MONASH University

Practical and Secure Keyword Search Services on Blockchain

Jun Zhao

Doctor of Philosophy

A Thesis Submitted for the Degree of Doctor of Philosophy at

Monash University in 2024

School of Information Technology

Copyright notice

Abstract

Blockchain-based cloud storage has seen significant growth in both academic and industrial sectors in recent years. A key feature missing in current blockchain-based cloud storage systems is the ability to perform keyword searches, as the encryption of stored files prevents the use of traditional search methods. To address this challenge and enable efficient keyword search without compromising privacy, blockchain-based searchable encryption schemes have become an area of increasing research interest. These schemes aim to enable secure, searchable access to encrypted data within blockchain environments.

This thesis investigates the challenges and solutions in developing blockchain-based Symmetric Searchable Encryption (SSE) systems, focusing on key aspects such as security, scalability, and incentives. First, a novel off-chain verification method for SSE systems is introduced, addressing both dishonest server nodes and users, ensuring security and on-chain scalability. Second, AegisDB, a non-verification-based secure blockchain databases, is proposed, introducing a novel load-balancing algorithm that improves scalability while preserving the security guarantees of blockchain systems. Third, SEARCHAIN, a proof-of-useful-work blockchain system for SSE queries, is presented, incorporating a novel committee selection algorithm to incentivise honest verification by auditors, ensuring service fairness. Through both theoretical analysis and empirical evaluation, this thesis demonstrates how these approaches enhance the security and practicality of blockchain-based SSE systems.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: _____

Print Name: Jun Zhao

Date: 5/11/2024

Publications during enrolment

Publications included in this thesis:

- Zhao, J., Yu, J., Yuan, X., Liu, J. K., & Zuo, C. (2023, November). Rethinking Practical Blockchain-Based Symmetric Searchable Encryption Services. In 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 307-317). IEEE.
- Zhao, J., Yu, J., Yuan, X., & Liu, J. K. (2024, July). AegisDB: Scalable Blockchain Database with Secure Decentralised Load Balancing. In Australasian Conference on Information Security and Privacy (pp. 105-119). Singapore: Springer Nature Singapore.
- Zhao, J., Yu, J., Yuan, X., & Liu, J. K. (2024) SEARCHAIN: Searchable Encryption As Rewarded-useful-work on Blockchain. (Submitted, in review)

Other publications not included in this thesis:

- Li, Z. Z., Zhao, J., Cui, H., Zheng, X., Ma, X, Liu, D. Y., & Au, M. H. (2024). In Proceedings of the 6th ACM International Symposium on Blockchain and Secure Critical Infrastructure. (In press)
- Zhang, X., Li, Z. Z., Cui, H., Zhao, J., & Chen, X. (2024). BDEC: Enhancing Learning Credibility via Post-Quantum Digital Credentials. In Provable and Practical Security: 18th International Conference, ProvSec 2024. (In press)

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and guided me throughout the course of this research and thesis.

First and foremost, I would like to thank my supervisors, Prof. Joseph Liu, A/Prof. Jiangshan Yu, and A/Prof. Xingliang Yuan, especially my main supervisor Prof. Joseph Liu for his invaluable guidance, insightful feedback, and unwavering support throughout my entire PhD journey. Their expertise, patience, and encouragement have been essential in shaping this thesis and my growth as a researcher. I would also like to extend my gratitude to Dr. Cong Zuo for his insights, and collaboration in various stages of this research.

I am also deeply grateful to the panel members, A/Prof. Ron Steinfeld, Dr. Shujie Cui, and Dr. Hui Cui, for their time, constructive comments, and suggestions, which have helped to refine my work and broaden my understanding of the subject matter. I would also like to express my sincere gratitude to A/Prof. Ron Steinfeld for his exceptional supervision and mentorship during my teaching fellow position. He was always willing to offer guidance, help and address any questions I had. Additionally, I am deeply thankful to my mentor and colleague, Dr. Shujie Cui for her invaluable support and advice throughout my teaching role.

On a personal note, I am deeply thankful to my family for their endless love, understanding, and encouragement. Their belief in me has been a constant source of strength and motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The rapid adoption of blockchain technology across various sectors has transformed how data is stored, validated, and shared. Blockchain introduces a decentralised, immutable, and transparent infrastructure that provides innovative alternatives for centralised cloud storage services [65, 70, 84, 99, 116]. However, as blockchain systems evolve to handle more sensitive and private data, the need for confidentiality and efficient data retrieval mechanisms has become increasingly critical [34,79].

One of the primary challenges in blockchain-based applications is the secure storage and querying of encrypted data. While blockchain ensures data integrity and transparency, it does not inherently address the privacy concerns that arise when storing sensitive information on a public ledger [34,92]. By design, all data stored on a blockchain is visible to all participants, which presents a significant problem when dealing with private or confidential data, such as personal health records, financial information, or proprietary business documents.

To address these privacy concerns, data on the blockchain is typically encrypted. However, encrypting data introduces new challenges, specifically, the need to perform secure searches on this encrypted data without decrypting it. Symmetric Searchable Encryption [69, 82, 114] is a cryptographic technique designed to enable searches over encrypted data, thereby addressing the challenge of maintaining privacy while allowing for efficient data retrieval. In SSE, an encrypted index is created and stored alongside the encrypted data to allow efficient and secure searching by the owner of the secret key. However, applying searchable encryption

to blockchain-based systems introduces additional complexities. The decentralised and distributed nature of blockchains makes it difficult to implement traditional, centralised models of searchable encryption, primarily due to three key challenges: the trust model, on-chain feasibility and incentives.

A primary challenge in applying searchable encryption to blockchain systems lies in the trust model. Traditional searchable encryption schemes are designed for a centralised environment, where the central server is assumed to be semi-honest and the clients are assumed to be honest [69, 82, 114]. In this context, "semi-honest" means that the server may attempt to breach the privacy of clients but will otherwise follow the querying protocol honestly. However, in a decentralised peer-to-peer blockchain network, this assumption is not practical. In such a scenario, both parties may act dishonestly. Blockchain nodes are typically assumed to be rational, meaning they may not follow the protocol if doing so is not in their best interest or profitable. On the other hand, the client can also be malicious, seeking to repudiate service fees. This difference in the trust model introduces significant challenges when applying searchable encryption to blockchain systems, as it requires designing protocols that can handle dishonest or self-interested behaviour from both users and nodes.

The inherent issues of blockchain systems [52] present another significant challenge when applying searchable encryption to blockchain networks. Blockchains, particularly public ones like Ethereum, are limited by factors such as transaction throughput and network latency. As the blockchain grows, these limitations make it increasingly difficult to maintain the efficiency of a blockchain-based searchable encryption system. First, storing encrypted indexes on-chain for secure searching becomes quickly infeasible due to high transaction fees for users and storage costs for replicated nodes. Second, executing queries on-chain via smart contracts is problematic because of the high gas costs. Third, query response times will be significantly slower compared to traditional searchable encryption systems with a centralised structure. When combined with the challenges posed by the trust model, these scalability issues create a more complex problem for applying searchable encryption in blockchain systems. The system must be designed to handle both malicious parties and high costs, often requiring that storage and computation be offloaded off-chain to ensure both efficiency and feasibility.

Another key problem in the context of searchable encryption on blockchain is that

the incentives for auditors, who are responsible for verifying the integrity and correctness of the search results, are often unclear. Auditors play a crucial role in ensuring that search results are accurate and that the system operates in a secure and trustworthy manner. In the existing studies, they are usually assumed to be voluntary and honest [4, 21, 92]. However, in decentralised blockchain environments, where auditors may be participants in the network, it becomes difficult to align their interests with the integrity of the system. Without a clear incentive structure, auditors may lack motivation to honestly verify search results. Furthermore, the cost of performing verifications on-chain can be high [92], which may encourage auditors to do the opposite, assuming they are rational actors. This lack of clear incentives can undermine the trustworthiness of the entire system, highlighting the need for incentive mechanisms that encourage auditors to act honestly in the blockchain context.

In a nutshell, the potential impact of building a practical and secure keyword search service for blockchain storage systems is significant but challenging. This research aims to address these challenges and contribute to the development of novel protocols that enable secure, efficient, and scalable searchable encryption on blockchain networks.

1.2 Research Questions

The main research question for this research is:

> **How to build a practical and secure blockchain-based searchable encryption system?**

This research aims to address all of the aforementioned challenges and ultimately develop a blockchain-based searchable encryption system that is secure against both dishonest users and dishonest blockchain nodes, while maintaining efficient on-chain storage and computational costs. Specifically, three key aspects of these challenges are examined. The investigation explores the interactions between these challenges and potential solutions. In particular, this thesis seeks to contribute to the body of knowledge by addressing the following research sub-questions:

- *RQ1: How can security and feasibility be achieved simultaneously under the blockchain trust model?*

  Storing and processing searchable encryption queries on-chain can enhance security but at the cost of significantly increased query response times and reduced overall system throughput compared to traditional searchable encryption systems. The performance can deteriorate even further when dishonest users and malicious nodes are considered, where additional verifications and dispute resolutions are needed. To address this issue, a new off-chain verification mechanism capable of handling both malicious parties needs to be introduced, which is challenging and has been overlooked in previous studies. This research question represents the first step toward achieving the primary goal of this research.

- *RQ2: How to achieve security and feasibility with a non-verification solution?*

  The challenge posed by RQ1 primarily stems from the complexity of off-chain verifications, which must handle both malicious parties while maintaining efficiency. RQ2 investigates a novel alternative solution to this problem by exploring different security models and mechanisms that do not rely on verifications.

- *RQ3: How to build a complete blockchain searchable encryption system with proof of useful work consensus?*

  The remaining challenge not addressed by RQ1 and RQ2 is the incentive structure of the system. RQ3 highlights this crucial aspect of the research, focusing on how to align the interests of all participants, including users, server nodes, and auditors, to ensure the system's security. This research question investigates how to design a consensus protocol for the blockchain that encourages honest behaviour from all parties. By answering this question, a blockchain-based searchable encryption system that meets all the design goals can be developed, thereby achieving the main goal of the research.

1.3 Contributions

This thesis makes three major contributions. First, it presents a secure and practical blockchain-based searchable encryption system that protects against both malicious users and malicious server nodes, using an off-chain verification method. Second, it introduces an alternative solution to the problem, AegisDB, which is non-verification-based and achieves similar security guarantees and performance. Finally, the thesis concludes the research with SEARCHAIN, a blockchain-based searchable encryption system that employs a proof-of-useful-work consensus, where serving and verifying searchable encryption queries entitle nodes to be selected as members of the mining committee.

1.3.1 Rethinking Practical Blockchain-Based Symmetric Searchable Encryption Services

The first contribution presents a construction of the blockchain-based Symmetric Searchable Encryption (SSE) system that employs a novel off-chain verification method to address both dishonest server nodes and dishonest users. This addresses RQ1. The chapter begins by defining the key requirements for a practical and secure blockchain-based searchable encryption system, posing three essential design criteria. The first two requirements are fast query response and low on-chain storage costs, which are crucial for the usability and feasibility of a blockchain-based SSE system. The third requirement is soundness, ensuring that dishonest behaviour from both users and service providers can be detected and prevented. Following this, a novel construction of the blockchain-based SSE system is proposed. Both theoretical analysis and empirical evaluation are conducted to demonstrate that the system meets all the desired properties.

1.3.2 AegisDB: Scalable Blockchain Database with Secure Decentralised Load Balancing

The second contribution focuses on a non-verification solution that addresses RQ2. Since no verification is required, this solution can be applied to general blockchain databases, regardless of whether the queries are encrypted. Blockchain databases

face performance limitations due to scalability issues inherent in the underlying blockchain. While existing solutions, such as sharding, can mitigate these scalability issues, they often compromise the security benefits provided by blockchain technology. To address this challenge, a novel non-verification method is explored to improve scalability without undermining the security assumptions of the blockchain. In this context, the classical concept of load balancing is reintroduced into blockchain database systems. A VRF(Verifiable Random Function)-based load-balancing algorithm is proposed, and AegisDB, a blockchain database system built on this algorithm, is constructed. AegisDB ensures scalability while preserving security guarantees similar to those of unsharded blockchains.

1.3.3 SEARCHAIN: Searchable Encryption As Rewarded-useful-work on Blockchain

The third contribution of this thesis introduces SEARCHAIN, a novel blockchain-based SSE system designed to address RQ3. SSE auditors play a crucial role in SSE systems, especially in blockchain environments, where service fairness must be ensured between dishonest server nodes and dishonest users. With the assistance of honest verifiers, fairness can be maintained. However, the incentives for verifiers to perform this work and act honestly have not been investigated in existing research. To address this problem, a novel committee selection algorithm is proposed to select verifier committee members, ensuring that their voting power is proportional to the number of active SSE services they manage. Verifiers are required to perform result verification for SSE services in order to participate in mining, and committee members validate each other's work to confirm membership. In this way, the committee selection algorithm provides both security and incentives for the system. The consensus mechanism of SEARCHAIN can be viewed as a proof-of-useful-work, where the useful work consists of serving and verifying SSE queries. To the best of our knowledge, SEARCHAIN is the first blockchain-based SSE system to offer clear incentives for auditors.

1.4 Thesis Structure

This thesis is organised as follows. Chapter 2 reviews the related literature relevant to this research. Chapter 3 presents a secure and practical blockchain SSE system based on the off-chain verification method. Chapter 4 introduces AegisDB, a blockchain database that achieves similar security and performance goals without relying on verifications. Chapter 5 explores the incentives of the parties involved in the blockchain SSE scenario by presenting SEARCHAIN, a blockchain SSE system with a proof-of-useful-work consensus. Chapter 6 outlines possible future directions related to this thesis. Finally, Chapter 7 concludes the thesis.

Chapter 2

Related Work

This chapter summarises the related work for this thesis. Note that in this chapter we discuss the literature that outlines the general background of the research area. The related work for each specific contribution is discussed in more detail in the corresponding chapter.

2.1 Blockchain Storage System

Blockchain storage systems [49, 65, 70, 84, 99, 116] leverage blockchain and peer-to-peer networks to distribute data storage across multiple nodes, removing the need for centralised intermediaries while enhancing data security and availability. These systems often utilise cryptographic techniques to ensure confidentiality and integrity, and employ replication strategies to improve data availability, making them resilient to single points of failure and censorship. Prominent examples of decentralised storage solutions include Storj [116], Filecoin [65], and Sia [70], each of which uses blockchain technology to incentivise participants for contributing storage resources. To ensure that data remains intact and stored on the nodes for the provisioned duration, nodes are required to submit cryptographic proofs [36, 39, 94] to the blockchain. While decentralised storage systems offer significant advantages in terms of security and availability, they face challenges related to scalability and performance limitations. Additionally, encrypted files can only be retrieved by their identifiers, rather than by searchable keywords, which impairs the usability of the system.

2.2 Blockchain Database

Blockchain database systems [3,15,29,30,42,57,59,66,77,107] have gained increasing attention due to their potential to eliminate reliance on centralised servers, offering enhanced security and fault tolerance. While the pioneering works in this area explore the feasibility of implementing practical blockchain-based systems comparable to traditional centralised databases in terms of performance, they each focus on different aspects and challenges. One key aspect is ensuring result integrity, a topic closely related to traditional verifiable databases. For example, [66] ensures query result integrity through on-chain verifications, while [15] employs an Authenticated Data Structure (ADS) stored on the blockchain to provide verifiability. Another important topic is transaction ordering [30, 42, 77, 107], a concern derived from traditional distributed database systems where executing dependent transactions in the wrong order can lead to data anomalies. While these studies focus on the database side, others concentrate more on the blockchain consensus side, exploring solutions such as sharding mechanisms to improve throughput of the blockchain database [57, 59].

2.3 Searchable Encryption

Searchable encryption is a cryptographic technique that enables searching over encrypted data without requiring decryption, ensuring both confidentiality and usability of sensitive information. This technique was first introduced with Symmetric Searchable Encryption (SSE) in the seminal work by Song et al. [10], which demonstrated how a user could securely store encrypted data on a remote server and still be able to perform search queries on it without exposing the data itself. Since then, considerable research efforts have been devoted to improving and expanding SSE schemes. Some early works [76, 82] focused on designing secure schemes and formalising the security definitions, while some others such as [69, 114] introduced dynamic SSE, allowing for updates without needing to regenerate the encrypted index. In addition to supporting basic keyword-based search, additional functionalities were introduced to make SSE systems more expressive and efficient. For example, boolean queries were incorporated into SSE systems [27,33], allowing users to perform more complex searches involving multiple keywords and logical operators. These schemes have primarily focused on the

assumption of an honest-but-curious server model, where the server is assumed to follow the protocol for search execution but may attempt to learn sensitive information from the encrypted data and search patterns [76, 82]. To deal with dishonest servers, verifiable SSE schemes were introduced [4, 8, 21, 48, 89], which aim to ensure the correctness of search results. In VSSE systems, the client can verify whether the server has computed the correct search results, thus mitigating the risks of tampering or malicious behaviour by the server.

2.4 Encrypted Database

Similar to searchable encryption schemes, encrypted databases offer a solution to address the privacy concerns for searchable data. One of the early works in this area, CryptDB [1], introduced an encrypted relational database system that allows users to perform SQL queries on encrypted data without compromising confidentiality. This system implement a SQL interface that allows users to submit standard, unencrypted SQL queries, which are then transformed into encrypted queries that can be executed on the encrypted database. This approach preserves the usability of the database while ensuring data confidentiality. In addition to relational databases, other systems have been developed to support encrypted queries in NoSQL databases [2,85,108]. These systems aim to offer similar security benefits for non-relational data models such as key-value stores and graphs.

2.5 Verifiable Database

In addition to privacy, another crucial security property for database systems is integrity. Verifiable databases [13,19,28,40,58,81,87,98,113] aim to ensure the integrity of data operations, particularly in environments where the database server cannot be fully trusted. These systems extend traditional database models by incorporating cryptographic techniques that enable clients to verify the correctness of query results. To achieve this, verifiable database systems often employ authenticated data structures (ADS) [13, 40, 81]. Such systems are particularly useful in cloud computing, blockchain-based databases, and scenarios involving outsourced or federated data storage, where clients require assurance that the data has not been tampered with.

Chapter 3

Rethinking Practical Blockchain-Based Symmetric Searchable Encryption Services

In this chapter, we pinpoint three important requirements for blockchain-based SSE systems to be practical. The first two requirements are fast query response and low on-chain storage cost. They are the key for a blockchain-based SSE system to be usable and feasible. The third requirement is soundness, which guarantees that dishonest behaviours from both users and service providers can be detected and prevented. To the best of our knowledge, none of the existing studies achieves these properties at the same time. To fill in this gap, we present a novel construction of the blockchain-based SSE system. We provide both theoretical analysis and empirical evaluation to show that the system achieves all the desired properties.

3.1 Introduction

The advent of blockchain technology has introduced disruptive innovations to many areas. Cloud storage is one of these areas which has been developing rapidly in recent years [84, 99]. Blockchain-based cloud storage systems such as Filecoin [65], Storj [116], and Sia [70], have given users who want to store their data on the cloud a choice alternative to the traditional centralised service providers like Google and Amazon. These systems have also established platforms for individual storage providers where they can get rewarded for the outsourced storage. Without trusting these individual storage providers, users can encrypt (if not already enforced by the system) their data before uploading it to the system for confidentiality. The uploaded files can only be retrieved by their identifiers, since these systems do not support searching over the content of encrypted files. This

has significantly impaired the usability of the system as content search is an essential functionality of file systems. To address this problem, Symmetric Searchable Encryption (SSE) [69,82,114] technology can be used to enable the keyword search feature on encrypted files. With SSE combined with blockchain technology, previous studies [34, 79, 92] try to build blockchain-based SSE schemes to fill this gap for decentralised storage systems.

