3525	0.0000	-578.4348	-552.2701
	PQ 8bit	-92.3162	0.0000	-105.4082	-79.2242	-101.0206	0.0000	-110.8115	-91.2297	-96.5932	0.0000	-109.6755	-83.5108
	PQ 9bit	-806.3864	0.0000	-819.4785	-793.2944	-892.4798	0.0000	-902.2707	-882.6888	-697.7856	0.0000	-710.8680	-684.7032
Memory	Brute Force	-1526.0844	0.0000	-1544.1042	-1508.0646	-2440.2872	0.0000	-2480.3740	-2400.2003	-284.5681	0.0000	-288.4244	-280.7118
	FAISS 7bit	-37.4810	0.0000	-55.5008	-19.4612	-64.2441	0.0001	-104.3310	-24.1573	-4.9914	0.0027	-8.8477	-1.1351
	FAISS 8bit	-45.0620	0.0000	-63.0818	-27.0422	-70.9919	0.0000	-111.0787	-30.9050	-6.0010	0.0001	-9.8573	-2.1447
	FAISS 9bit	-51.9433	0.0000	-69.9631	-33.9235	-77.9724	0.0000	-118.0593	-37.8856	-8.5752	0.0000	-12.4315	-4.7189
	HNSW	-1620.8367	0.0000	-1638.8565	-1602.8169	-2530.3419	0.0000	-2570.4288	-2490.2551	-307.1155	0.0000	-310.9718	-303.2592
	IVFPQ 7bit	-37.2500	0.0000	-55.2698	-19.2302	-59.1540	0.0004	-99.2409	-19.0672	-4.6038	0.0079	-8.4601	-0.7475
	IVFPQ 8bit	-46.6995	0.0000	-64.7193	-28.6797	-68.3539	0.0000	-108.4407	-28.2670	-6.1728	0.0001	-10.0291	-2.3165
	IVFPQ 9bit	-49.8982	0.0000	-67.9180	-31.8784	-81.5525	0.0000	-121.6393	-41.4657	-8.0830	0.0000	-11.9393	-4.2267
	PQ 7bit	-33.3663	0.0000	-51.3861	-15.3465	-50.9589	0.0035	-91.0458	-10.8721	\	\	\	\
	PQ 8bit	-37.2215	0.0000	-55.2413	-19.2017	-59.0342	0.0004	-99.1210	-18.9473	-4.5680	0.0086	-8.4243	-0.7117
	PQ 9bit	-45.3016	0.0000	-63.3214	-27.2818	-69.9897	0.0000	-110.0765	-29.9028	-6.5164	0.0000	-10.3727	-2.6601

TABLE V
BASIC PERFORMANCE OF DIFFERENT BLOCKCHAINS

Metric		**SemantiChain**	**MSTDB**	**Monoxide**
Index Construction Time (s)	QUORA	7.08	7.35	/
	MSMARCO	7.33	9.26	/
Average Search Time (s)	QUORA	9.39	9.04	/
	MSMARCO	11.75	7.51	/
Memory(MB)	QUORA	9.31	196.64	/
	MSMARCO	9.72	260.11	/
Throughput(tx/s)	/	813.20	9.79	489.92
Latency(ms)	/	22.97	2287.73	104.11

TABLE VI
NDCG EXPERIMENTAL RESULTS OF BLOCKCHAINS

Top_k		**MSTDB**	**SemantiChain**
1	QUORA	0.0440	0.7670
	MSMARCO	0.0312	0.5679
3	QUORA	0.0669	0.8189
	MSMARCO	0.0509	0.6414
5	QUORA	0.1012	0.8382
	MSMARCO	0.0784	0.6688
10	QUORA	0.1830	0.8548
	MSMARCO	0.1445	0.6912
20	QUORA	0.3080	0.8632
	MSMARCO	0.2453	0.7041

the number of shards in Monoxide is fixed, its transaction throughput is also far below the transaction broadcast rate of 1000tx/s, indicating there is still room for scalability improvement. Moreover, the significant number of transactions consistently waiting for verification in the transaction pool contributes to increased latency in both comparison schemes.

Table VI shows the retrieval performance of different blockchains. Since Monoxide is not designed for information retrieval, and MSTDB can only store textual data, we chose QUARO and MSMARCO datasets and MSTDB for comparison. It can be seen that Semantichain's NDCG is 45.88% higher than MSTDB. This is because MSTDB uses keywords as the semantic representation of data, without considering contextual influences, thus its semantics are not precise enough. Additionally, in QUARO and MSMARCO, text lengths are limited, and semantic features based on keywords tend to produce many similar results in large-scale data, leading to decreased retrieval performance.

VIII. CONCLUSION

This article primarily addresses the functional deficiencies of blockchain in big data storage and retrieval. To this end, we introduced a novel blockchain sharding technique called *Semantic Sharding* to tackle the scalability issues of semantic blockchain as a storage architecture. Building upon this, we developed the *Semantic Sharding Blockchain* (SSB), which surpasses the limitations of blockchain storage file types by understanding and utilizing data semantics. To ensure the security and scalability of SSB, we proposed the *Po2RW* consensus protocol and a reconfiguration strategy to dynamically balance between them, enabling SSB to adapt to different application scenarios. Additionally, we proposed a distributed retrieval architecture called *Semantic Information Retrieval* (SIR) to achieve retrieval of big data within the blockchain, leveraging distributed advantages to reduce the time and space costs of ANN retrieval algorithms. Finally, by integrating SSB with SIR, we presented a trusted retrieval blockchain based on semantic sharding called *SemantiChain*. Its security and effectiveness are validated through secure analysis and experiments, proving superior to state-of-the-art blockchain database architectures.

