Searchain: Blockchain-based Private Keyword Search in Decentralized Storage

Peng Jiang[a,*], Fuchun Guo[b,*], Kaitai Liang[c], Jianchang Lai[b], Qiaoyan Wen[a]

[a] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.*
[b] *Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Australia.*
[c] *Department of Computer Science, University of Surrey, UK*

Abstract

Blockchain-based distributed storage enables users to share data without the help of a centralized service provider. Decentralization eliminates traditional data loss brought by compromising the provider, but incurs the possible privacy leakage in a way that the supplier directly links the retrieved data to its ciphertext. Oblivious keyword search (OKS) has been regarded as a solution to this issue. OKS allows a user to retrieve the data associated with a chosen keyword in an oblivious way. That is, the chosen keyword and the corresponding ciphertext are unknown to the data supplier. But if the retrieval privilege is with an authorized keyword set, OKS is unavailable due to one-keyword restriction and public key encryption with keyword search (PEKS) might lead to high bandwidth consumption.

In this paper, we introduce *Searchain*, a blockchain-based keyword search system. It enables oblivious search over an authorized keyword set in the decentralized storage. Searchain is built on top of a novel primitive called *oblivious keyword search with authorization (OKSA)*, which provides the guarantee of keyword authorization besides oblivious search. We instantiate a provably secure OKSA scheme, featured with one-round interaction and constant size communication cost in the transfer phase. We apply OKSA and ordered multisignatures (OMS) to present a Searchain protocol, which achieves oblivious peer-to-peer retrieval with order-preserving transaction. The analysis and evaluation show that Searchain maintains reasonable cost without loss of retrieval privacy, and hence guarantees its practicality.

Keywords: Decentralized Storage, Oblivious Keyword Search, Authorization, Blockchain

*Corresponding author
Email addresses: pennyjiang0301@gmail.com (Peng Jiang), fuchun@uow.edu.au (Fuchun Guo)

1. Introduction

Data storage with encryption is essential for data suppliers to protect their sensitive data from being compromised by network attackers. However, traditional storage systems (e.g., Google, Dropbox and One drive) need an individual service provider to transfer and store the encrypted data. Although considering integrity protection or deduplication [1, 2, 3, 4], these storage systems may suffer from potential security threats (e.g., malware or man-in-the-middle attacks) due to lack of end-to-end encryption. Bitcoin [5] has triggered a new trend of decentralized computing. It brings a great advantage: decentralized control, i.e., no one owns or controls the network. Single monolithic blockchain technology [6] provides an elegant method to achieve decentralized storage. A blockchain is a list of blocks, each covering the encrypted data with verifiability in the current transaction and referring back to previous blocks.

Modern storage systems employ blockchain technology and public key encryption to flexibly share encrypted data, just via a federation of nodes with voting permissions, that is, a peer-to-peer network [7]. Figure 1 depicts the basic framework of blockchain-based storage, where a node speaks to the rest of the nodes without a central party and all of blocks recording transactions are linked to a chain according to their orders. Such a peer-to-peer storage system removes the reliability to the service provider and addresses the security shortcomings from network attacks. It has three features, namely *Decentralized control*, *Immutability* (written data is tamper-resistant and the block is ordered) and *Independent ability to create & transfer assets*.

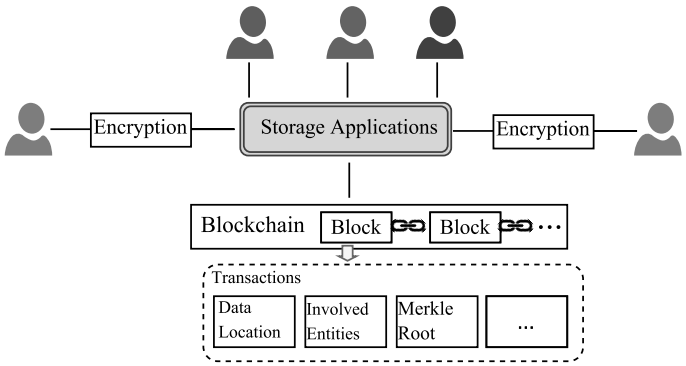


Figure 1: Blockchain-based Storage.