However, we pinpoint three key requirements for the blockchain-based SSE systems to be practical, and to the best of our knowledge, none of the existing work achieves all these requirements. The first requirement is that the expected response time of search queries needs to be short for the system to be usable. Compared to the traditional client-server structured SSE systems where the response time is mainly depending on the computation time and the network transfer time of the results, in the existing blockchain-based SSE systems [16, 79, 92] the queries are sent with blockchain transactions and the result computation happens after the transaction is confirmed, which introduces the transaction confirmation time as an extra overhead. Since this overhead is usually non-trivial (e.g. Bitcoin takes around 10 mins to confirm a transaction [52]), the usability of these systems is significantly impacted.

The second requirement is that the on-chain storage cost for the system needs to be efficient. Some previous studies [34, 79] build the system in a way that the SSE encrypted index is stored on the blockchain. This approach could be problematic due to the following reasons. On one hand, since the entire blockchain data is replicated in every full node in the blockchain network, and the encrypted index can be quite large for each user, the storage requirement for the blockchain nodes will quickly become unacceptable with the number of users increasing. On the other hand, the service fees for each individual user can be unaffordable, especially with the encrypted index stored inside the smart contract [16,79]. With the medium-sized Enron email dataset as an example, the encrypted index could be around 10 GB [33]. According to the Ethereum yellow paper [31], the recent gas rate [102] and the recent price of ETH [63], storing such an amount of data in the smart contract costs hundreds of millions in USD.

The third key requirement for the blockchain-based SSE system is that it should be secure against dishonest service providers and dishonest users, which is especially important in the blockchain P2P network scenario where neither of the parties can be trusted. While a dishonest service provider may be reluctant to return correct

search results after receiving the fees, a dishonest user may try to repudiate fees after getting the results. It turns out that there are trade-offs between this essential security requirement and the previous two. In order to improve the response time, the system can be designed such that the service provider computes and returns the results without waiting for the transaction confirmation. But in the case of a dishonest user who double-spends the fees, the security is breached because the transaction is not on-chain yet and thus not protected by the properties of the blockchain. In order to improve the on-chain storage cost, the encrypted index can be stored off-chain, while only the metadata is committed to the blockchain [92, 105]. But when a user claims that they didn't receive the correct results, it is difficult to determine which party was being dishonest. Although the service provider can prove that they have generated the correct results which match the metadata on the blockchain, it does not necessarily mean that these correct results have been sent to the user. Similarly, it could also be the user claiming that they did not receive the results while they actually did, trying to repudiate the fees.

In this research, we aim to address these trade-offs and build a blockchain-based SSE system that meets all the aforementioned requirements. To get optimal response time without enabling the user to double spend the service fees, our system requires the user to deposit preallocated service fees in the setup phase, which works similarly to payment channels. The service fee is deducted from the deposit for each query after the search transaction is confirmed. In this way, the service providers can answer the query before the transaction confirmation as long as there is enough deposit. To achieve on-chain storage efficiency, the results need to be sent off-chain, but at the same time their correctness and availability to the user need to be verifiable by the blockchain. With Publicly Verifiable SSE schemes, the correctness of the results can be verified publicly, but not the availability. For this purpose, we design a data structure *bulletin* for off-chain result storage. Combining the bulletin with PVSSE, the blockchain network can verify if the correct results are available to the user.

Contributions. As mentioned previously, the three requirements are crucial for all the blockchain-based SSE schemes that aim to be practical, and fulfilling all of them is challenging. In this research, we aim to address these challenges. Specifically, we make the following contributions:

- We study the secure blockchain-based SSE system, where both the user and the service provider can be dishonest. In our security model, we incorporate missing response attacks, where a malicious service provider withholds results for some queries. This aspect has been overlooked in previous studies but is of great importance. We formally define the refined security property as the soundness property. To satisfy soundness, we introduce the result bulletin and the heartbeat protocol, which allow the system to determine whether each query is answered by the service provider on time.

- Based on the result bulletin, we construct the first blockchain-based SSE system that achieves soundness and efficient on-chain storage simultaneously. Additionally, our system is the first blockchain-based SSE that that matches the response time of non-blockchain-based SSE schemes.

- We implement our system with Python and Ethereum blockchain. We run the implemented prototype in a private Ethereum blockchain network and evaluate the performance of our implementation. The experiments show that the response time, on-chain storage cost and user gas cost are improved compared to the previous blockchain-based SSE schemes.

3.2 Related Work

Blockchain-Based SSE. Blockchain-based SSE is a line of work that combines blockchain technology and Symmetric Searchable Encryption such that the SSE service is running on top of the blockchain system [16, 26, 34, 71, 79, 92, 105]. We categorise the current existing studies into the following two types:

On-chain Solutions. In this approach [16, 26, 34, 71, 79], the blockchain replaces the server in the traditional client-server SSE structure. The encrypted index is stored on the blockchain and the service fees are the smart contract fees and/or the transaction fees paid to the blockchain. While [26] and [34] only use the blockchain as cloud storage and require the data owners to compute the search results by themselves, [16, 71, 79] delegate this job to smart contracts such that the results are also computed by the blockchain. In this case, the on-chain storage complexity for each service is $O(n_w \cdot n_f)$, where n_w is the number of keywords and n_f is the number of file identifiers in the encrypted index. The total on-chain storage complexity for the blockchain system is $O(n_s \cdot n_w \cdot n_f)$, where n_s is the total number of SSE services in the system. Considering in a blockchain system, a copy

TABLE 3.1: Comparison of Blockchain-Based SSE.

Scheme	Response Time	On-Chain Storage	Soundness	
			U	S
[34]	-	$O(n_w \cdot n_f)$	-	-
[79]	$T + C$	$O(n_w \cdot n_f)$	✓	-
[16]	$T + C$	$O(n_w \cdot n_f)$	✓	-
[92]	$T + C$	$O(n_q + n_u)$	×	×
This work	C	$O(n_q + d)$	✓	✓

Soundness-U and Soundness-S are the soundness property against dishonest users and dishonest service providers respectively (section 3.3.4), n_w is the total number of keywords and n_f is the total number of files in the encrypted index, n_q is the total number of queries to the SSE service and n_u is the total number of update operations, d is the duration of the service in the number of epochs, T is the average transaction confirmation time, C is the average results computation time for a query plus the network transfer time for the results.

of the entire blockchain data is stored on every full node, this storage requirement is unacceptable. Also, by using such an amount of on-chain storage, especially within smart contracts, the fees can be unaffordable for each individual user.

Off-chain Solutions. This approach [92, 105] still has the role of service providers who are responsible for storing the encrypted index and computing the results, similarly to the traditional client-server SSE structure. The encrypted index and results are sent off-chain between the user and the service provider while only the metadata is committed to the blockchain. As an example, the on-chain storage complexity for [92], which supports dynamic index updates, is $O(n_q + n_u)$ for each service, where n_q is the number of queries and n_u is the number of update operations in the duration of the service. This linear growth of the on-chain storage makes the system more feasible, but as we discussed in section 3.1, the security becomes weaker compared to the on-chain solutions. Cai et al. [92] propose that when a dispute arises, the service provider sends the encrypted index to a group of trusted blockchain nodes and they re-execute the query on-chain, such that the user is guaranteed to get correct results. However, which party was dishonest remains unclear for the same reason mentioned in section 3.1, potentially leading to further attacks. Also, this operation is expensive in both computation and on-chain storage. Since the query will be re-executed on-chain, the storage complexity becomes $O(n_q \cdot n_r)$, where n_q is the number of queries executed on-chain and n_r is the number of file identifiers in the results for each query. Since n_q (resp., n_r) and n_w (resp., n_f) are in the same order, the on-chain storage complexity degrades to the same for an on-chain solution.

Symmetric Searchable Encryption. The notion of Symmetric Searchable Encryption was first introduced by Song et al. [10]. Afterwards, many efforts [69, 76, 82, 114] have been devoted to this research field to improve the security and efficiency of SSE schemes. Also, additional functionalities such as boolean queries [27, 33] have been proposed. These schemes only consider an honest-but-curious server that is interested in breaching the privacy of the client but will compute the search results honestly. To deal with dishonest servers, Verifiable SSE schemes were proposed [4, 8, 21, 48, 89], where the correctness of the search results can be verified.

Decentralised Storage Systems. Decentralised storage systems are cloud storage systems built on top of blockchain networks [65,70,116] The network is formed by storage providers who are willing to earn money by outsourcing their storage space and users who want to store files and pay for the service. For the integrity of the outsourced files, some of these systems [65,70] require the storage providers to submit proofs [36,39,94] to the blockchain that the files are intact and retrievable. For privacy, users can encrypt the files before uploading them to these systems. Due to this reason, the files are retrieved by their identifiers. The absence of the keyword search feature in these systems has motivated the studies on blockchain-based SSE systems.

3.3 System Model

3.3.1 System Overview

We consider a permissionless blockchain with three entities, namely service providers, users, and verifiers. A participant can play multiple roles. Service providers earn fees by providing SSE services to users. Verifiers, also known as miners in permissionless blockchains, maintain the blockchain (e.g. verifying transactions and creating blocks). In our system, verifiers additionally verify search results and help resolve disputes between service providers and users.

To achieve on-chain storage efficiency, we introduce an off-chain storage structure *bulletin* on the service provider side. The search results are published on the bulletin such that they do not flood the blockchain while still being verifiable by the verifiers. After answering each search query, the service provider creates a record on the bulletin containing the results. The digests of the bulletin are committed to the blockchain, which requires much less on-chain storage space.

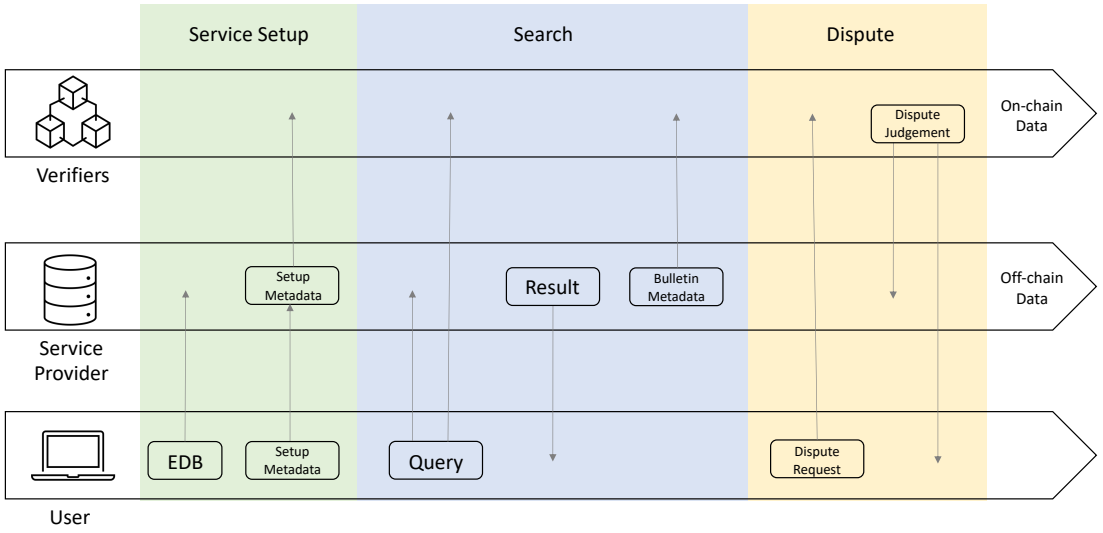


FIGURE 3.1: System Overview.

There are three main processes in the SSE service cycle, setup, search, and dispute. During the setup process, the user and the service provider both sign a transaction to set up an SSE service. A bulletin is established for the service and the address is included in the setup transaction. To prevent double spending, a deposit is included in the setup transaction for service fees. After the service is set up, the user can query the service by broadcasting a search transaction to the network. To achieve the optimal query response time, the service provider does not wait for the search transaction to be confirmed on-chain. Instead, the service provider begins to compute the search results as soon as they receive a valid search transaction. The service provider then sends the results to the user, and put the results on the bulletin for a period of time. After the search transaction is confirmed on-chain, the service fee is deducted from the deposit and transferred to the service provider. If the results are not available or appear to be incorrect, the user can raise a dispute against the provider. The dispute will be resolved by the verifiers. They check the results on the bulletin and the digests on the blockchain to judge if the service provider is misbehaving. If the service provider is found dishonest, they will be punished and the user will get back the service fee.

3.3.2 Syntax

The blockchain-based SSE system is a set of protocols $\Sigma = (\mathbf{Setup}, \mathbf{Search}, \mathbf{Dispute})$ running among three parties including a user $\mathbf{U}$, a service provider $\mathbf{S}$ and the verifiers $\mathbf{V}$:

- $(K; \mathtt{SD}; id_s, \mathtt{BC}') \leftarrow \textbf{Setup}(\mathtt{DB}; \mathtt{BC})$ is a protocol run among **U**, **S** and **V** to setup the SSE service. **U** takes as input a document collection DB. **V** takes as input the blockchain BC. At the end of the protocol, **U** outputs a secret key K for the encrypted index, **S** outputs some off-chain server-side data SD, **V** outputs the id of the service id_s and a new state of the blockchain $\mathtt{BC}'$.
- $(\{R, \perp\}; \mathtt{SD}'; id_q, \mathtt{BC}') \leftarrow \textbf{Search}(w, K; \mathtt{SD}; id_s, \mathtt{BC})$ is a protocol run among **U**, **S** and **V** to search a keyword in the encrypted index. **U** takes as input a keyword w and the secret key K. **S** takes as input the server-side data SD. **V** takes as input the id of the service id_s and the blockchain BC. At the end of the protocol, **U** outputs the search results R or $\perp$ where $R = \mathtt{DB}(w)$, **S** outputs a new state of the server-side data $\mathtt{SD}'$, **V** outputs the id of the query id_q and a new state of the blockchain $\mathtt{BC}'$.
- $(\{\mathbf{U}, \mathbf{S}\}, \mathtt{BC}') \leftarrow \textbf{Dispute}(id_q; \mathtt{BC}, \mathtt{SD})$ is a protocol run by **U** and **V** to raise and judge a dispute between **U** and **S**. **U** takes as input the query id id_q. **V** takes as input the blockchain BC and the server-side data SD. At the end of the protocol, **V** outputs a judgement on whether **U** or **S** is dishonest, and a new state of the blockchain $\mathtt{BC}'$.

3.3.3 Threat Model

We assume that both the user and the service provider may launch an attack when they can get some profit by doing it. For the service provider, we consider two main threats which are incorrect result and missing response. For the user, fraudulent disputes are considered the main threat. The system requires collaterals from the users and the service providers. If they behave honestly, the collaterals will be returned to them after the service is finished. If dishonest behaviours are detected, the collaterals are taken. For the verifiers, we follow the conventional "honest majority" assumption [17] in the blockchain area, thus at least 50% of the miners in the system are honest.

Incorrect Result. The service provider may delete the encrypted index or a part of it to reduce the storage cost, which makes the service provider not able to compute the search results for some queries. Even if the service provider can, they may be reluctant to do it because of the processing cost. In the case of a user query, the service provider can generate some results arbitrarily and answer the query with these incorrect results. With the bulletin and the underlying SSE scheme being verifiable, the mitigation to this attack can be straightforward.

Missing Response. For the same reason mentioned above, the service provider may not have the correct results for some queries. Instead of forging the results, they can simply ignore the queries. We have briefly discussed why this attack is more challenging than the previous one in section 3.1. By introducing the bulletin, our system also provides the evidence of whether the results are available to the user. However, what the service provider still can do is ignoring the search query until a dispute request is sent by the user. To pass the verification, the service provider computes the results and puts them on the bulletin immediately before the verification. This attack makes the results only available to the user during a dispute verification. To guarantee the user be able to retrieve the results whenever they want to, we consider a generalised version of this threat: at any time, the correct results are not available on the bulletin. Since the dispute operation can be expensive and hence frequently doing it is infeasible, preventing this attack is challenging.

Fraudulent Dispute. Users are on the weaker side in the system since they need to pay the fees first to get their query processed. But a dishonest user may try to raise a fraudulent dispute after receiving the correct results. If the fraudulent dispute is not judged correctly, the user can get back the service fee, which breaches the security of the system.

3.3.4 Design Goals

Our main goal is to achieve optimal query response time, on-chain storage efficiency and security at the same time. For the response time, we aim to eliminate the transaction confirmation overhead such that the response time is the same as the traditional non-blockchain-based SSE schemes. For the on-chain storage efficiency, we have discussed in section 3.1 that storing the encrypted index or the search results on the blockchain is not efficient. The size of the data submitted to the blockchain for each operation should only be depending on the system security parameter, which can be considered constant after the system is set up. For security, we aim to achieve soundness and privacy.

Soundness. There are three main threats from dishonest service providers and dishonest users, which are discussed in section 3.3.3. The system should be secure against these attacks. If the attacks are accountable, they can be prevented by the collateral mechanism based on our assumptions. This requires the dispute protocol

always outputs the correct judgement. We define this ability of the system to detect the dishonest behaviours as the soundness property.

Definition 3.1 (Soundness)**.** For the blockchain-based SSE system $\Sigma = (\mathbf{Setup}, \mathbf{Search}, \mathbf{Dispute})$, let id_q be the id of some query processed by the search protocol $(\{R, \perp\}; \mathtt{SD}'; id_q, \mathtt{BC}') \leftarrow \mathbf{Search}(w, K; \mathtt{SD}; id_s, \mathtt{BC})$ for some keyword w, where K is the secret key and $\mathtt{SD}$ is the server-side data for some service id_s established with the setup protocol $(K; \mathtt{SD}; id_s, \mathtt{BC}') \leftarrow \mathbf{Setup}(\mathtt{DB}; \mathtt{BC})$ for some database $\mathtt{DB}$. We say that the blockchain-based SSE system Σ is sound if $\forall w$, $\forall \mathtt{DB}$, when the result output is $\perp$, the probability that the dispute protocol $(\{\mathbf{U}, \mathbf{S}\}, \mathtt{BC}') \leftarrow \mathbf{Dispute}(id_q; \mathtt{BC}, \mathtt{SD})$ does not output $\mathbf{S}$ is negligible, and when the result output is R where $R = \mathtt{DB}(w)$, the probability that the dispute protocol $(\{\mathbf{U}, \mathbf{S}\}, \mathtt{BC}') \leftarrow \mathbf{Dispute}(id_q; \mathtt{BC}, \mathtt{SD})$ does not output $\mathbf{U}$ is negligible.

Privacy. The privacy property of an SSE scheme ensures that no useful information is leaked, which can lead to the direct exposure of the searched keyword and plaintext data. For this property, we follow the definition presented in [33, 45].

3.4 Preliminaries

3.4.1 Publicly Verifiable SSE

A Publicly Verifiable SSE (PVSSE) scheme is a Verifiable SSE scheme that the correctness of results can be verified by the public [44,45,86], while in a traditional VSSE [4] the results can only be verified by one who has the secret key of the data owner. During the setup phase of the scheme, a verification key is generated and given to the public. When searching a keyword, a proof is generated together with the search results. The proof can be used to verify the correctness of the results with the verification key. In our system, the PVSSE is used to detect incorrect results from dishonest providers. A PVSSE scheme consists of a set of algorithms $\Pi = (\mathbf{SSESetup}, \mathbf{SSEGenSearchToken}, \mathbf{SSESearch}\ , \mathbf{SSEVerify})$ running among three parties including a user, a service provider and a verifier:

- $(K, vk, \mathtt{EDB}) \leftarrow \mathbf{SSESetup}(\mathtt{DB})$ is an algorithm run by the user to setup the encrypted database. It takes as input a document collection $\mathtt{DB}$ and outputs an encrypted database $\mathtt{EDB}$, a secret key K and a verification key vk. Note that the vk can be empty depending on the construction of the scheme.