In our future work, we will focus on optimizing SemantiChain from the perspectives of SSB and SIR. For SSB, the decision-making process for sharding strategies can be optimized based on the specific environment. For SIR, although distributed retrieval already offers advantages, we can implement a pipeline retrieval process to expedite retrieval speed, especially when handling multiple simultaneous queries.

References

[1] Y. Cao, C. Yi, G. Wan, H. Hu, Q. Li, and S. Wang, "An analysis on the role of blockchain-based platforms in agricultural supply chains," *Transportation Research Part E: Logistics and Transportation Review*, vol. 163, p. 102731, Jul. 2022.

[2] H. H. Khan, M. N. Malik, Z. Konečná, A. G. Chofreh, F. A. Goni, and J. J. Klemeš, "Blockchain technology for agricultural supply chains during the covid-19 pandemic: Benefits and cleaner solutions," *Journal of Cleaner Production*, vol. 347, p. 131268, 2022.

[3] Y. Li, C. Tan, W. Ip, and C. Wu, "Dynamic blockchain adoption for freshness-keeping in the fresh agricultural product supply chain," *Expert Systems with Applications*, vol. 217, p. 119494, 2023.

[4] J. Dong, C. Song, S. Liu, H. Yin, H. Zheng, and Y. Li, "Decentralized peer-to-peer energy trading strategy in energy blockchain environment: A game-theoretic approach," *Applied Energy*, vol. 325, p. 119852, 2022.

[5] K. Almutairi, S. J. Hosseini Dehshiri, S. S. Hosseini Dehshiri, A. X. Hoa, J. Arockia Dhanraj, A. Mostafaeipour, A. Issakhov, and K. Techato, "Blockchain technology application challenges in renewable energy supply chain management," *Environmental Science and Pollution Research*, vol. 30, no. 28, pp. 72 041–72 058, 2023.

[6] A. Umar, D. Kumar, and T. Ghose, "Blockchain-based decentralized energy intra-trading with battery storage flexibility in a community microgrid system," *Applied Energy*, vol. 322, p. 119544, 2022.

[7] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain: Research and Applications*, vol. 3, no. 2, p. 100067, Jun. 2022.

[8] S. A. Khan, M. S. Mubarik, S. Kusi-Sarpong, H. Gupta, S. I. Zaman, and M. Mubarik, "Blockchain technologies as enablers of supply chain mapping for sustainable supply chains," *Business Strategy and the Environment*, vol. 31, no. 8, pp. 3742–3756, Dec. 2022.

[9] Q. Zhu, C. Bai, and J. Sarkis, "Blockchain technology and supply chains: The paradox of the atheoretical research discourse," *Transportation Research Part E: Logistics and Transportation Review*, vol. 164, p. 102824, 2022.

[10] M. N. Halgamuge, G. K. Munasinghe, and M. Zukerman, "Time estimation for a new block generation in blockchain-enabled internet of things," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 535–557, 2024.

[11] Y. Ren, D. Huang, W. Wang, and X. Yu, "Bsmd: A blockchain-based secure storage mechanism for big spatio-temporal data," *Future Generation Computer Systems*, vol. 138, pp. 328–338, 2023.

[12] Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, "SEBDB: Semantics Empowered BlockChain DataBase," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. Macao, Macao: IEEE, Apr. 2019, pp. 1820–1831.

[13] E. Zhou, Z. Hong, Y. Xiao, D. Zhao, Qingqi Pei, S. Guo, and R. Akerkar, "MSTDB: A Hybrid Storage-Empowered Scalable Semantic Blockchain Database," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–17, 2022.

[14] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[15] F. Gong, L. Kong, Y. Lu, J. Qian, and X. Min, "An Overview of Blockchain Scalability for Storage," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, May 2023, pp. 516–521.

[16] F. Hashim, K. Shuaib, and N. Zaki, "Sharding for Scalable Blockchain Networks," *SN Computer Science*, vol. 4, no. 1, p. 2, Oct. 2022.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

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

[19] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, pp. 53–72, 2016.

[20] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *arXiv preprint arXiv:1407.3561*, Jul. 2014.

[21] G. Danezis and S. Meiklejohn, "Centrally Banked Cryptocurrencies," *arXiv preprint arXiv:1505.06895*, Dec. 2015.

[22] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol For Open Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna Austria: ACM, Oct. 2016, pp. 17–30.

[23] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 583–598.

[24] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, 2019, pp. 95–112.

[25] H. Huang, X. Peng, J. Zhan, S. Zhang, Y. Lin, Z. Zheng, and S. Guo, "BrokerChain: A Cross-Shard Blockchain Protocol for Account/Balance-based State Sharding," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. London, United Kingdom: IEEE, May 2022, pp. 1968–1977.

[26] Z. Zhen, X. Wang, H. Lin, S. Garg, P. Kumar, and M. S. Hossain, "A dynamic state sharding blockchain architecture for scalable and secure crowdsourcing systems," *Journal of Network and Computer Applications*, p. 103785, 2023.

[27] S. Zhang and J.-H. Lee, "Double-spending with a sybil attack in the bitcoin decentralized network," *IEEE transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5715–5722, 2019.

[28] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[29] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[30] A. Aghaebrahimian, "Quora Question Answer Dataset," in *Text, Speech, and Dialogue*, K. Ekštein and V. Matoušek, Eds. Cham: Springer International Publishing, 2017, vol. 10415, pp. 66–73.

[31] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang, "MS MARCO: A Human Generated MAchine Reading Comprehension Dataset," *arXiv preprint arXiv:1611.09268*, Oct. 2016.

[32] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[33] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019.

[34] N. Reimers, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2020.

[35] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.