- $\tau \leftarrow$ **SSEGenSearchToken**(w, K) is an algorithm run by the user to generate a search token. It takes as input a keyword w and the user's secret key K, and outputs a search token τ corresponding to w.

- $(R, \pi) \leftarrow$ **SSESearch**$(\tau, \texttt{EDB})$ is an algorithm run by the service provider to do the search. It takes as input a search token τ and the index $\texttt{EDB}$. After running, it outputs the results R, and the corresponding proof π, to verify R.

- $\{ACCEPT, REJECT\} \leftarrow$ **SSEVerify**(τ, R, π, vk) is an algorithm run by the verifier, which can be anyone who has the vk, to determine whether R is the correct answer to τ.

Correctness. A PVSSE scheme is correct if for all $\texttt{DB}$ and w the probability that the search results $R \neq \texttt{DB}(w)$, or the verification outputs $REJECT$ is negligible, given all the parties follow the protocol honestly.

Soundness. A PVSSE scheme is sound if for all $\texttt{DB}$ and w the probability that the verification outputs $ACCEPT$ while $R \neq \texttt{DB}(w)$ is negligible.

Confidentiality. The confidentiality is defined with the two games $\mathbf{SSEReal}^{\mathbf{\Pi}}_{\mathcal{A}}(\lambda)$ and $\mathbf{SSEIdeal}^{\mathbf{\Sigma}}_{\mathcal{A},\mathcal{S}}(\lambda)$. While in $\mathbf{SSEReal}^{\mathbf{\Pi}}_{\mathcal{A}}(\lambda)$ the actual SSE protocol is executed, in $\mathbf{SSEIdeal}^{\mathbf{\Sigma}}_{\mathcal{A},\mathcal{S}}(\lambda)$ the protocol transcript is generated by a simulator. A PVSSE scheme is secure if the probability that any P.P.T. adversary is able to distinguish the two games is negligible.

3.4.2 Blockchain Randomness Beacon

A blockchain randomness beacon (BRB) is a blockchain system that produces a public random value in each epoch, which is unpredictable in advance and immutable once produced [6]. In our system, the BRB is used to provide unpredictable randomness to prevent precomputations of committed bulletin digests. It is defined as the following:

Definition 3.2 (BRB)**.** A BRB consists of an algorithm **GetRandomness** used to extract randomness value from the beacon:

- $seed_t \leftarrow$ **GetRandomness**$(t, \texttt{BC})$ is an algorithm run by anyone who can access the blockchain. It takes as input an epoch number t up to the current

epoch and the blockchain data BC, then output the random value $seed_t$ of the randomness beacon for epoch t.

Unpredictability. Let t' be the latest epoch number of the BRB, the BRB is unpredictable if for all $t > t'$ the probability that the value $seed_t$ can be computed by any P.P.T adversary is negligible.

3.5 System Design

3.5.1 Result Bulletin

A result bulletin is created for each SSE service during the setup process by the service provider **S**. It is a versioned public cloud storage space for the user to get search results. Specifically, it consists of a set of records and a set of snapshots:

$$bul = ((rec_0, rec_1, ..., rec_n), (ss_{t-1}, ss_{t-2}, ..., ss_{t-l}))$$

Each record $rec_i, i \in [0, n]$ on the bulletin is a tuple (id_q, R, π), where n is the total number of records on the bulletin, id_q is the query id indicating which query this record is answering, R is the result and π is the corresponding proof. The records are sorted based on id_q. Each snapshot $ss_i, i \in [t-1, t-l]$ is a historical view of the bulletin at epoch i, which contains the references to all the records on the bulletin at epoch i. The number t is the current epoch number, and l is the lifetime of a record and a snapshot. After l epochs, a record or a snapshot can be deleted from the bulletin.

In each epoch t, the service provider **S** needs to run **Heartbeat** protocol to commit the status of the bulletin to the blockchain. The service provider first makes sure that all known queries (i.e. queries that are on-chain) to her service id_s are answered on the bulletin. Then they can delete the records that have exceeded the lifetime, and creates a snapshot ss_t for the bulletin. Finally, the service provider creates and signs a heartbeat transaction and broadcasts it to the blockchain network.

$$\mathbf{Trans_{hb}} = (id_s, h_{hb})$$

In the above heartbeat transaction, id_s is the id of the service, and h_{hb} is the heartbeat value. The heartbeat value is computed as $\mathbf{H}(seed_t||ss_t)$, where $\mathbf{H}$ is a cryptographic hash function, $seed_t$ is the output of BRB at epoch t, and ss_t is the snapshot of the bulletin at epoch t.

Algorithm 1: Heartbeat

SERVICE PROVIDER **S**:
1. $ss_t \leftarrow \emptyset$
2. **foreach** $rec_i \in bul$ **do**
3. add rec_i to ss_t
4. **end**
5. $seed_t \leftarrow$ **GetRandomness**$(t, \texttt{BC})$
6. $h_{hb} \leftarrow \mathbf{H}(seed_t||ss_t)$
7. Prepare $\mathbf{Trans_{hb}} = (id_s, h_{hb})$
8. Sign the transaction and get $\langle \mathbf{Trans_{hb}} \rangle_{\mathbf{S}}$
9. Broadcast the signed transaction $\langle \mathbf{Trans_{hb}} \rangle_{\mathbf{S}}$ to the network

VERIFIERS **V**:
10. **if** $\langle \mathbf{Trans_{hb}} \rangle_{\boldsymbol{S}}$ *is valid* **then**
11. Include $\langle \mathbf{Trans_{hb}} \rangle_{\mathbf{S}}$ in the blockchain BC
12. **end**

3.5.2 Setup

To set up an SSE service, a user **U** first runs **SSESetup** with some DB to get EDB, vk and K. Afterwards, the user finds a service provider **S** in the marketplace that suits her needs. **U** sends the EDB to **S**, **S** prepares a bulletin bul. **S** stores EDB and bul as the server-side data. They both sign a setup transaction and broadcast the transaction to the blockchain network.

$$\mathbf{Trans_{setup}} = (h_{\texttt{EDB}}, d, vk, addr, dep)$$

In the above setup transaction, $h_{\texttt{EDB}}$ is the digest of EDB, d is the duration of the service, vk is the verification key of EDB, $addr$ is the address of the bulletin, and dep is the deposited service fee.

3.5.3 Search

To search in an established service id_s, the user **U** first generates a search token τ by running **SSEGenSearchToken** with the key K for id_s and a chosen keyword w. After the user has the token, they create and sign a search transaction and broadcasts it to the blockchain network.

Algorithm 2: Setup

USER **U**:
1. $(K, vk, \texttt{EDB}) \leftarrow \textbf{SSESetup}(\texttt{DB})$
2. $h_{\texttt{EDB}} \leftarrow \textbf{H}(\texttt{EDB})$
3. Choose duration d for the service
4. Prepare enough balance dep
5. Send EDB to **S**

SERVICE PROVIDER **S**:
6. Prepare a bulletin bul and get $addr$

USER **U** AND SERVICE PROVIDER **S**:
7. Prepare $\textbf{Trans}_{\textbf{setup}} = (h_{\texttt{EDB}}, d, vk, addr, dep)$
8. Sign the transaction and get $\langle \textbf{Trans}_{\textbf{setup}} \rangle_{\textbf{U},\textbf{S}}$
9. Broadcast the signed transaction $\langle \textbf{Trans}_{\textbf{setup}} \rangle_{\textbf{U},\textbf{S}}$ to the network

VERIFIERS **V**:
10. **if** $\langle \textbf{Trans}_{\textbf{setup}} \rangle_{U,S}$ *is valid* **then**
11. Include $\langle \textbf{Trans}_{\textbf{setup}} \rangle_{\textbf{U},\textbf{S}}$ in the blockchain BC
12. Output the id of $\textbf{Trans}_{\textbf{setup}}$ as id_s
13. **end**

$$\textbf{Trans}_{\textbf{search}} = (id_s, \tau)$$

In the above search transaction, id_s is the id of the service, and τ is the search token. Upon receiving this transaction, the service provider **S** runs the **SSESearch** with τ and the EDB of id_s to compute search results R and the proof π. Then **S** sends the results (R, π) back to the user **U**. After **S** sees the transaction included on the blockchain, they create a record (id_q, R, π) on the bulletin bul of id_s, where id_q is the id for the search transaction.

If **S** is dishonest, **U** may not receive the results. In this case, **U** can wait until the search transaction is included in the blockchain and then try to retrieve the results from the bulletin. If the results are not available on the bulletin either, **U** can run the **Dispute** protocol. Another case where **U** needs to run **Dispute** protocol is that the results are incorrect. This can be checked with **SSEVerify** by **U** after receiving the results.

3.5.4 Dispute

Within the lifetime l of a query id_q, the user **U** can raise a dispute for the query. They create and sign a dispute transaction and broadcasts it to the blockchain network.

$$\textbf{Trans}_{\textbf{dispute}} = (id_q)$$

Algorithm 3: Search

USER **U**:
1. $\tau \leftarrow$ **SSEGenSearchToken**(w, K)
2. Prepare $\mathbf{Trans_{search}} = (id_s, \tau)$
3. Sign the transaction and get $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$
4. Broadcast the signed transaction $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$ to the network

SERVICE PROVIDER **S**:
5. Upon receiving $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$, get τ from the transaction
6. $(R, \pi) \leftarrow$ **SSESearch**(τ, EDB)
7. Send (R, π) to **U**
8. **while** $\langle \mathbf{Trans_{search}} \rangle_{U} \notin BC$ **do**
9. Broadcast $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$ and wait
10. **end**
11. Get id_q for $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$ from BC
12. Create a record (id_q, R, π) on the bulletin *bul*

USER **U**:
13. **if** (R, π) *is not received from* ***S*** **then**
14. **while** $\langle \mathbf{Trans_{search}} \rangle_{U} \notin BC$ **do**
15. Broadcast $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$ and wait
16. **end**
17. **if** $(R, \pi) \notin bul$ **then**
18. Output $\perp$ and abort
19. **end**
20. Retrieve (R, π) from *bul*
21. **end**
22. Retrieve vk from BC
23. **if** **SSEVerify**$(\tau, R, \pi, vk) \neq ACCEPT$ **then**
24. Output $\perp$ and abort
25. **end**
26. Output R

VERIFIERS **V**:
27. **if** $\langle \mathbf{Trans_{search}} \rangle_{U}$ *is valid* **then**
28. Include $\langle \mathbf{Trans_{search}} \rangle_{\mathbf{U}}$ in the blockchain BC
29. Output the id of $\mathbf{Trans_{search}}$ as id_q
30. **end**

In the above dispute transaction, id_q is the id of the query in question. After this transaction is included in the blockchain, the verifiers **V** will do three checks to resolve the dispute. First, they check if there is a record for the query id_q on the bulletin and check if the results are correct by running **SSEVerify**. Then they check if the same record is in every snapshot after the query is on-chain. Finally, they check if there is any missing heartbeat or invalid heartbeat values. If any check does not pass, a consensus is made by **V** that **S** is dishonest. Otherwise, **U** is considered dishonest due to raising a fraudulent dispute.

Algorithm 4: Dispute

USER **U**:

Prepare $\mathbf{Trans_{dispute}} = (id_q)$
Sign the transaction and get $\langle \mathbf{Trans_{dispute}} \rangle_{\mathbf{U}}$
Broadcast the signed transaction $\langle \mathbf{Trans_{dispute}} \rangle_{\mathbf{U}}$ to the network

VERIFIERS **V**:

if $\langle \mathbf{Trans_{dispute}} \rangle_{U}$ *is valid* **then**
  Include $\langle \mathbf{Trans_{dispute}} \rangle_{\mathbf{U}}$ in the blockchain BC
end
Get id_q from $\mathbf{Trans_{dispute}}$
Get $\mathbf{Trans_{search}}$ from BC where $id = id_q$
Get id_s, τ from $\mathbf{Trans_{search}}$
Get $\mathbf{Trans_{setup}}$ from BC where $id = id_s$
Get vk, $addr$ from $\mathbf{Trans_{setup}}$
Retrieve bul at $addr$
if $id_q \in bul$ **then**
  Get the record (id_q, R, π) from bul
  if $\mathbf{SSEVerify}(\tau, R, \pi, vk) = REJECT$ **then**
    Output **S** and abort `// invalid results`
  end
else
  Output **S** and abort `// missing results`
end
$t \leftarrow$ the current epoch number
$t' \leftarrow$ the epoch number of $\mathbf{Trans_{search}}$
foreach *epoch* $i \in [t', t]$ **do**
  Get ss_i from bul
  if $id_q \notin ss_i$ **then**
    Output **S** and abort `// missing results at epoch` i
  end
  if $\nexists \mathbf{Trans_{hb}}(\mathbf{Trans_{hb}} \in BC[i], id_s \in \mathbf{Trans_{hb}})$ **then**
    Output **S** and abort `// missing heartbeat`
  end
  Get h_{hb} from $\mathbf{Trans_{hb}} \in \mathtt{BC}[i]$ for id_s
  $seed_i \leftarrow \mathbf{GetRandomness}(i, \mathtt{BC})$
  if $\mathbf{H}(seed_i || ss_i) \neq h_{hb}$ **then**
    Output **S** and abort `// invalid heartbeat value`
  end
end
S is honest, output **U**

3.6 Security Analysis

Soundness. To analyse the soundness property, we discuss the two different cases separately. In the first case, where the result output is $\perp$, the **Dispute** protocol should output **S** with overwhelming probability. For this case, we consider a P.P.T adversary running in time $Poly(\lambda)$ who can tamper with and recover the bulletin at any epoch t. Based on our construction, we consider that the user may try to retrieve the results from the bulletin at any time within the lifetime of the query. The correct results should be available on the bulletin at all times within the

lifetime, otherwise the result output is considered $\perp$, and the **Dispute** protocol should output **S** for this query. Let id_s be the id of a service established with some database DB. Let id_q be the id of a query for the service id_s with some keyword w, which is on-chain at some time t. Let a dispute transaction for the query id_q be on-chain at some epoch t_D, where $t+1 \leq t_D \leq t+l$. For all DB, w and t_D, we have:

Lemma 3.3. *If the correct results for the query id_q cannot be retrieved at epoch t_D, the probability that the* **Dispute** *protocol does not output **S** is negligible, if the PVSSE scheme Π is sound.*

Proof. Let *bul* be the result bulletin for id_s at epoch t_D, and $rec = (id_q, R, \pi)$ be the record on the bulletin for id_q. Given the correct results for the query id_q cannot be retrieved from the bulletin at epoch t_D, we have $R \neq \mathtt{DB}(w)$. If the **Dispute** protocol does not output **S**, we have $\mathbf{SSEVerify}(R, \pi, \cdot) = ACCEPT$. In this case,

$$\begin{aligned} &\Pr(\mathbf{Dispute}(id_q, \cdot) \neq \mathbf{S}) \\ =&\Pr(\mathbf{SSEVerify}(R, \pi, \cdot) = ACCEPT \mid R \neq \mathtt{DB}(w)) \\ \leq&negl(\lambda) \end{aligned}$$

□

Lemma 3.4. *If the correct results for the query id_q can be retrieved at epoch t_D, but cannot be retrieved at epoch t_M, where $t+1 \leq t_M < t_D$, the probability that the* **Dispute** *protocol does not output **S** is negligible, if the hash function* **H** *is collision-resistant and the BRB is unpredictable.*

Proof. Let ss_{t_M} be the snapshot of the bulletin at epoch t_M, and $hb_{t_M} = \mathbf{H}(seed_{t_M} || ss_{t_M})$ be the hearbeat value on the blockchain. Given the correct results for the query id_q cannot be retrieved from the bulletin at epoch t_M, we have $rec^{corr} \notin ss_{t_M}$. At the same time, we assume that $ss^{corr}_{t_M}$ is the correct snapshot at epoch t_M with $rec^{corr} \in ss^{corr}_{t_M}$, and $hb^{corr}_{t_M} = \mathbf{H}(seed_{t_M} || ss^{corr}_{t_M})$ is the corresponding heartbeat value. Hence we get $ss_{t_M} \neq ss^{corr}_{t_M}$. If the **Dispute** does not output **S**, we have $hb_{t_M} = hb^{corr}_{t_M}$. In this case,

$$
\begin{aligned}
&\Pr(\mathbf{Dispute}(id_q, \cdot) \neq \mathbf{S}) \\
=&\Pr(hb_{t_M} = hb_{t_M}^{corr} \mid ss_{t_M} \neq ss_{t_M}^{corr}) \\
=&\Pr(\mathbf{H}(seed_{t_M}||ss_{t_M}) = \mathbf{H}(seed_{t_M}||ss_{t_M}^{corr}) \mid ss_{t_M} \neq ss_{t_M}^{corr}) \\
\leq& negl(\lambda)
\end{aligned}
$$

The heartbeat value may be computed before epoch t_M when $ss_{t_H} = ss_{t_M}^{corr}$, where $t+1 \leq t_H < t_M$. However, at this time the value $seed_{t_M}$ is not known yet, which the adversary have to make a guess. Assuming the guessed randomness value is $seed_{t_M}^{t_H}$, we have $hb_{t_M} = \mathbf{H}(seed_{t_M}^{t_H}||ss_{t_H})$. If the **Dispute** does not output **S**, we have $hb_{t_M} = hb_{t_M}^{corr}$. In this case,

$$
\begin{aligned}
&\Pr(\mathbf{Dispute}(id_q, \cdot) \neq \mathbf{S}) \\
=&\Pr(hb_{t_M} = hb_{t_M}^{corr} \mid t_H < t_M) \\
=&\Pr(\mathbf{H}(seed_{t_M}^{t_H}||ss_{t_H}) = \mathbf{H}(seed_{t_M}||ss_{t_M}^{corr}) \mid t_H < t_M) \\
=&\Pr(\mathbf{H}(seed_{t_M}^{t_H}||ss_{t_M}^{corr}) = \mathbf{H}(seed_{t_M}||ss_{t_M}^{corr}) \mid t_H < t_M) \\
=&\Pr(seed_{t_M}^{t_H} = seed_{t_M} \mid t_H < t_M) \\
\leq& negl(\lambda)
\end{aligned}
$$

□

In the second case, where the result output is R where $R = \mathtt{DB}(w)$, the **Dispute** protocol should output **U** with overwhelming probability. For this case, we consider a P.P.T adversary running in time $Poly(\lambda)$ who can query the service with any chosen keyword w, and adaptively choose id_q to run **Dispute** protocol given the bulletin records. Based on our construction, in this case the user is able to retrieve the correct results from the bulletin at any epoch within the lifetime of the query. Let id_s be the id of a service established with some database $\mathtt{DB}$. Let id_q be the id of a query for the service id_s with some keyword w, which is on-chain at some time t. Let a dispute transaction for the query id_q be on-chain at some epoch t_D, where $t+1 \leq t_D \leq t+l$. For all $\mathtt{DB}$, w and t_D, we have:

Lemma 3.5. *If the correct results for the query id_q can be retrieved at all epoch t_H, where $t+1 \leq t_H \leq t_D$, the probability that the* **Dispute** *protocol does not output* ***U*** *is negligible, if the PVSSE scheme* Π *is correct.*

Proof. Let *bul* be the result bulletin for id_s at epoch t_D, ss_{t_H} be the snapshot for the bulletin at epoch t_H, and $rec = (id_q, R, \pi)$ be the record for id_q. Given the correct results for the query id_q can be retrieved from the bulletin at all epoch t_H, we have $R = \mathtt{DB}(w)$ and $rec \in ss_{t_H}$ for all t_H. If the **Dispute** protocol does not output $\mathbf{U}$, we then have $\mathbf{SSEVerify}(R, \pi, \cdot) = REJECT$. In this case,

$$\begin{aligned} &\Pr(\mathbf{Dispute}(id_q, \cdot) \neq \mathbf{U}) \\ =&\Pr(\mathbf{SSEVerify}(R, \pi, \cdot) = REJECT \mid R = \mathtt{DB}(w)) \\ \leq& negl(\lambda) \end{aligned}$$

□

Theorem 3.6. *If the PVSSE scheme* Π *is correct and sound, the hash function* **H** *is collision-resistant, and the BRB is unpredictable, then the blockchain-based SSE system* Σ *is sound.*

Proof. Concluding the lemmas, given that the PVSSE scheme Π is sound, the hash function **H** is collision-resistant, and the BRB is unpredictable, if the correct search results are unretrievable from the bulletin at any epoch, the probability that the **Dispute** protocol does not output **S** is negligible (lemma 3.3 and 3.4). Given that the PVSSE scheme Π is correct, if the correct search results are retrievable from the bulletin at all epochs, the probability that the **Dispute** protocol does not output **U** is negligible (lemma 3.5). □

Privacy. For privacy, we consider a P.P.T adversary $\mathcal{A}$ who is capable of running adaptive chosen-keyword attacks. In our model, $\mathcal{A}$ is trying to distinguish the real bulletin and protocol transactions from simulated ones. Let $\mathcal{L}_{setup}$ and $\mathcal{L}_{search}$ be the leakage functions for **SSESetup** and **SSESearch** protocols of the PVSSE scheme Π respectively, we have:

Theorem 3.7. *If the PVSSE scheme* Π *is* $(\mathcal{L}_{setup}, \mathcal{L}_{search})$*-secure against adaptive chosen-keyword attacks, the blockchain-based SSE system* Σ *is* $(\mathcal{L}_{setup}, \mathcal{L}_{search})$*-secure against adaptive chosen-keyword attacks.*

Proof. The simulator can generate dummy $(vk^*, \texttt{EDB}^*)$ values given $\mathcal{L}_{setup}$, and (τ^*, R^*, π^*) values given $\mathcal{L}_{search}$, for any $\texttt{DB}$ and w, which $\mathcal{A}$ can distinguish from the corresponding real protocol outputs with negligible probability, given Π is $(\mathcal{L}_{setup}, \mathcal{L}_{search})$-secure against adaptive chosen-keyword attacks. To generate the dummy bulletin, the simulator replaces the (R, π) with (R^*, π^*) for each record $rec \in bul$. To generate the dummy Heartbeat transaction transcript, the generator computes each heartbeat value with the dummy bulletin records for each epoch t. To generate the dummy Setup transaction, the generator computes $h_{\texttt{EDB}^*}$ with $\texttt{EDB}^*$, and replace vk with vk^*. To generate the dummy Search transaction, the generator replace τ with τ^*. The rest intercations are the same as the real protocol. Since the probability that $\mathcal{A}$ can distinguish $(vk^*, \texttt{EDB}^*, \tau^*, R^*, \pi^*)$ from $(vk, \texttt{EDB}, \tau, R, \pi)$ is negligible, the probability $\mathcal{A}$ can distinguish the simulated bulletin and protocol transactions from the real ones is negligible. □

3.7 Implementation and Performance Evaluation

We implement a prototype of the system with Python and Ethereum blockchain. The result bulletin is implemented as a web service running on the service provider side. Each record is written to the web directory with search id as the identifier and is retrieved via HTTP protocol. For the randomness beacon, we use Drand [11] and get the randomness output from the nodes using its Python client package. For the PVSSE scheme, we implement the second construction proposed in [45] which can build on top of any existing SSE scheme and make it publicly verifiable. The underlying SSE scheme we choose to implement is adopted from [69]. We implement the PRF with HMAC-SHA256 and PRP with AES. The hash function, symmetric encryption scheme and signature scheme are SHA-256, AES-CBC with 256-bit keys and RSA with 2048-bit keys from the Python Crypto library. A smart contract is deployed on the blockchain to maintain the on-chain data for the system. All the protocol transactions (e.g. Setup, Search, etc.) are sent to this smart contract.

We establish a private Ethereum network with 6 nodes running the official Go Ethereum programme [78]. The blockchain is a private chain mined from a custom genesis block and the block interval is adjusted to around 15 seconds, which is similar to the public Ethereum blockchain. These nodes also run the Drand node programme to produce randomness beacon values. All the nodes work as the verifier role in our system to resolve disputes and one of them also acts as the

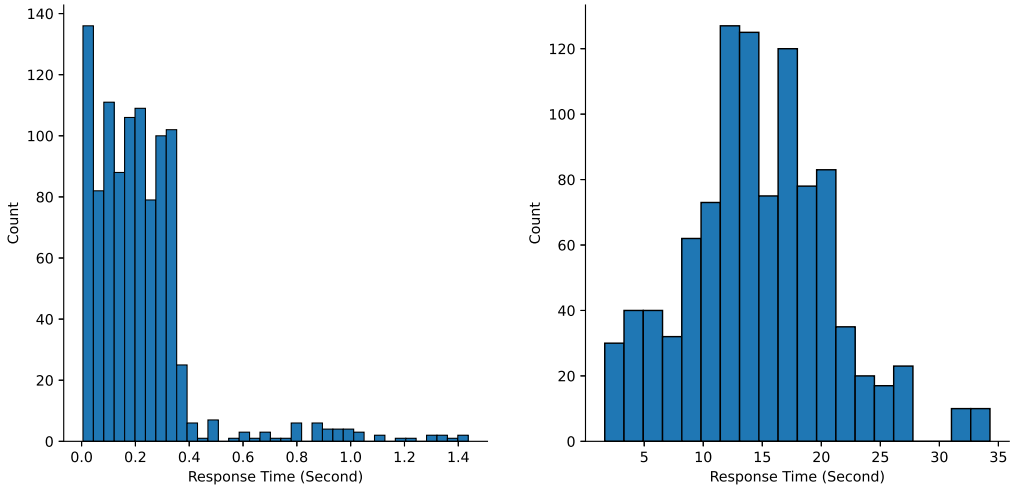
Table 3.2: Service Setup Time For Different Databases.

Num of (w, id) Pairs	EDB Size	EDB Building	EDB Serialisation	Setup Transaction
1,000	73kB	1.2s	0.003s	13.7s
16,000	1.1MB	9.8s	0.01s	13.3s
128,000	9MB	65s	7.3s	13.5s
256,000	18MB	137s	37s	12.9s
512,000	36MB	277s	173s	14.5s
1,024,000	73MB	545s	689s	13.1s

service provider to provide the SSE service. A client is connected to the network via another node that is not the service provider. The client sends transactions through the HTTP-API provided by the Go Ethereum programme running on the connected node. All these nodes and the client are virtual machines with Linux operating system running on a host machine with i9-9900 CPU and 32 GB memory. Each virtual machine is assigned 2 virtual CPU cores and 2GB memory, and they are connected via a 100 Mb/s virtual ethernet. We conduct a series of experiments to evaluate the performance of the implemented prototype on service setup time, query response time and on-chain storage cost.

3.7.1 Service Setup Time

The service setup process can be broken down into the following tasks. First, the user builds the EDB from her plain database, which includes generating the encrypted index for the scheme [69] and the public verification data for the scheme [45]. Second, the EDB is serialised with the Python Pickle package such that it can be stored in the file system or sent over the network. After the serialisation, the EDB is sent to the service provider. For the EDB sizes that we test, sending the data is always within a few seconds via the virtual network. But it turns out that the data serialisation time is non-trivial when the EDB is large. At last, a Setup transaction is sent to the blockchain. After the transaction is confirmed, the service setup process is completed. The average confirmation time for this transaction is the block interval of the blockchain when the network is not overwhelmed by transactions. We generate several databases with different numbers of (w, id) pairs. For each database, we repeat the setup process 100 times. Table 3.2 shows the average time taken for the tasks for each database. The transaction confirmation overhead introduced by our system is constant.



(a) Off-Chain Mode (b) On-Chain Mode

FIGURE 3.2: Response Time Distribution

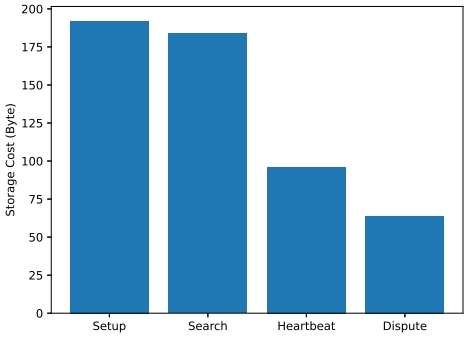


FIGURE 3.3: On-Chain Storage

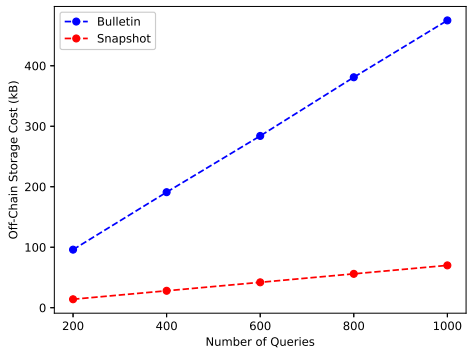


FIGURE 3.4: Off-Chain Storage

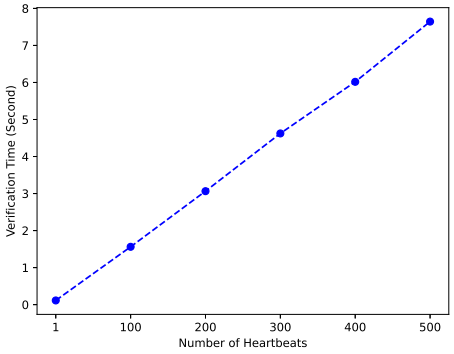


FIGURE 3.5: Verification Time

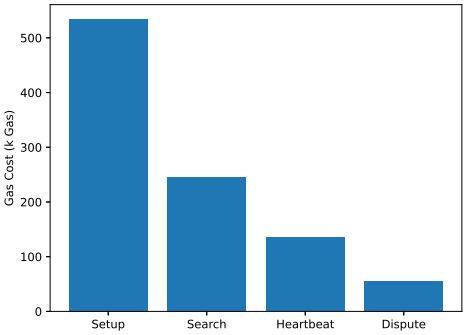


FIGURE 3.6: Gas Cost

3.7.2 Query Response Time

The prototype is implemented such that the service provider can run in two different modes. In the on-chain mode, the service provider gets the query from the

on-chain search transactions, which behaves like the previous schemes [16, 79, 92]. On the other hand, the off-chain mode is the behaviour described in our protocol, where the service provider gets the query from the broadcasted but not yet confirmed valid search transactions. We randomly select 1000 keywords from the original database, and send the queries to two service providers running in the two different modes. The response time distributions are shown in figure 3.2. In off-chain mode, for most of the queries the response time is less than 0.5 second, whereas in on-chain mode, the average response time is around 15 seconds because of the transaction confirmation. It shows that our system has a significant improvement in the query response time compared to the previous blockchain-based SSE schemes.

3.7.3 On-Chain Storage Cost

From the presented construction of the blockchain-based SSE system Σ, only the search token is stored on the blockchain for each query. In addition, a heartbeat value is stored on the blockchain in each epoch. In this case, the total on-chain storage complexity for each service is $O(n_q + d)$, where n_q is the total number of queries for the service and d is the service duration in the number of epochs. Compared to the multilinear complexity of on-chain solutions (section 3.2), this is a significant improvement. Figure 3.3 shows the experimental results of the on-chain storage cost. We can see that any of the 4 operations only sends data less than 200 bytes, which is trivial compared to the on-chain solutions.

3.7.4 Other Experimental Results

Figures 3.4, 3.5 and 3.6 show the off-chain storage cost, the verification time and the gas cost respectively. The extra off-chain storage space taken on the service provider side other than storing the EDB is depending on the frequency of search queries. Figure 3.4 shows how much space the result bulletin and a snapshot take when the number of queries within the lifetime l increases. When there are 1000 queries within 500 epochs, the size of the bulletin and a snapshot is around 470 kB and 70 kB. We consider this off-chain storage overhead acceptable given the capacity and price of modern storage devices. For the verification time, in the setting ($l = 500$) at most 500 heartbeats are verified. The results show that the longest verification takes less than 8 seconds. The gas cost for a search in our system is similar to the previous off-chain solution [92], and is around 1/20 of the

previous on-chain solution [79]. When comparing heartbeat versus committing results for every search query [92], if the user queries more frequently than the heartbeats happen (i.e. more than one query per epoch), our system will have a lower cost, and vice versa.

3.8 Conclusion

In this research, we study the practical and secure blockchain-based SSE systems. We analyse the previous studies on this topic and show they cannot achieve both on-chain storage efficiency and soundness at the same time. We construct a blockchain-based SSE system with the proposed off-chain data structure bulletin, publicly verifiable SSE and randomness beacon. We show that our construction achieves the security and performance requirements. With pre-allocated service fees, our system also achieves optimal query response time which is independent of the transaction confirmation time of the blockchain platform.

Chapter 4

AegisDB: Scalable Blockchain Database with Secure Decentralised Load Balancing

Blockchain databases face performance limitations due to the scalability issues of blockchain technology. While existing solutions such as sharding can help mitigate these scalability challenges, they often compromise the security guarantee provided by blockchain. In this chapter, we investigate a novel method to improve scalability for blockchain databases, while preserving the security advantages inherited from blockchain technology. To achieve this, we reintroduce the classical concept of load balancing to blockchain database systems. We present the Verifiable Random Round Robin (V3R) load-balancing algorithm, and propose a blockchain database system, AegisDB, based on this algorithm. In AegisDB, workloads are distributed to the server nodes using V3R, ensuring scalability while preserving a similar level of security guarantees as unsharded blockchains.

4.1 Introduction

Blockchain technology has rapidly emerged as a revolutionary concept in recent years, with its potential to transform various industries such as finance, healthcare, supply chain management, and more. It was initially designed as a distributed and decentralised database specifically for storing cryptocurrency transactions [72]. Compared to traditional distributed database systems, blockchain provides unique features such as decentralisation, tamper-proof transaction history, and Byzantine fault tolerance. These properties are not only crucial for financial databases but are also desirable for more general use cases, particularly in a trustless shared database setting among multiple independent organisations.

Despite its potential benefits, blockchain faces some critical challenges when used as a general-purpose distributed database system. The most significant problem is limited performance. Taking the Bitcoin blockchain as an example, its current throughput is approximately 7 transactions per second [52], which is significantly lower than the typical requirements of modern distributed database systems. This limitation arises from the sequential nature of the blockchain data structure, where each transaction must wait in a queue before being included in a mined block that is subsequently attached to the chain. The overall throughput is determined by the average number of blocks mined per second multiplied by the average number of transactions contained within a block. Unlike traditional distributed database systems, where the throughput is proportional to the number of server nodes, in a blockchain system, introducing additional nodes does not increase throughput because every node simply replicates all on-chain operations. This is known as the blockchain scalability problem [52].

One of the developing solutions to the blockchain scalability problem is sharding [52] [91] [101]. In this approach, the blockchain network is divided into smaller shards, with server nodes and clients assigned to these shards to run their own blockchain. While the throughput can scale proportionally with the number of shards, it comes with some trade-offs. The security bar is significantly lowered from requiring 51% mining power in the entire network to that of a single shard [111]. Additionally, since a user's assigned shard is easy to determine, it is quite practical for attackers to launch targeted attacks. Intuitively, another direction is to execute read-only operations off-chain, effectively eliminating the throughput limitation of the blockchain for search queries. This approach is also essential for ensuring a reasonable response time for search queries and has already been considered by existing blockchain database systems [66] [15]. Assuming the targeted use cases are read-heavy, this approach can improve the overall throughput significantly.

However, while this approach eliminates the limitations of the blockchain, it also compromises the security guarantees provided by the blockchain. Since the read operation is executed off-chain, the results do not reflect the consensus of the majority. Consequently, if an attacker intends to deceive a client with spurious query results, they only need to corrupt a single node that provides the query response, instead of the majority of miners. In the trustless setting, the presence of malicious insiders exacerbates this problem further. To ensure confidence in the results, existing studies [66] [15] have proposed various verification methods

for query results. However, these verification processes all involve some on-chain operations, which again impacts the scalability. Due to this reason, achieving scalability and security simultaneously remains a challenging task, and to the best of our knowledge, none of the previous research can achieve both.

In this research, we aim to construct a general-purpose blockchain database for the trustless shared database setting, addressing the aforementioned problems. We present AegisDB, a blockchain database built on top of Algorand blockchain [93]. AegisDB utilises the blockchain as a reliable storage for the database transactions, upon which a shared state database is established among all the nodes. It also provides a flexible database interface to the end-users. We introduce a secure decentralised load balancing mechanism, Verifiable Randomised Round Robin (V3R), with a Verifiable Random Function [51] (VRF)-based selection algorithm replacing the central load balancer. With this approach, a read-only query is executed by a selected portion of nodes, which can be adjusted as needed, rather than being executed on-chain by all nodes. This allows AegisDB to scale out for read-heavy workloads while maintaining a similar level of security for these queries as if they were executed on-chain. Users are not required to perform additional verifications or other on-chain steps, significantly reducing computation and communication overheads.

Our Contributions

- We present AegisDB, a general-purpose blockchain database designed for the trustless shared database setting. AegisDB takes an off-chain non-verification approach for queries that optimises response time and provides scalability, without compromising security. Compared to verification-based approaches, our system eliminates the need for user verification, thereby reducing the computation, communication, and storage overhead associated with verification data.

- We provide formal definitions for a novel abstraction in the database layer called the Decentralised Append-Only Transaction Log (DATL). DATL abstracts any blockchain into an append-only, versioned database transaction log, serving as an essential storage component for modern transaction-based database systems.

- We introduce V3R, a decentralised load balancing algorithm based on VRF. With this algorithm, each query is executed off-chain by a selected group

of server nodes. It is the key to provide a similar level of security as if the query were executed on-chain while still being able to scale out.

- We implement a prototype of AegisDB. The experiments show that the scalability, server computation overheadn and user computation overhead are improved compared to the existing systems.

4.2 Related Work

4.2.1 Blockchain Database

The closest related work we have identified are [66] and [15]. Both studies consider a shared blockchain database setting involving untrusted parties. In [66], query result integrity is ensured through on-chain verifications. They propose what they call an offline verification approach where on-chain verifications can be batched and postponed to improve query response time. To achieve scalability, the study implements sharding by running multiple instances of different blockchain systems as shards. However, as discussed earlier, incorporating on-chain operations in the verification process comes at the expense of overall throughput. Furthermore, the employed sharding approach significantly reduces the security threshold and is susceptible to targeted attacks.

In [15], an Authenticated Data Structure (ADS) is stored on the blockchain to provide verifiability. However, the verification process is implemented using smart contracts, which results in on-chain operations and faces the same scalability challenges. Moreover, the utilisation of ADS introduces substantial on-chain storage overhead and imposes additional computations for both servers and clients.

There are other works exploring the concept of blockchain databases. For example, [106] is a commercial blockchain database system built using MongoDB and the Tendermint blockchain. However, this system does not address the off-chain read security concerns like the previous two systems. It is worth noting that all three of these systems only support key-value-based data models. In contrast, [3] explores a relational blockchain database that enables more expressive queries. However, it does not consider the off-chain read security problem either.

4.2.2 Verifiable Database

A verifiable database is a type of database that provides mechanisms for ensuring the integrity and authenticity of its stored data [28] [98] [58] [13] [81] [40] [113] [19] [87]. It allows users to verify the correctness of the data stored in the database without relying solely on trust in the database administrator or the underlying infrastructure. By incorporating verifiability into the database design, users can independently verify the integrity and authenticity of the stored data, enhancing trust and reducing the reliance on centralized authorities. Our system provides similar security guarantees, namely integrity of the outsourced data, but through a different security model.

4.2.3 Blockchain Sharding

Blockchain sharding is a line of research [91] [101] [46] [111] trying to improve the scalability and performance of blockchain systems. It involves dividing the blockchain network into multiple smaller groups called shards, each capable of processing a subset of transactions or smart contracts. By partitioning the network into shards, the overall processing capacity of the blockchain can be increased, allowing for higher transaction throughput and reduced latency. Our system uses off-chain techniques to improve scalability, which does not lower the security bar like most of the sharding schemes.

4.3 Preliminaries

4.3.1 Blockchain

A blockchain is a digital ledger of transactions that is maintained across a decentralised network of computers. It is designed to be secure, transparent, and tamper-proof, making it ideal for applications where trust and security are paramount. At its core, a blockchain is a series of blocks that are linked together in chronological order. Each block contains a list of transactions that have been validated and confirmed by a network of nodes. Once a block is added to the chain, it cannot be altered or deleted, ensuring the integrity of the ledger. The blockchain network is a distributed network of nodes that are connected to each other through a peer-to-peer protocol. Each node in the network maintains a copy of the blockchain ledger and participates in the validation and verification of transactions. A consensus is

the process by which nodes in the network agree on the current state of the ledger. In a blockchain network, consensus is achieved through a consensus algorithm that determines which blocks are added to the blockchain and which are rejected.

4.3.2 Verifiable Random Functions

Verifiable random functions (VRFs) are cryptographic tools that allow for the generation of random values that can be verified by other parties without revealing the underlying source of randomness. VRFs can be used in a variety of applications, such as in the generation of random numbers for use in secure protocols or in the selection of committee members for decentralised systems. In these applications, the verifiability of the VRF output ensures that the selection process is fair and unpredictable, without the need for a trusted third party. VRFs operate by taking a secret key and an input value, and producing an output value that is a function of both the key and input. The output value appears to be completely random, but can be verified by anyone with access to the public key. Additionally, the output value cannot be predicted or influenced by anyone who does not have access to the secret key. VRF plays an important role in the consensus protocol of Algorand blockchain.

4.4 System Model

4.4.1 System Overview

Our proposed system considers a trustless shared database use case. There are many practical scenarios for this use case, and one example could be the following: Suppose that a group of organisations collaborate closely on a project. Each organisation has important data that needs to be shared to facilitate collaboration or management. The users could either be organisation insiders or external entities such as customers, government department, etc. However, each organisation operates independently and may be in competition with one another from time to time. In some cases, they may be incentivised to behave maliciously (e.g. by providing deceptive information) to gain an advantage. Therefore, both scalability and security are crucial properties of such systems.

The above system includes two main entities: server nodes and users. Server nodes are computational units owned by different organisations. They are connected

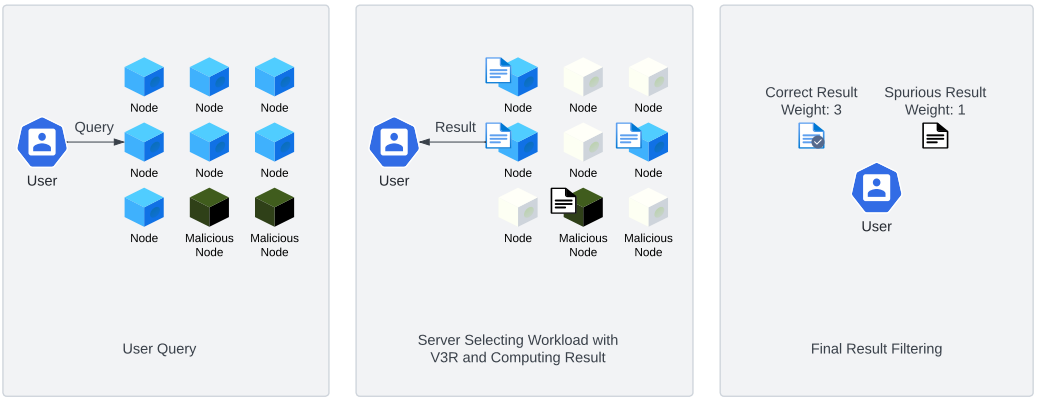


FIGURE 4.1: Query Processing in AegisDB

by the blockchain network and provide database services as a whole, similar to a traditional distributed database cloud. As we mentioned previously, the users may come from one of the organisations, or can be an external entity. They can access the database service by submitting transactions to the blockchain network, through a few server nodes that they may or may not trust. A server node comprises two main components: a blockchain layer and a database layer. The blockchain layer provides reliable storage and exposes its interface to the upper database layer. We abstract this layer as a Decentralised Append-Only Transaction Log (DATL). The DATL is responsible to store an append-only, incremental versioned transaction log for the database layer and synchronising it with other nodes in the blockchain network via its consensus protocol. The database layer hosts a state database, each state of which is reached by executing the database events in the incremental transaction log provide by the blockchain layer. A flexible relational or No-SQL interface is provided by this layer to the end-users.

How a database transaction is processed in AegisDB depends on the type of operations included in the transaction. For a write transaction, the operations are first written into the DATL (i.e., committed to the blockchain). Then the nodes compute and update the current state of the state database accordingly. For a read transaction, each server node first runs the V3R algorithm (section 4.5.3) to determine if it is selected to answer this query. If the node is selected, it computes the query results and sends the results back to the user together with the weight and the VRF proof computed in V3R. After receiving the results, if there is any divergence in the results, the user sums up the weights of the results and accepts the result with the greatest weight. This procedure is illustrated in Figure 4.1.

4.4.2 Threat Model

The main threat that we consider is a single server node or a group of nodes that behave maliciously. Practical reasons can be that the nodes may be temporarily under the control of unauthorised intruders, or there may be insiders in the owner organisation who just decide to behave maliciously. In either case, we simply call the adversary an attacker. The attacker can refuse to process any transactions, or tamper with the query results to their benefit, such as providing false information. This attack can either be targeted or untargeted. A targeted attack only affects some specific user or queries related to some specific data, and vice versa.

In the Algorand blockchain, the nodes are identified by the public keys of their owners. Their voting power is weighted by the amount of Algorand tokens associated with their public key, which is referred to as "stakes". Hence the consensus type, Proof of Stake. A typical BFT-type consensus algorithm, including Algorand, assumes that no more than 1/3 of stakeholders are malicious, weighted by their stake. In this research, we follow this assumption and assume that an attacker cannot control more than 1/3 of the total stake in the blockchain network. Based on the scenario we discussed previously, we assume the group runs their own permissioned blockchain. While in a public blockchain the nodes can go online and offline arbitrarily, in a permissioned blockchain the nodes are usually considered dedicated servers and are expected to be always online. Since new servers also cannot join without permissions, the total number of nodes in the network is assumed to be relatively stable. For the attackers, we assume that they run in probabilistic polynomial time and cannot break the VRF efficiently.

4.5 System Design

4.5.1 Blockchain Layer

The blockchain layer in our system works as a decentralised reliable storage for the upper database layer. Vertically, it receives the database transactions from the database layer, and stores them in a versioned append-only ledger. The database layer can query the historical versions of the ledger, and get a log of database events between any two versions. Horizontally, all the server nodes working in this layer forms a blockchain network. They synchronise their own ledgers (i.e. local views) with other nodes through the blockchain consensus, such that all the nodes

agree on a total order of the database transactions. To facilitate these functions, we introduce a new data structure in this layer, the Decentralized Append-Only Transaction Log (DATL), which we will present next.

A Decentralised Append-Only Transaction Log (DATL) provides an append-only, versioned database transaction log. The append-only transaction log is the essential storage component of modern database systems that enables features such as event audition and rollback to history versions. The DATL synchronises the transaction log through a blockchain network, which provides desirable features such as decentralisation, immutability and total order of transactions guaranteed by the blockchain consensus algorithm. Our abstraction can be implemented with any existing blockchain platforms, but in this research we choose Algorand blockchain because of its forkless BFT consensus and relatively short transaction confirmation time. Formally, the DATL consists of the following 5 protocols:

- $\mathtt{L} \leftarrow \textbf{Init}(\lambda)$ is a protocol run by the server nodes to initialise the data structure. It takes as input a public parameter λ and outputs an empty Decentralised Append-Only Transaction Log $\mathtt{L}$.
- $rcpt \leftarrow \textbf{Submit}(\mathtt{L}, tx)$ is a protocol run by the server nodes to submit database transactions to the DATL. It takes as input a database transaction tx and the log $\mathtt{L}$, and outputs a transaction receipt $rcpt$. Note that this transaction submission is an asynchronous process, which later on requires the **Sync** protocol to include the all the submitted transactions into $\mathtt{L}$.
- $\mathtt{L}' \leftarrow \textbf{Sync}(\mathtt{L})$ is a protocol run by the server nodes to synchronise their local views of the DATL. It takes as input the log $\mathtt{L}$, and outputs an updated version $\mathtt{L}'$ which all the nodes have reached consensus on.
- $ver \leftarrow \textbf{Version}(\mathtt{L})$ is a protocol run by the server nodes to get the current version number of the DATL. It takes as input the log $\mathtt{L}$, and outputs its current version number ver.
- $\mathtt{T} \leftarrow \textbf{Diff}(\mathtt{L}, ver_a, ver_b)$ is a protocol run by the server nodes to compare the differences between two historical versions of the DATL. It takes as input the log $\mathtt{L}$ and two version numbers ver_a and ver_b, and outputs a set of transactions $\mathtt{T}$ that is in the version ver_b but not in ver_a of the DATL.

A database transaction consists of a sequence of database operations (e.g. SQL statements in SQL-based database), and is the atomic unit used in this layer to constitute the ledger. The database transactions must be broadcasted to all the server nodes throughout the blockchain network and eventually the total order of them must be determined by the blockchain consensus. To implement DATL with a blockchain, the database transactions can be carried in the blockchain transactions. Typically, blockchain transactions can carry arbitrary data, such as the "note" field in the Algorand transactions. The content of database transactions can be written in it and hence be directly included in the blockchain. This approach is straightforward and no extra communication is needed. However, when the size of the database transaction exceeds the limit that a blockchain transaction can carry, the database transaction needs to be divided into smaller ones. In practice, the atomic property of the divided transactions may not always hold, depending on the database design and the size limit of the underlying blockchain (e.g. 1kB for Algorand). To implement the DATL using any blockchain system while preserving the atomic property, the extra (d, m, n, t) headers should be added to the blockchain transactions, where d is the database, m is the global transaction number of the transaction in that database, n is the slice number of the transaction and t is the total number of slices for the transaction. The DATL should execute the transactions according to the atomic rule and the global order of the transactions should be preserved.

4.5.2 Database Layer

The database layer provides either a relational or a No-SQL database interface to access the data stored in the blockchain layer. It stores a state database, through which all the user requests are processed. The database layer is responsible for keeping the state database up-to-date. When the state of the underlying blockchain layer changes (e.g. a new block is mined), the database layer receives a signal from it. Then the database layer requests the current version number of the blockchain layer and compares it with the version number of the state database. If the version of the state database is behind, the database layer can query the difference between the two versions, and get a transaction set `T` to be executed. By executing the transactions in `T`, the state database updates to the latest version. Algorithm 5 shows this process.

Algorithm 5: Update State Database

Input : a state database D
a DATL instance L
Output: a new state of D

```
1 v_L ← Version(L)
2 if v_L > D.version then
3     T ← Diff(L, D.version, v_L)
4     D.Execute(T)
5     D.version ← v_L
6 end
```

A user interacts with a server node via database transactions. When a database transaction is received by the database layer, it is first broadcasted to the peer nodes. Afterwards, the transaction is processed differently depending on whether it is a write transaction or a read transaction. If it is a write transaction, it is submitted to the blockchain layer for on-chain storage. If it is a read transaction, the database layer runs V3R to check if itself is the selected node to answer this query. If it is selected, it runs the query in the state database and return the results to the user. If it is not selected, no further processing is performed.

4.5.3 Secure Load Balancing

In the traditional centralised distributed database scenario, the query loads are distributed to server nodes by a load balancer, such that each server node only processes a portion of the total loads. This approach can significantly improve the scalability, but it is also quite obvious that this approach cannot be used in the scenario presented earlier in this chapter. First, the load balancer would be an ideal target for attackers, and even worse, considering such a powerful node may become malicious in a trustless scenario. Furthermore, to be secure against targeted attacks, the selected node for processing next queries should not be predictable, which is not the case for most of the centralised load balancing algorithms, either static or dynamic.

To address the above problems, we introduce a secure decentralised load balancing algorithm, Verifiable Randomised Round Robin (V3R), which does not require a centralised load balancer. Similar to the committee selection in Algorand, V3R uses VRF to select nodes based on their identifications (i.e. public key) and weighted by their stakes (i.e. the amount of Algorand tokens). The probability that each public key is selected is proportional to its stake.

Formally, let (pk, sk) be the pair of public and secret keys to identify some node i. The node i has a stake of s in its account associated with its public key pk, and the total stakes of all nodes are S. The system parameter τ is a threshold for the selection. Upon receiving a read transaction with a hash digest d, The node i computes the VRF outputs using the algorithm 6. This selection is weighted by j. Consider that each unit of node i's stake is treated as a sub-identity of i, and j indicates exactly how many of these sub-identities are chosen by the algorithm, which follows binomial distribution $B(k; s, \frac{\tau}{S})$. After the client receives all the weighted results, if any result has a summed weight greater than or equal to $\frac{\tau}{2}$, it should be accepted by the client as the correct result.

Algorithm 6: V3R

Input : secret key sk
transaction digest d
selection threshold τ
stake of the node s
total stake S
Output: VRF output $hash$
VRF proof π
weight j

$(hash, \pi) \leftarrow \mathbf{VRF}_{sk}(d)$
$j \leftarrow 0$
while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; s, \frac{\tau}{S}), \sum_{k=0}^{j+1} B(k; s, \frac{\tau}{S})\right)$ **do**
    j++
end
return $\langle hash, \pi, j \rangle$

Two-Tier Load Balancing Since server nodes are identified by their public keys, there could be cases where two or more nodes share the same public key. This situation is natural, as nodes established by the same organisation may simply use the same identity. While this doesn't impact the security assumptions, it adds extra flexibility to the system. All nodes using the same identity (i.e., from the same organisation) can be managed with a centralised method within that organisation, thus allowing an extra layer of centralised load balancing mechanism to be added. The organisation's centralised load balancer can then act as a representative in the network to participate in tier-1 load distribution using V3R.

Incentives Although the system is designed as a permissioned blockchain, connecting a consortium of organisations, there could be cases where the database service is open to entities outside the consortium. Servers may require incentives to provide the service, and transaction fees may apply. In such cases, smart contracts can be deployed to assist. Each query transaction can be sent with a

condition that, upon providing the correct VRF proof corresponding to the query, the transaction fee is transferred to the prover. Each selected honest server can upload their VRF proof to automatically receive payment after answering the query. It's important to note that a selected malicious server can also get paid without answering the query. This malicious behavior is expected and does not impact the security guarantee that users can obtain correct results.

4.6 Security Analysis

Assuming the attacker has a stake of s, and the total stakes in the system are S. The system threshold is τ and the user accepts the result when there is a weighted result for which the weight is greater than or equal to $\frac{\tau}{2}$. The attacker's goal is to trick the user to accept a spurious result sent from the attacker. In order for the attack to be successful, the attacker must be able to generate VRF proofs for a spurious result with its weight greater than or equal to $\frac{\tau}{2}$. Based on the assumptions, the attacker cannot break VRF efficiently, and the attacker succeeds when more than $\frac{\tau}{2}$ weighted attacker are selected by V3R to answer the query. So the probability that the attack succeeds is

$$\Pr(Success) = \begin{cases} 0, & \text{if } s < \frac{\tau}{2} \\ \sum_{k=\frac{\tau}{2}}^{s} B(k; s, \frac{\tau}{S}), & \text{otherwise} \end{cases}$$

When $s \geq \frac{\tau}{2}$,

$$\begin{aligned} \Pr(Success) =& \sum_{k=\frac{\tau}{2}}^{s} B(k; s, \frac{\tau}{S}) \\ =& \sum_{k=0}^{\frac{\tau}{2}} B(k; s, \frac{\tau}{S}) \\ =& F(\frac{\tau}{2}; s, \frac{\tau}{S}) \end{aligned}$$

According to Hoeffding's inequality, the above probability has a bound

$$\Pr(Success) = F(\frac{\tau}{2}; s, \frac{\tau}{S}) \leq \exp(-2s\left(\frac{\tau}{S} - \frac{\tau}{2s}\right)^2)$$

Based on the sytem assumption, we have $s < \frac{S}{3}$.

$$\begin{aligned}\Pr(Success) &< \exp(-\frac{\tau^2}{6S}) \\ &< neg(\tau)\end{aligned}$$

Finally, the probability that the attacker can be successful is a negligible function of τ.

4.7 Implementation and Experimental Results

We implement a prototype of AegisDB in Python, where MySQL is selected as the database component and Algorand is selected as the blockchain component. The choices are made because of their performance and functionalities. In future, any alternative blockchain can be used to implement the DATL as long as its transactions can carry custom data. Switching to alternative database components is even easier, requiring only the implementation of a new connector. We establish a local network with 16 AegisDB nodes. To simulate the geographic distribution, we add 100 ms latencies to the packets. The blockchain is a private Algorand chain using a custom genesis block, but has similar characteristics as the Algorand public blockchain. Each node is a Linux virtual machine assigned with 2 virtual CPU cores and 2GB memory, and they are connected via a 100 Mb/s virtual ethernet. We use TPC-H [7] with a scale factor of 1 as the workload, which has 8 tables including around 8 million records and 22 different types of SQL queries.

4.7.1 General Performance

In this experiment, we measure the throughput of the system by Transaction Per Second (TPS). We only use Q1 of TPC-H, which is a less computation-intensive query, as testing local database processing time is not the main purpose of this experiment. Figure 4.2 shows the performance of the system with different τ

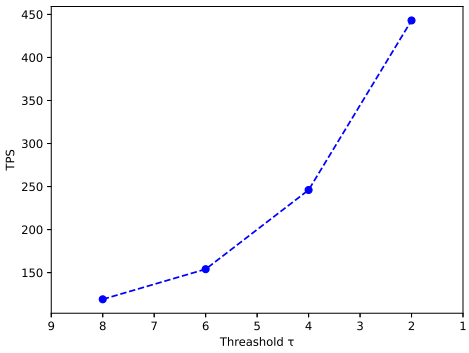


FIGURE 4.2: Query Throughput

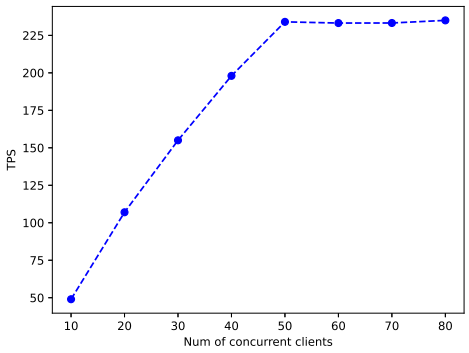


FIGURE 4.3: On-Chain Throughput

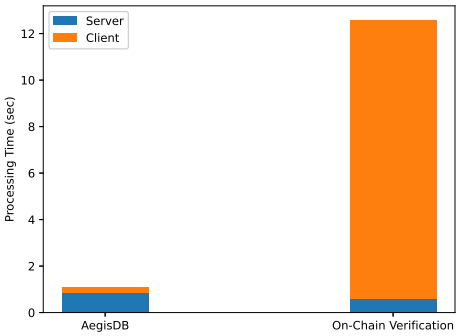


FIGURE 4.4: Processing Time

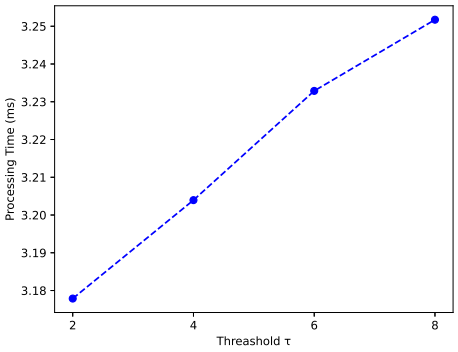


FIGURE 4.5: Client Processing Time

values. With a lower τ value, each query is executed by fewer nodes, and the system has a higher overall throughput. However, with a lower τ value, the security is weakened as we discussed in section 4.6. In practice, we can introduce more nodes to increase the overall throughput of the system, while setting the τ value at a reasonably high value to keep the system secure. This experiment shows the scalability of AegisDB. We also test the on-chain write performance of AegisDB. We connect multiple clients to the server nodes sending write transactions. Each of the client will send an update statement modifying the "REGION" table in the TPC-H database. Figure 4.3 shows that with the number of clients increasing, the on-chain throughput eventually caps at around 230 TPS. Although the overall on-chain write throughput of AegisDB is lower than its off-chain query throughput, it is greater than the systems based on Ethereum.

4.7.2 Query Processing Time

This experiment shows the query processing time of AegisDB, namely how long it takes for a user to get results for a search query. For this experiment, we use the on-chain processing scheme of [66] as a baseline to show the benefit of off-chain processing. For each query, there are two parts of processing, the server part and the client part. The latter is necessary for the user to be sure about the correctness of the results. In AegisDB, the server part includes computing the search results and running the V3R algorithm. The client part includes hashing and comparing the hash digest of the results received from multiple server nodes. In the on-chain processing scheme, the server needs no extra work other than computing the search results, but the client needs to do on-chain verification of the results, which is time-consuming due to the long transaction confirmation time. Figure 4.4 shows this comparison, where random selections of all 22 TPC-H queries are used and the average processing time is measured. Figure 4.5 shows the change of client processing time when the threshold τ increases. It is essentially a linear correlation since when τ increases the number of results received by the client increases proportionally. We observe that the client processing time is negligible since they are mainly hashing operations which are efficient.

4.8 Conclusion

In this research, we present AegisDB, a novel approach to solving the scalability problem of blockchain database systems. Our scheme achieves scalability without compromising on security, unlike normal sharding schemes. Notably, read queries do not need to go through transaction confirmation, resulting in response times comparable to non-blockchain-based distributed database systems. Moving forward, our goal is to achieve similar response times for write operations as well. This direction presents an interesting but challenging avenue of study that significantly contributes to the field of blockchain databases.

Chapter 5

SEARCHAIN: Searchable Encryption As Rewarded-useful-work on Blockchain

In this chapter, we introduce SEARCHAIN, a novel blockchain-based SSE system aimed at addressing the incentives for blockchain SSE systems. We propose a novel committee selection algorithm for the system to select verifier committee members, ensuring their voting powers are proportional to the number of active SSE services they manage. Verifiers are required to perform result verification work for SSE services to participate in mining. With this approach, the committee selection algorithm provides both security and incentives for the system. We implement a prototype of SEARCHAIN and deploy the system to a local test network. The experimental results demonstrate the feasibility and efficiency of SEARCHAIN.

5.1 Introduction

The emergence of blockchain technology has brought disruptive innovations to various sectors [84, 99], including cloud storage, which has seen rapid development in recent years. Blockchain-based cloud storage systems like Filecoin [65] and Storj [116] offer users an alternative to traditional centralised providers such as Google and Amazon. These platforms also provide opportunities for individual storage providers to earn rewards for outsourced storage. Users can enhance data confidentiality by encrypting their data before uploading it, but current systems lack the capability to search encrypted file content, thereby limiting usability. Symmetric Searchable Encryption (SSE) technology [69, 82, 114] can address this limitation by enabling keyword searching on encrypted files. However, maintaining a search index in this scenario, whether by the user or a centralised server, is undesirable.

To bridge the gap, the searchable encryption service can also be outsourced to the blockchain network, mirroring the storage service. Nevertheless, this introduces a new obstacle not found in traditional client-server structured searchable encryption systems, which is how to ensure security against both dishonest service providers and dishonest users. This becomes especially pertinent in a blockchain scenario where neither party can be inherently trusted. In a typical blockchain setup, nodes are often regarded as rational actors inclined toward actions that yield profit. In the context of searchable encryption, these nodes might opt to delete user data or provide incorrect search results to conserve storage and computational resources. Conversely, users could attempt to evade search fees by disputing the correctness of the search results. Therefore, maintaining fairness between users and service nodes is essential for the system to be practical.

There are also efficiency concerns when the system is built upon a blockchain. The primary requirement is to ensure efficient on-chain storage costs for the system. Certain prior investigations [34, 79] adopt an approach where the SSE encrypted index is stored directly on the blockchain. However, this approach presents significant storage scalability challenges, given that the entire blockchain data is replicated across every full node in the network and that the encrypted index can be substantial for each user. It turns out that this efficiency requirement also encounters some trade-offs with the fairness concern above. When everything is on-chain, fairness can be inherently guaranteed by the blockchain consensus, assuming the honest majority. To mitigate on-chain storage costs, the encrypted index can be stored off-chain, with only metadata committed to the blockchain. In this case, guaranteeing fairness becomes much more challenging. For example, even if the service provider can demonstrate generating accurate results aligned with blockchain metadata, it doesn't guarantee transmission to the user.

A potential solution to this problem could involve a specialised group of verifier nodes. These nodes would function as a trusted party to resolve disputes regarding fairness between users and service providers. When a user disputes that they didn't receive the correct results, these verifier nodes can recompute the search results for the user and/or adjudicate the correctness of the results to resolve the dispute. However, two problems in this model, overlooked by previous works, remain challenging. Firstly, the incentives for the verifiers are not clear. While service providers receive service fees, there are no rewards for the verifiers' efforts. Given the assumption of rationality for the verifiers, incentivising their work becomes

imperative. Secondly, collusion between malicious verifiers and service providers needs consideration. The design of verifier selection and the adjudication process to uphold fairness pose challenging tasks.

This chapter presents SEARCHAIN, a blockchain-based SSE system designed to ensure service fairness between users and service providers. We propose a novel committee selection algorithm to pair with BFT-type consensus, incorporating SSE result verification as part of the blockchain mining process. The leader committee in SEARCHAIN is called verifiers, and their elected voting power is proportional to the number of active SSE services they manage. In each epoch, a verifier participates in the mining process by verifying the correctness of other SSE services. Block rewards serve as incentives for the contributions of verifiers. SEARCHAIN can achieve service fairness under strong collusion assumptions (e.g. all malicious verifiers colluding with a malicious service provider). Furthermore, SEARCHAIN enhances on-chain efficiency by conducting all SSE service-related data storage and computation off-chain.

Our Contributions. In this research, we make the following contributions:

- We introduce SEARCHAIN, a blockchain-based SSE system that simultaneously achieves service fairness and on-chain efficiency.
- We propose a novel committee selection algorithm where the voting powers of the committee members are proportional to the number of active SSE services they are serving, and SSE result verification is integrated into the mining process of the protocol. This makes both serving SSE service and verifying the results incentivised useful work.
- We perform a formal security analysis, which shows our proposed system achieves all the security goals.
- We implement a prototype of SEARCHAIN and deploy the system to a local test network. The experimental results demonstrate that the response time is significantly improved compared to previous blockchain-based solutions, and the consensus overhead introduced by the committee selection algorithm is acceptable.

5.2 Related Work

Blockchain-based SSE [16,26,34,71,79,92,105] is an emerging field that integrates blockchain technology with SSE, enabling the SSE service to operate on top of the blockchain system. In this domain, the SSE server is often perceived as potentially malicious, raising concerns about its willingness to provide accurate results after receiving the client's payment for the service. Consequently, blockchain technology is employed to ensure fairness. In some investigations [16, 26, 34, 71, 79], the encrypted index is stored on the blockchain, with service fees comprising smart contract fees and/or transaction fees remitted to the blockchain. While [26] and [34] solely employ the blockchain as cloud storage for the encrypted index, necessitating data owners to compute search results independently, [16, 71, 79] delegate this task to smart contracts, enabling on-chain computation of results. This approach solves the fairness problem as long as the honest majority assumption of blockchain holds, but the on-chain storage cost is impractical. Also, this approach usually has a query response time ranging from tens of seconds to minutes, depending on the underlying blockchain, which is prohibitively high compared to the traditional server-client structure. In some other studies [92, 105], the encrypted index and results are sent off-chain between the user and the service provider, with only metadata committed to the blockchain. In the event of a disagreement, the service provider transmits the encrypted index to a group of trusted blockchain nodes. These nodes then re-execute the query on-chain and return the correct results to the user. The assumption is made that these nodes are trusted and operate voluntarily, yet this assumption may not be valid due to the lack of clarity regarding their incentives.

Symmetric Searchable Encryption was originally introduced by Song et al. [10]. Subsequently, numerous endeavours [69, 76, 82, 114] have been dedicated to advancing the security and practicality of SSE schemes. Additional functionalities have also been proposed, such as boolean queries [27,33]. These schemes typically address scenarios involving an honest-but-curious server, which aims to breach client privacy but computes search results faithfully. To address the challenge of malicious servers, Verifiable SSE schemes were introduced [4,8,21,48,89], enabling verification of the correctness of search results. This line of work usually considers a different security model than SEARCHAIN; neither do they need to address the efficiency problems associated with the blockchain scenario.

5.3 Preliminaries

5.3.1 Publicly Verifiable SSE (PVSSE)

In a traditional verifiable SSE (VSSE), the results can only be verified by someone possessing the secret key of the data owner. In contrast, a publicly verifiable SSE (PVSSE) is a scheme in which the correctness of results can be verified by the public [44, 45, 86]. A PVSSE scheme comprises a set of algorithms (**Setup**, **GenToken**, **Search**, **Verify**) executed among three parties: a user, a service provider, and a verifier.

- $(K, vk, \texttt{EDB}) \leftarrow$ **Setup**$(\texttt{DB})$ is an algorithm run by the user to set up the encrypted database. It takes as input a document collection DB and outputs an encrypted database EDB, a secret key K and a verification key vk. Note that the vk can be empty depending on the scheme construction.
- $\tau \leftarrow$ **GenToken**(w, K) is an algorithm the user runs to generate a search token. It takes as input a keyword w and the user's secret key K and outputs a search token τ corresponding to w.
- $(R, \pi) \leftarrow$ **Search**$(\tau, \texttt{EDB})$ is an algorithm run by the service provider to search. It takes as input a search token τ and the index EDB. After running, it outputs the results R, from which the user can derive the queried identifiers $\texttt{DB}(w)$, and the corresponding proof π, to verify R.
- $\{ACCEPT, REJECT\} \leftarrow$ **Verify**(τ, R, π, vk) is an algorithm run by the verifier, which can be anyone who has the vk, to determine whether R is the correct answer to τ.

Soundness. A PVSSE scheme is sound if for all DB and w the probability that **Verify**(τ, R, π, vk) outputs $ACCEPT$ while $R \neq \texttt{DB}(w)$ is negligible, where $(K, vk, \texttt{EDB}) \leftarrow$ **Setup**$(\texttt{DB})$, $\tau \leftarrow$ **GenToken**(w, K), and $(R, \pi) \leftarrow$ **Search**$(\tau, \texttt{EDB})$.

5.3.2 Randomness Beacon

A blockchain randomness beacon is a blockchain-based system that generates a random value during each epoch. This value is designed to be unpredictable and requires time for computation. Once computed, the random value is permanently

recorded on the blockchain and cannot be altered [6]. In our system, this functionality is characterised as follows:

- $seed_t \leftarrow \textbf{GetRandom}(t)$ is an algorithm run by anyone accessing the blockchain. It takes as input an epoch number t and outputs the random value $seed_t$ of the randomness beacon for epoch t.

Unpredictability. Let t' be the latest epoch, the randomness beacon is unpredictable if for all $t > t'$ the probability that the value $seed_t$ can be computed by any PPT adversary is negligible.

5.3.3 Verifiable Random Functions

Verifiable random functions (VRFs) [51] are cryptographic tools that produce a random output so that anyone can verify that the output is genuinely random and generated from a specific input. VRFs are applicable in scenarios where secure generation of random numbers is essential. Their verifiability ensures fairness and unpredictability in these processes, eliminating the need to rely on a central authority. In our system, a VRF is abstracted as the following algorithms:

- $(r, \pi_r) \leftarrow \textbf{Eval}(sk, x)$ is an algorithm run by the prover to generate a random output from a given input. It takes as input a seed x and the prover's secret key sk, and outputs the evaluated random value r along with a proof π_r.
- $\{ACCEPT, REJECT\} \leftarrow \textbf{Verify}(pk, x, r, \pi_r)$ is an algorithm run by anyone to verify if r is the correct VRF output evaluated from x, with the proof π_r and the prover's public key pk.

Pseudorandomness. A VRF is pseudorandom if for all PPT adversaries the output r is indistinguishable from random values.

5.4 System Model

5.4.1 System Overview

Our proposed system consists of a permissionless blockchain in which members are divided into three primary roles based on their purposes and behaviours: service

providers, users, and verifiers. Service providers earn income by delivering SSE services. For simplicity, we model a service provider serving multiple services to multiple users as multiple individual service providers. Thus, each service provider handles at most one service. Users are individuals who participate in our system to utilise the SSE services, paying service fees to service providers for processing their SSE queries. Verifiers represent a specialised subset of service providers who, in addition to serving SSE queries, also verify the query results of other service providers.

The system consists primarily of the SSE service cycle and consensus. The SSE service cycle encompasses two primary processes: Setup and Search. During Setup, the user and the service provider sign a transaction to establish an SSE service. The service provider then sets up a publicly accessible cloud storage *bulletin* for storing and verifying search results. The *bulletin* stores a limited number of the most recent search results, while the expired results are removed to save storage cost and verification time. Once the service is set up, users can initiate queries through the Search process. Queries are conducted by the user sending a search transaction to the blockchain network. The service provider responds to the query by posting corresponding results on the *bulletin* associated with this service. Subsequently, the user can retrieve the results from the bulletin. We call a service provider with a running service within its duration an active service provider. During the Setup phase, the service provider is required to commit a fixed amount of stake to the setup transaction. This stake serves dual purposes: as collateral in case provider misbehaves in serving SSE queries and as a mining stake to prevent sybil attacks.

Consensus is elucidated by dividing the system's timeline into epochs. Within each epoch, the consensus process initiates with verifier selection. Every active service provider computes a pseudorandom value using the VRF and the current epoch's randomness seed. If this value falls below the selection threshold, the provider is chosen as a verifier. Subsequently, the verifier identifies a set of other service providers as their verification peers and validates the search results on the *bulletin* of said peers. Following verification, the verifier signs and broadcasts a *join_committee* message to join the committee for the current epoch. Each participating verifier verifies the validity of all other verifiers' *join_committee* messages, after which they identify all the valid verifier committee members and execute a BFT-type protocol within the committee to propose new blocks.

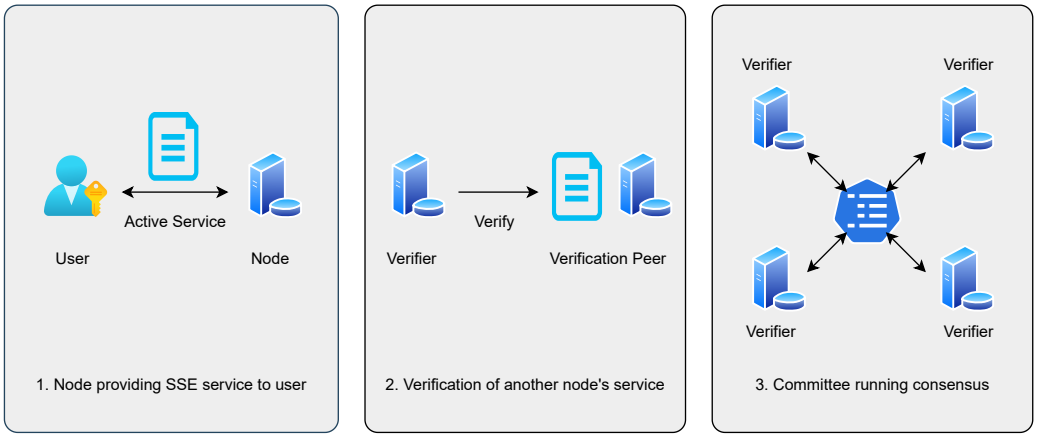


FIGURE 5.1: System Overview

5.4.2 Threat Model

In the SSE service cycle, a malicious service provider may ignore user search queries or provide false search results. This behavior can sometimes be profitable, as it saves on storage and computation costs, motivating a rational service provider to engage in such malicious actions. Although verifiers can verify search results to ensure service fairness, a dishonest verifier could collude with malicious service providers to conceal their malicious behaviour.

We assume that no entity within the network possesses the capability to breach cryptographic primitives, i.e. they are not able to solve computationally hard problems. We assume the network to be synchronous with some bounded latency. Furthermore, we assume that at any time, the number of malicious active service providers is less than 1/3 of the total number of active service providers in the network.

5.4.3 Design Goals

Our proposed design of the system needs to achieve the following properties:

- **Service Fairness**: Honest service providers should be rewarded with service fees, and honest users should obtain accurate search results for their queries. The dishonest behaviours identified in the threat model are detected and punished through the consensus of verifiers in the blockchain network.

- **Consensus Safety and Liveness**: The system retains the same safety and liveness properties as the BFT-type consensus adopted in the block proposal phase under the same network synchrony assumptions. We do not introduce any additional assumptions or compromise these guarantees.
- **On-Chain Storage and Computation Efficiency**: The storage and computation tasks for the SSE service data are performed off-chain by the respective responsive service provider rather than being conducted on-chain and duplicated across all nodes. This approach enhances the efficiency of on-chain storage and computation.

5.5 System Design

5.5.1 Service Setup

Service setup initiates the SSE service cycle, representing the initial process in the cycle. During this phase, the user and the service provider generate and transfer all necessary data required for subsequent query servicing. At the end of the process, they agree on a *Setup* transaction and send it into the blockchain network. The setup process comprises the following steps:

1. **SSE Setup**: In this step, a user **U** first runs **SSE.Setup** with some plaintext database $\texttt{DB}$ to get encrypted database $\texttt{EDB}$, verification key vk and secret key K. Formally, **U** runs $(K, vk, \texttt{EDB}) \leftarrow \textbf{SSE.Setup}(\texttt{DB})$. Then the user computes the digest $h_{\texttt{EDB}}$ of the $\texttt{EDB}$, by running $h_{\texttt{EDB}} \leftarrow \mathbf{H}(\texttt{EDB})$, where **H** is a cryptographic hash function. **U** also chooses a duration d for the service to be running.
2. **Sending Data**: In this step, the user **U** finds a service provider **S** in the marketplace that suits her needs. **U** sends the data ($\texttt{EDB}$, $h_{\texttt{EDB}}$, vk, d) to **S**.
3. **Transaction Construction**: Upon receiving the data from **U**, the service provider **S** stores $\texttt{EDB}$ and prepares a public bulletin for storing query results, for which the address is $addr$. **S** then calculates the service fee fee depending on the storage cost and service duration. Service provider **S** prepares a *Setup* transaction $(h_{\texttt{EDB}}, d, vk, addr, fee, s)$, where s is the mining stake required by the system for the service provider **S**. Then **S** sends the transaction to the user **U**.

4. **Transaction Signing**: Upon receiving the *Setup* transaction, the user U verifies everything is correct. If **U** accepts the transactions, then both **U** and **S** sign the transaction, formally $\langle h_{\texttt{EDB}}, d, vk, addr, fee, s\rangle_{\mathbf{U},\mathbf{S}}$, and send the transaction into the blockchain network. The verifier committee will verify the transaction for the current epoch and include it in the blockchain if it is valid. Once the transaction is on-chain, the service is active, and the service id is the transaction id of the *Setup* transaction. Note that if the service ends after the duration d with the service provider **S** not having any malicious behaviours, the stake s will be returned to **S**. Otherwise, the stake s will be burned.

5.5.2 Search

The search process is the primary and most frequently utilised function within an SSE service cycle. Once the service is active, users can commence querying the service provider using the search process. This process comprises the following steps:

1. **Token Generation**: To search in the encrypted database, an encrypted search token needs to be generated from the plaintext keyword. In this step, the user **U** first runs **SSE.GenToken** with the secret key K and a chosen keyword w to generate a search token τ. Formally, $\tau \leftarrow \textbf{SSE.GenToken}(w, K)$.

2. **Sending Transaction**: After generating the search token, **U** creates and signs a *Search* transaction $\langle id_S, \tau\rangle_{\mathbf{U}}$ and broadcasts it to the blockchain network, where id_S is the service id. The verifier committee for the current epoch will verify the transaction and then include it in the blockchain if it is valid.

3. **Server Response**: Upon seeing the *Search* transaction, the service provider **S** computes the search results by running **SSE.Search** with τ and the $\texttt{EDB}$. Formally, $(R, \pi) \leftarrow \textbf{SSE.Search}(\tau, \texttt{EDB})$, where R is the search results and π is the corresponding PVSSE proof. Afterward, **S** creates and signs a record $\langle id_q, R, \pi\rangle_{\mathbf{S}}$, and puts it in the bulletin associated with the service, where id_q is the transaction id for the *Search* transaction.

Upon retrieving the search results from the bulletin, the user **U** can verify the correctness by running **SSE.Verify**. However, this step is unnecessary as the

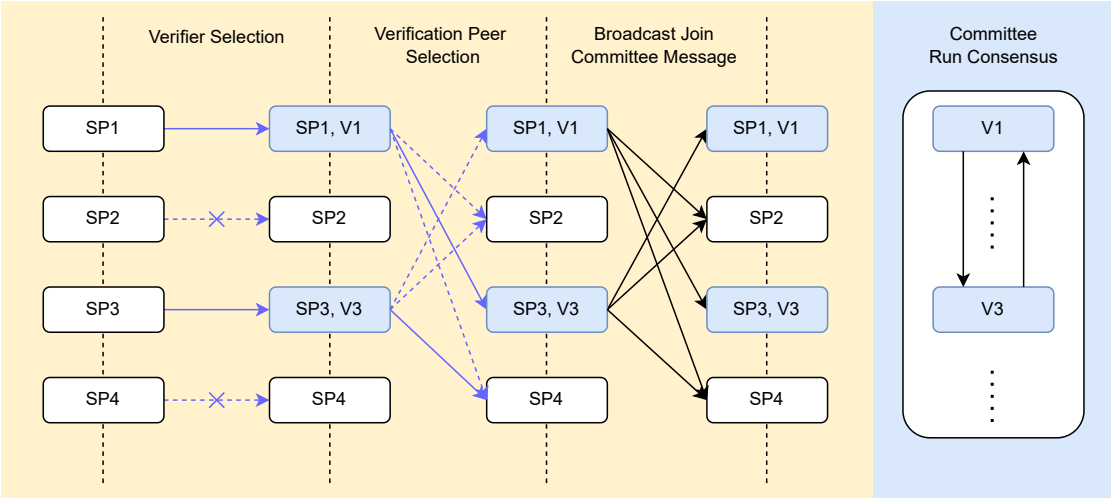


FIGURE 5.2: Committee Selection

verifier committee will perform it. If the design goals are achieved, fairness will be guaranteed by the system, and any rational service provider will not post incorrect results on the bulletin. We will discuss this in the following sections.

5.5.3 Committee Selection

Consensus, offering security and liveness, is particularly crucial in SEARCHAIN, as it also incentivises verifiers to verify the correctness of SSE search results. Consequently, the SEARCHAIN consensus also serves as the security foundation for the SSE service cycle.

At some epoch t, an active service provider $\mathbf{V}$ participates in the consensus by running the verifier selection algorithm. First, $\mathbf{V}$ computes a pseudorandom value a with $\mathbf{VRF.Eval}(sk, seed_t)$, where $\mathbf{VRF}$ is a Verifiable Random Function, sk is the secret key of $\mathbf{V}$, and $seed_t$ is the randomness beacon output for epoch t. If $\frac{a}{2^{hashlen}} < \delta_a$, where $hashlen$ is the hash length of the VRF and δ_a is a public system parameter called active threshold, then $\mathbf{V}$ is selected as a verifier and proceed with the following steps.

Second, $\mathbf{V}$ computes another pseudorandom value p_S with $\mathbf{VRF.Eval}(sk, seed_t||id_S)$, where $||$ is concatenation and id_S is the service id of an active service. Verifier $\mathbf{V}$ computes p_S values for all active services id_S. If $\frac{p_S}{2^{hashlen}} < \delta_p$, $\mathbf{V}$ adds id_S to a set VP and p_S to a set SS, where δ_p is another public system parameter called passive threshold. In this case, the set VP contains the id of the services that $\mathbf{V}$ needs to verify, and the service providers of these services are called the verification peers of $\mathbf{V}$.

In the third step, **V** verifies the bulletin of all verification peers. For each $id_S \in VP$, **V** verifies all the records on the bulletin against their corresponding *Search* transaction on the blockchain. If all the *Search* transactions are answered and the results pass the verification of **SSE.Verify**, **V** marks id_S as *Accept*, otherwise, **V** marks id_S as *Reject*. Next, **V** prepares and signs a *join_committee* message $\langle t, pk, a, \pi_a, VP, SS, OC \rangle_{\mathbf{V}}$, where the set OC contains $Accept, Reject$ outcomes for each verification peer. **V** broadcast the message to the blockchain network.

Algorithm 7: Verifier Selection

$seed_t \leftarrow$ **GetRandom**(t)
$a, \pi_a \leftarrow$ **VRF.Eval**$(sk, seed_t)$
if $\frac{a}{2^{hashlen}} < \delta_a$ **then**
  $VP \leftarrow \emptyset$
  $SS \leftarrow \emptyset$
  $OC \leftarrow \emptyset$
  for $\forall$ *active* id_S **do**
    $p_S, \pi_{p_S} \leftarrow$ **VRF.Eval**$(sk, seed_t || id_S)$
    if $\frac{p_S}{2^{hashlen}} < \delta_p$ **then**
      $VP \leftarrow VP \cup \{id_S\}$
      $SS \leftarrow SS \cup \{(p_S, \pi_{p_S})\}$
      $oc \leftarrow Accept$
      for $\forall$*Search transaction* $tx \in$ *service* id_S **do**
        get τ, R, π, vk from *blockchain* and *bulletin*
        if $R = \emptyset$ **then**
          $oc \leftarrow Reject$
          break
        end
        if SSE.Verify$(\tau, R, \pi, vk) = REJECT$ **then**
          $oc \leftarrow Reject$
          break
        end
      end
      $OC \leftarrow OC \cup \{oc\}$
    end
  end
  prepare message $(t, pk, a, \pi_a, VP, SS, OC)$
  sign $\langle t, pk, a, \pi_a, VP, SS, OC \rangle_{\mathbf{V}}$
  send $\langle t, pk, a, \pi_a, VP, SS, OC \rangle_{\mathbf{V}}$ to *blockchain*
end

Finally, all the verifiers, upon receiving the message above of other verifiers, verify the message. This includes verifying that the VRF values are indeed less than their corresponding thresholds, verifying all the bulletin records, and checking if the final outcome matches the *Accept*/*Reject* in the message. If all the verifications pass, the sender of the message is recognised as a valid verifier committee member. Afterwards, the verifier committee runs a BFT-type consensus protocol to propose blocks. The verification of the proposed blocks will include the outcomes of all

the verified services. If the outcome is *Reject*, the service will be terminated, and the stake in the *Setup* transaction will be burned. The block rewards for each created block will be shared by the verifier committee, weighted by the size of the verification peer set $|VP|$ of each verifier.

5.6 Security Analysis

In this section, we discuss the security properties of SEARCHAIN. To achieve service fairness, SEARCHAIN needs to guarantee that users can get the correct search results for all the queries. We state the following theorem:

Theorem 5.1. *Let id_q be the id of some on-chain query for some active service, $\boldsymbol{S}$ be the provider for this service, b be the bulletin for this service, $\langle id_q, R, \pi \rangle_{\boldsymbol{S}}$ be the bulletin record for id_q, and R^* be the correct results for id_q. Given that $\delta_a > 0$, $\delta_p > 0$, and PVSSE is sound, if $\exists id_q$ s.t. $\langle id_q, R, \pi \rangle_{\boldsymbol{S}} \notin b$ or $R \neq R^*$, then the probability that the service provider $\boldsymbol{S}$ will not lose their stake is negligible.*

Proof. Let the probability that the verification outcome $oc = Accept$ for $\mathbf{S}$ by an honest verifier be P_{acc}. If $\exists id_q$ s.t. $\langle id_q, R, \pi \rangle_{\mathbf{S}} \notin b$ or $R \neq R^*$, according to algorithm 7, $P_{acc} < neg(\lambda)$ if PVSSE is sound, where neg is a negligible function, and λ is the security parameter for PVSSE. Let n be the total number of active service providers. In each epoch, let the probability that $\mathbf{S}$ is selected as a verification peer by at least one honest verifier in the committee be P_{sel}, then $P_{sel} = \sum_{k=1}^{n} B(k; \frac{2}{3} \times n, \delta_a \times \delta_p)$, where B is a binomial distribution. The probability that $\mathbf{S}$ is not selected as a verification peer by any honest verifier in the committee, $1 - P_{sel}$, is a negligible function of n if $\delta_a \times \delta_p > 0$, denoted as $1 - P_{sel} < neg(n)$. Then in each epoch the probability that $\mathbf{S}$ will not lose their stake is $P = (P_{sel} \times P_{acc}) + (1 - P_{sel}) < neg(n, \lambda)$. If multiple epochs are considered, let the probability that $\mathbf{S}$ will not lose their stake after t epochs be P_t, then P_t is also a negligible function of t, denoted as $P_t < neg(t, n, \lambda)$. Therefore, the probability that the service provider $\mathbf{S}$ will not lose their stake is negligible. □

Given Thereom 5.1, and assuming the rationality of malicious service providers, we expect that within SEARCHAIN, malicious providers will not attempt to breach the service fairness property by ignoring user search queries or providing incorrect answers. This is discussed in more details in section 5.7.

The consensus safety of SEARCHAIN mainly depends on the verifier selection algorithm and the adopted BFT-type protocol. For the verifier selection algorithm, no service provider should be able to gain an advantage during the verifier selection process. This holds when the VRF is pseudorandom and the randomness beacon is unpredictable. Consequently, when the committee size $n \times \delta_a \times (1-(1-\delta_p)^n)$ is sufficiently large, it is straightforward to see that SEARCHAIN achieves consensus safety based on our assumptions.

5.7 Incentives

In this section, we discuss the incentives of nodes in SEARCHAIN. A node's utility in each epoch can be expressed as the verifier reward they receive minus the loss of their stake if they act as a malicious service provider. Formally, $U(\sigma_1, \sigma_2) = P_{\sigma_1} \times W_{\sigma_1} \times r - P_{\sigma_2} \times s$, where σ_1 is the strategy the node choose as an verifier, σ_2 is the strategy the node choose as a service provider, P_{σ_1} is the probability that the node joins the committee with strategy σ_1, $W_{\sigma_1} \times r$ is the weighted block reward share the node gets with strategy σ_1, P_{σ_2} is the probability that the node's stake gets burned with strategy σ_2, and s is the node's mining stake.

Let V be a selected verifier at some epoch t, and S be a verification peer selected by V using VRF. V can choose the following actions: 1. Verify S honestly. 2. Mark S as *Accept* regardless of the true verification outcome. 3. Exclude S from the VP list. According to our security model and the protocol design, the latter two choices will lower P_{σ_1} and/or W_{σ_1}, which will consequently decrease the utility. Therefore, V has a dominant strategy of always being honest as a verifier. As a result, S has a best response of always being honest as a service provider to avoid losing their stake. Furthermore, if we consider an entity that operates multiple service providers, the utility will be multiplied by the number of active services they run. This indicates that the utility for an entity in SEARCHAIN is determined by both the amount of useful work they contribute and the stake they hold.

5.8 Implementation and Experimental Evaluation

To evaluate the performance of SEARCHAIN, we implemented a prototype of the system and conducted several experiments using a test network. The implementation consists of two main components: the SSE service and the consensus

mechanism for blockchain. For the SSE service, we utilised the second construction proposed in [45], which builds upon any existing SSE scheme to make it publicly verifiable. The underlying SSE scheme we selected is based on [69]. We implemented the pseudorandom function (PRF) using HMAC-SHA256 and the pseudorandom permutation (PRP) with AES. Other cryptographic components include the hash function SHA-256, the symmetric encryption scheme AES-CBC with 256-bit keys, and the signature scheme RSA with 2048-bit keys. VRF was adopted from a Python implementation of ECVRF[1]. The result bulletin is implemented as a web service using the HTTP protocol on the service provider's side.

We built upon a Python implementation of PBFT for the consensus mechanism, integrating the service provider logic and the committee selection algorithm to construct the SEARCHAIN consensus. We adopted Drand[2] as the randomness beacon. We established a network comprising 16 nodes on a cloud VPS service, with each node running Ubuntu 20.04 on a virtual machine configured with 8 virtual cores of Intel Xeon CPUs and 8 GB of memory. The nodes are interconnected via a 100 Mb/s virtual Ethernet with emulated latency. Peer addresses are hardcoded in a configuration file, which eliminates the need for peer discovery for simplicity. All nodes function as active service providers, serving SSE queries to clients and participating in committee selection for consensus. We set the system parameters $\delta_a = 0.8$ and $\delta_p = 0.2$. Each node also operates a Drand node running in a Docker environment to collaboratively produce randomness. Additionally, the nodes expose a json-api for transaction submissions. Another VM is employed to run multiple instances of the client program for transaction submissions to multiple nodes.

5.8.1 SSE Service Performance

We conduct several experiments to evaluate the overall performance of the SSE service. First, we assess the performance of the setup protocol. The service setup process can be broken down into the following tasks. The user begins by building the encrypted database from a plaintext database, which involves generating the encrypted index for the base SSE scheme [69] and the public verification data for the PVSSE scheme [45]. Next, the EDB is transmitted to the service provider.

[1]https://github.com/nccgroup/draft-irtf-cfrg-vrf-06
[2]https://github.com/drand/drand

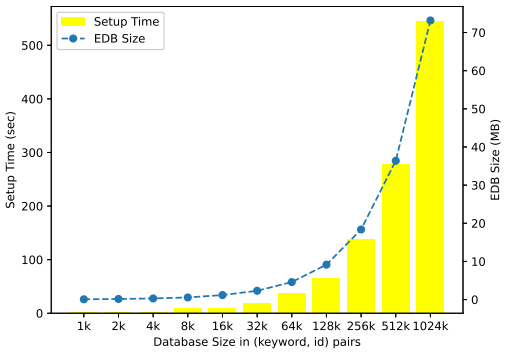


FIGURE 5.3: Service Setup

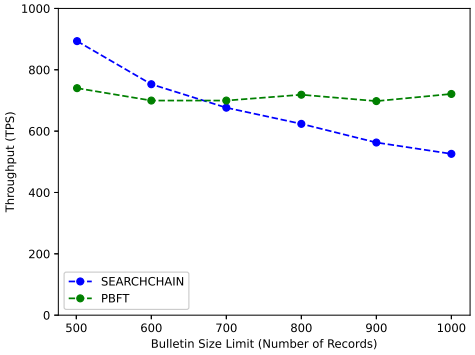


FIGURE 5.4: Consensus Overhead

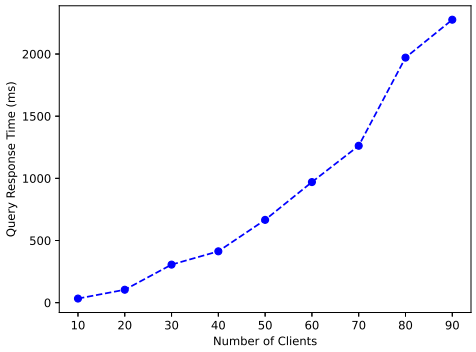


FIGURE 5.5: Query Response Time

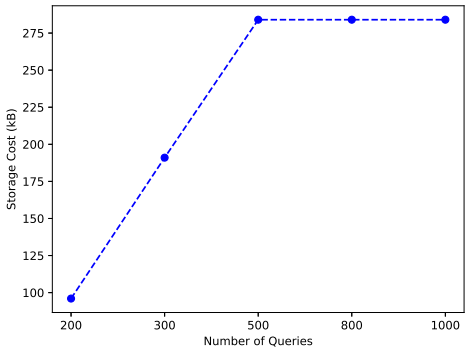


FIGURE 5.6: Service Provider Storage Cost

Finally, a setup transaction is sent to the blockchain, and once the transaction is confirmed, the service setup process is complete. We generated several encrypted databases with varying EDB sizes in terms of the number of (w, id) pairs. Figure 5.3 illustrates the average setup time and EDB size for each database, showing that both increase as the database size grows.

In the second experiment, we use multiple clients to send queries to a single service provider and measure the average response time. The EDB used in this experiment contains 16,000 keyword-id pairs, and the response time is measured from the moment the search transaction is submitted to the node until the result record is retrievable on the bulletin. Figure 5.5 shows that the average response time increases as the number of concurrent clients rises. Notably, the expected response time is significantly less than that observed in some previous studies using the on-chain approach discussed in Section 5.2, which can range from tens of seconds to minutes. We also evaluate the storage costs for the service provider. As shown in Figure 5.6, storage requirements increase as the number of processed queries

grows, primarily due to the larger size of the result bulletin. However, with a bulletin size limit set to 500 records, the storage cost eventually caps at around 280 kB.

5.8.2 Consensus Overhead

To evaluate the overhead introduced by the committee selection algorithm, we measure the overall throughput of SEARCHAIN under various bulletin size limit settings. We have clients continuously send queries until the system reaches full load, at which point we measure the system's throughput. To establish a baseline for comparison, we repeat the experiment using the same settings but turn off committee selection. In this baseline case, the consensus algorithm becomes bare PBFT, while the nodes still run and verify the SSE services. From figure 5.4, we observe that the throughput of the SEARCHAIN consensus decreases as the bulletin size limit increases. This decline is attributed to the greater number of SSE records that need to be verified during each round of committee selection, which prolongs the round time. Additionally, it is important to note that, under the same settings, the bare PBFT consensus exhibits a constant but slightly lower overall throughput compared to the theoretical throughput without the overhead. This difference arises because PBFT has a larger committee size.

5.9 Conclusion

This chapter explores the security and incentive aspects of blockchain-based SSE systems. We introduce SEARCHAIN, a system designed to achieve both service fairness and on-chain efficiency concurrently. In SEARCHAIN, the verification of SSE search results is performed by a verifier committee, selected using a verifier selection algorithm that employs VRF. To qualify as a verifier, a node must first establish itself as a service provider and operate an active SSE service. The voting power of verifiers is proportional to the number of SSE services they run. During the mining process, verifiers use VRF to select verification peers and verify their SSE search results, effectively incentivising both the provision of SSE services and the verification of results. Our formal security analysis demonstrates that SEARCHAIN achieves service fairness within our security model. Compared to the previous studies, SEARCHAIN provides an efficient off-chain solution for the service fairness problem that improves query response time and on-chain storage overhead. Furthermore, to the best of our knowledge, SEARCHAIN is the first

blockchain-based SSE system with clear incentives for SSE auditors to conduct the verification and to behave honestly.

Chapter 6

Future Directions

The development of practical and secure searchable encryption services on blockchain has made significant strides in this research area. However, there are still challenges and open questions for possible future studies. This chapter discusses potential future directions for advancing this research field.

- **Sharded encrypted blockchain database.** While this thesis employs an off-chain non-sharded approach to improve scalability, an alternative direction for future research is blockchain sharding. Sharding involves partitioning the blockchain network into smaller, independent groups, known as shards, with each shard responsible for processing a subset of transactions or smart contracts [46, 91, 101]. This partitioning can improve the overall throughput of the system. However, sharding introduces significant challenges, particularly regarding security and cross-shard communication. One of the main concerns is the potential weakening of the blockchain's security guarantees, as each shard has less total computation power or stakes. Addressing this requires improvements in the consensus protocols for the shards [111]. Another challenge is managing cross-shard transactions, which occur when transactions involve multiple shards. Minimising these cross-shard transactions through efficient shard allocation is critical to reducing overhead. Both of these challenges become more challenging when encrypted databases are concerned, presenting interesting problems that could form the basis for future research.

- **Multi-user blockchain SSE.** Current blockchain-based SSE systems primarily focus on single-user scenarios, which may not be sufficient for real-world applications where multi-party access are essential. While multi-user

SSE schemes have been extensively studied in traditional settings [82,88,117, 118], integrating these schemes into blockchain networks presents unique challenges that have not been thoroughly investigated. Specifically, it requires addressing issues related to the privacy of individual users while allowing for collaborative search functionality in the blockchain trust model. This could be another interesting direction for future research.

- **Verifiable SSE supporting expressive queries.** Current verifiable SSE schemes only support single-keyword queries but not more expressive query types, such as boolean or range queries [27,33]. Given the critical importance of result integrity that has been discussed in this thesis, the development of verifiable SSE schemes that can handle expressive queries would significantly enhance the usability of the system while maintaining the verifiability that is essential for scenarios such as blockchain networks. This can be a priliminary work for equiping the blockchain SSE systems with expressive queries, representing an important direction for future research.

In conclusion, the future directions of blockchain-based searchable encryption systems offer numerous opportunities to enhance functionality, security, and scalability. As SSE technology continues to evolve, integrating advanced SSE schemes with rich features can further improve the usability of both blockchain-based SSE systems and blockchain-based storage systems. However, these advanced schemes may encounter new security challenges when applied within the blockchain security model. Addressing these challenges will be crucial for the widespread adoption of feature-rich blockchain-based SSE systems. Furthermore, due to the inherent nature of blockchain, scalability remains a significant limitation that must be addressed. This challenge becomes even more critical in storage- and communication-intensive applications, such as blockchain-based storage systems and blockchain-based SSE. By advancing these areas, blockchain-based searchable encryption systems can emerge as a viable solution for secure and decentralised data retrieval, meeting the growing demand for privacy-preserving search systems in various real-world applications.

Chapter 7

Conclusion

This thesis demonstrates how to build a practical and secure blockchain-based keyword search system. To achieve this goal, three major contributions are made to address different aspects of the challenges using different methods.

The first contribution of this thesis introduces a novel design for a blockchain-based SSE system, incorporating an innovative off-chain verification method to address the challenges posed by coexisting dishonest server nodes and users. The chapter begins by outlining the fundamental requirements for developing a secure and practical blockchain-based searchable encryption system, which are framed as three key design criteria. Based on these foundational principles, a new blockchain-based SSE construction is proposed. The system is rigorously evaluated through both theoretical analysis and empirical testing, demonstrating that it satisfies all performance requirements and is secure within the defined security model.

The second contribution introduces a non-verification-based solution for blockchain databases. With this approach, we present AegisDB, a blockchain database system that incorporates a VRF-based load-balancing algorithm. Unlike sharding schemes, our method achieves scalability without compromising security. Notably, read queries are not subject to transaction confirmation, resulting in response times that are comparable to those of non-blockchain-based distributed database systems. The integrity of query results is ensured through the blockchain consensus and the load-balancing algorithm, without requiring any verification process. Additionally, this approach mitigates the risk of targeted attacks, which is a weakness for sharded blockchain databases.

The third contribution of this thesis introduces SEARCHAIN, a novel proof-of-useful-work blockchain designed to serve SSE queries. In the first contribution, we introduced the role of verifiers to ensure service fairness between potentially dishonest server nodes and users. However, the incentives for verifiers, as well as in other existing research, remain unclear. SEARCHAIN addresses this gap by proposing a novel committee selection algorithm to choose verifier committee members, ensuring that their voting power is proportional to the number of active SSE services they manage. To participate in the mining process, verifiers are required to perform result verification for SSE services, with the correctness of their work supervised by the consensus of all committee members. This approach provides both security and effective incentives for the system, ensuring that verifiers are motivated to act honestly.

In conclusion, this thesis has made significant contributions to the field of blockchain-based searchable encryption by addressing key challenges related to security, scalability, and incentives. Through the development of novel methods and systems, we have demonstrated how to build a practical, secure, and scalable SSE system that can operate effectively within the trust model and scalability constraints of blockchain environments.

Bibliography

[1] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proceedings of the twenty-third ACM symposium on operating systems principles*, 2011, pp. 85–100.

[2] R. Poddar, T. Boelter, and R. A. Popa, "Arx: An encrypted database using semantically secure encryption," *Cryptology ePrint Archive*, 2016.

[3] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, "Blockchain meets database: Design and implementation of a blockchain relational database," *arXiv preprint arXiv:1903.01919*, 2019.

[4] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable Dynamic Symmetric Searchable Encryption: Optimality and Forward Security." *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 62, 2016.

[5] Y. Yang, H. Lin, X. Liu, W. Guo, X. Zheng, and Z. Liu, "Blockchain-based verifiable multi-keyword ranked search on encrypted cloud with fair payment," *IEEE Access*, vol. 7, pp. 140 818–140 832, 2019.

[6] B. Bünz, S. Goldfeder, and J. Bonneau, "Proofs-of-delay and randomness beacons in ethereum," *IEEE Security and Privacy on the blockchain (IEEE S&B)*, 2017.

[7] "TPC-H Benchmark." [Online]. Available: https://www.tpc.org/tpch/

[8] Z. Wan and R. H. Deng, "VPSearch: Achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data," *IEEE transactions on dependable and secure computing*, vol. 15, no. 6, pp. 1083–1095, 2016.

[9] E. Rezaee, A. M. Saghiri, and A. Forestiero, "A Survey on Blockchain-Based Search Engines," *Applied Sciences*, vol. 11, no. 15, p. 7063, 2021.

[10] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000.* IEEE, 2000, pp. 44–55.

[11] "Drand - A Distributed Randomness Beacon Daemon." [Online]. Available: https://github.com/drand/drand

[12] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[13] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, pp. 21–41, 2004.

[14] N. Gailly, M. Maller, and A. Nitulescu, "SnarkPack: Practical SNARK Aggregation," *Cryptology ePrint Archive*, 2021.

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

[16] L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future generation computer systems*, vol. 95, pp. 420–429, 2019.

[17] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual international conference on the theory and applications of cryptographic techniques.* Springer, 2015, pp. 281–310.

[18] H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," *Computers & Electrical Engineering*, vol. 73, pp. 32–45, 2019.

[19] Y. Zhang, J. Katz, and C. Papamanthou, "IntegriDB: Verifiable SQL for outsourced databases," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1480–1491.

[20] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. Ieee, 2017, pp. 557–564.

[21] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 917–922.

[22] M. Li, J. Zhu, T. Zhang, C. Tan, Y. Xia, S. Angel, and H. Chen, "Bringing Decentralized Search to Decentralized Services," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021, pp. 331–347.

[23] C. Cai, X. Yuan, and C. Wang, "Hardening distributed and encrypted keyword search via blockchain," in *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, 2017, pp. 119–128.

[24] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 598–609.

[25] S. Williams, V. Diordiiev, L. Berman, and I. Uemlianin, "Arweave: A protocol for economically sustainable information permanence," *Arweave Yellow Paper, https://www. arweave. org/yellow-paper. pdf*, 2019.

[26] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," *arXiv preprint arXiv:1711.01030*, 2017.

[27] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 94–124.

[28] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 863–874.

[29] C. Yue, T. T. A. Dinh, Z. Xie, M. Zhang, G. Chen, B. C. Ooi, and X. Xiao, "GlassDB: An Efficient Verifiable Ledger Database System Through Transparency," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1359–1371, 2023.

[30] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich, "Blurring the lines between blockchains and database systems: the case of hyperledger fabric," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 105–122.

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

[32] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual international cryptology conference*. Springer, 2018, pp. 757–788.

[33] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual cryptology conference*. Springer, 2013, pp. 353–373.

[34] D. Adkins, A. Agarwal, S. Kamara, and T. Moataz, "Encrypted blockchain databases," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 241–254.

[35] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM international conference on management of data*, 2017, pp. 1085–1100.

[36] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International conference on the theory and application of cryptology and information security*, 2008, pp. 90–107.

[37] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 455–471.

[38] 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)*, 2018, pp. 583–598.

[39] B. Fisch, "Tight proofs of space and replication," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 324–348.

[40] D. Papadopoulos, S. Papadopoulos, and N. Triandopoulos, "Taking authenticated range queries to arbitrary dimensions," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 819–830.

[41] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 863–880.

[42] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, and Others, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[43] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," *Cryptology ePrint Archive*, 2013.

[44] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, "Verifiable searchable symmetric encryption from indistinguishability obfuscation," in *Proceedings of the 10th ACM symposium on information, computer and communications security*, 2015, pp. 621–626.

[45] A. Soleimanian and S. Khazaei, "Publicly verifiable searchable symmetric encryption based on efficient cryptographic components," *Designs, Codes and Cryptography*, vol. 87, no. 1, pp. 123–147, 2019.

[46] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 931–948.

[47] Nebulas Team, "The value-based blockchain operating system and search engine," *Nebulas Technical White Paper, https://www. nebulas. io/docs/NebulasTechnicalWhitepaper.pdf*, 2017.

[48] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *International conference on financial cryptography and data security*. Springer, 2012, pp. 285–298.

[49] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[50] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, vol. 6, pp. 31 077–31 087, 2018.

[51] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[52] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, and Others, "On scaling decentralized blockchains," in *International conference on financial cryptography and data security*. Springer, 2016, pp. 106–125.

[53] Q. Tang, "Towards blockchain-enabled searchable encryption," *arXiv preprint arXiv:1908.09564*, 2019.

[54] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "Smartpool: Practical decentralized pooled mining," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1409–1426.

[55] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Annual cryptology conference*. Springer, 2013, pp. 90–108.

[56] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain technology for cloud storage: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–32, 2020.

[57] S. Gupta, S. Rahnama, J. Hellings, and M. Sadoghi, "Resilientdb: Global scale resilient blockchain fabric," *arXiv preprint arXiv:2002.00160*, 2020.

[58] A. E. Kosba, D. Papadopoulos, C. Papamanthou, M. F. Sayed, E. Shi, and N. Triandopoulos, "Faster Verifiable Set Computations," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 765–780.

[59] Z. Hong, S. Guo, E. Zhou, W. Chen, H. Huang, and A. Zomaya, "GriDB: Scaling Blockchain Database via Sharding and Off-Chain Cross-Shard Mechanism," *Proceedings of the VLDB Endowment*, vol. 16, no. 7, pp. 1685–1698, 2023.

[60] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 831–843.

[61] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data privacy management, cryptocurrencies and blockchain technology*. Springer, 2017, pp. 297–315.

[62] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Ieee, 2018, pp. 1545–1550.

[63] "Coinbase." [Online]. Available: https://www.coinbase.com/converter/eth/usd

[64] M. Chiesa Alessandro and Green, L. Jingcheng, M. Peihan, M. Ian, and M. Pratyush, "Decentralized Anonymous Micropayments," in *Advances in Cryptology – EUROCRYPT 2017*, J. B. Coron Jean-Sébastien and Nielsen, Ed. Cham: Springer International Publishing, 2017, pp. 609–642.

[65] J. Benet and N. Greco, "Filecoin: A decentralized storage network," *Protoc. Labs*, pp. 1–36, 2018.

[66] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann, and R. Ramamurthy, "BlockchainDB: A shared database on blockchains," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1597–1609, 2019.

[67] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[68] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," *arXiv preprint arXiv:1708.03778*, 2017.

[69] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M. Steiner, and Others, "Dynamic searchable encryption in very-large databases: Data structures and implementation," *Cryptology ePrint Archive*, 2014.

[70] D. Vorick and L. Champine, "Sia: Simple decentralized storage," *Retrieved May*, vol. 8, p. 2018, 2014.

[71] C. Xu, L. Yu, L. Zhu, and C. Zhang, "A blockchain-based dynamic searchable symmetric encryption scheme under multiple clouds," *Peer-to-Peer Networking and Applications*, vol. 14, no. 6, pp. 3647–3659, 2021.

[72] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[73] X. Yan, X. Yuan, Q. Ye, and Y. Tang, "Blockchain-based searchable encryption scheme with fair payment," *IEEE Access*, vol. 8, pp. 109 687–109 706, 2020.

[74] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 668–679.

[75] IPSE Team, "IPSE: A Search Engine Based on IPFS," *IPSE White Paper, https://ipfssearch. io/IPSE-whitepaper-en. pdf*, 2019.

[76] E.-J. Goh, "Secure indexes," *Cryptology ePrint Archive*, 2003.

[77] P. Ruan, D. Loghin, Q.-T. Ta, M. Zhang, G. Chen, and B. C. Ooi, "A transactional perspective on execute-order-validate blockchains," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 543–557.

[78] "Go Ethereum." [Online]. Available: https://github.com/ethereum/go-ethereum

[79] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2018, pp. 792–800.

[80] R. Guerraoui, P. Kuznetsov, M. Monti, M. Pavlovic, D.-A. Seredinschi, and Y. Vonlanthen, "Scalable byzantine reliable broadcast (extended version)," *arXiv preprint arXiv:1908.01738*, 2019.

[81] A. Miller, M. Hicks, J. Katz, and E. Shi, "Authenticated data structures, generically," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 411–423, 2014.

[82] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[83] R. Jain and S. Prabhakar, "Trustworthy data from untrusted databases," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 529–540.

[84] O. F. Cangir, O. Cankur, and A. Ozsoy, "A taxonomy for Blockchain based distributed storage technologies," *Information Processing & Management*, vol. 58, no. 5, p. 102627, 2021.

[85] A. Papadimitriou, R. Bhagwan, N. Chandran, R. Ramjee, A. Haeberlen, H. Singh, A. Modi, and S. Badrinarayanan, "Big data analytics over encrypted datasets with seabed," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 587–602.

[86] L. Sardar and S. Ruj, "Fspvdsse: A forward secure publicly verifiable dynamic sse scheme," in *International Conference on Provable Security*. Springer, 2019, pp. 355–371.

[87] Y. Zhang, C. Papamanthou, and J. Katz, "Alitheia: Towards practical verifiable graph processing," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 856–867.

[88] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International conference on the theory and application of cryptology and information security*. Springer, 2010, pp. 577–594.

[89] K. Kurosawa and Y. Ohtaki, "How to update documents verifiably in searchable symmetric encryption," in *International Conference on Cryptology and Network Security*. Springer, 2013, pp. 309–328.

[90] BitClave, "Active Search Ecosystem," *BitClave White Paper, https://cryptorating. eu/whitepapers/BitClave*, 2017.

[91] 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*, 2016, pp. 17–30.

[92] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 131–144, 2018.

[93] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[94] A. Juels and B. S. Kaliski Jr, "PORs: Proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 584–597.

[95] Swarm Team, "Swarm: Storage and communication infrastructure for a self-sovereign digital society," *Swarm White Paper, https://www.ethswarm.org/swarm-whitepaper.pdf*, 2021.

[96] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 175–192.

[97] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography*, pp. 277–346, 2017.

[98] C. Fournet, M. Kohlweiss, G. Danezis, Z. Luo, and Others, "ZQL: A Compiler for Privacy-Preserving Data Processing." in *USENIX Security Symposium*, 2013, pp. 163–178.

[99] N. Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *Journal of Network and Computer Applications*, vol. 162, p. 102656, 2020.

[100] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco, "Single secret leader election," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 12–24.

[101] 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.

[102] "Etherscan." [Online]. Available: https://etherscan.io/gastracker

[103] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "P2P Mixing and Unlinkable Bitcoin Transactions." in *NDSS*, 2017, pp. 1–15.

[104] Presearch, "The Community-Powered Search Engine," *Presearch White Paper, https://whitepaper. io/coin/presearch*, 2017.

[105] C. Cai, X. Yuan, and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[106] BigchainDB GmbH, "Whitepaper: BigchainDB 2.0 The Blockchain Database," *white paper, BigChainDB*, no. May, pp. 1–14, 2018. [Online]. Available: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

[107] F. M. Schuhknecht, A. Sharma, J. Dittrich, and D. Agrawal, "chainifyDB: How to get rid of your Blockchain and use your DBMS instead." in *CIDR*, 2021.

[108] X. Yuan, Y. Guo, X. Wang, C. Wang, B. Li, and X. Jia, "EncKV: An encrypted key-value store with rich queries," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 423–435.

[109] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, 2016, pp. 45–59.

[110] J. Chen and S. Micali, "Algorand," *arXiv preprint arXiv:1607.01341*, 2016.

[111] 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.

[112] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings*

of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 195–209.

[113] D. Papadopoulos, C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Practical authenticated pattern matching with optimal proof size," *Proceedings of the VLDB Endowment*, vol. 8, no. 7, pp. 750–761, 2015.

[114] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965–976.

[115] S. Bajaj and R. Sion, "CorrectDB: SQL engine with practical query authentication," *Proceedings of the VLDB Endowment*, vol. 6, no. 7, pp. 529–540, 2013.

[116] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.

[117] E. De Cristofaro, Y. Lu, and G. Tsudik, "Efficient techniques for privacy-preserving sharing of sensitive information," in *International Conference on Trust and Trustworthy Computing*. Springer, 2011, pp. 239–253.

[118] S.-F. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for boolean queries," in *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I 21*. Springer, 2016, pp. 154–172.