In the practical point of view, data search is necessary while the traditional encryption limits search capabilities. To bridge encryption and search, searchable encryption [8] is designed to allow search over encrypted data without revealing the keyword. Searchable encryption can be realized in either the symmetric setting or the asymmetric setting. Although the symmetric searchable encryption (SSE) enjoys better efficiency [9, 10], it suffers from complicated secret key distribution/management in data sharing stage. To address this issue, Boneh et al. introduced a more flexible primitive, namely public key encryption with keyword search (PEKS) [11] that enables users to search data in the asymmetric encryption setting. Although supporting keyword search

in the asymmetric encryption setting, PEKS is incompatible with decentralized data storage, which requires peer-to-peer communication between nodes without the aid of the server.

Motivation. In the decentralized storage, each node represents a supplier (who speaks and shares its data with others) or a user (who listens and wants to retrieve data). For users, privacy is on the top of the priority list [12], since the data retrieval leakage will reveal their private interests. Users often prefer the retrieval to link no ciphertext, i.e., reveal no additional information. To tackle the issue, Ogata and Kurosawa [13] introduced an interesting notion, called oblivious keyword search (OKS). With a two-party oblivious transfer protocol, OKS allows the user to retrieve the plain data containing the keyword of his choice while the supplier learns nothing about the chosen search keyword and the retrieved plain-cipher data. However, in some special databases, such as databases for commercial secrets and databases for DNA information, data is highly confidential. In such a scenario, users are with various retrieving rights. That is, the choice of a keyword must be within an authorized keyword set previously specified by the supplier and the user. To be precise, if W is the authorized keyword set for a user, the user can search data associated with any keyword $w \in W$ and meanwhile, the retrieving rights corresponding to all the encrypted data associated with W is granted by the supplier. The supplier is able to verify that the chosen keyword belongs to W, while unknowing what the keyword is.

This paper presents a blockchain-based keyword search called *Searchain* that aims to enable oblivious search over conditional keyword privacy in the decentralized storage. Searchain is not a trivial combination of blockchain technique and data retrieval technology. It builds on top of a new notion named oblivious keyword search with authorization (OKSA). The core idea of OKSA is to generate the trapdoor based on the authorized keyword set, the received token as well as the secret key of the supplier, so that the supplier can only know the search keyword belonging to the authorized keyword set but cannot distinguish which one it is. OKSA allows Searchain to support peer-to-peer keyword search and to preserve the retrieval privacy with order by further combining the ordered multisignatures (OMS).

Naive Solution. OKS supports only one keyword and cannot be directly extended to the context where a keyword set is needed. A potential approach is that the supplier encrypts all trapdoors of keywords in W (we assume $|W| = n$) and runs a 1-out-of-n oblivious transfer protocol with the user on each encrypted trapdoor. This however comes to the price of the linear size communication cost between the supplier and each user. It is obvious that this solution does not scale well due to high bandwidth overhead (i.e. $O(n)$).

Our Contributions. To summarize, our contributions are fourfold.

- We introduce the Searchain architecture, a peer-to-peer keyword search system. It allows the user to search his/her interested data and hide the retrieval privacy in the decentralized environment, while the block can record the retrieval transaction with its order. It removes the dependency on a third party and hence eliminates the threat of compromising service provider in data retrieval.

- We propose the notion of OKSA, which augments OKS with the idea of embedding an authorized keyword set. It provides authorization and verification for the search keyword in an oblivious way. We propose a provably secure OKSA instantiation with constant-size transfer communication.
- We present the design of Searchain by using OKSA. The protocol supports encrypted retrieval over an authorized keyword set, while hiding the search keyword and data. The block, generated by OMS to record the retrieval transaction, works seamlessly with keyword search and hence reliably preserves the transaction order.
- We evaluate our proposed Searchain protocol and the results show its scalability and practicality in the decentralized storage.

Differences Between This Work and The Conference Version [14]. In this version, we start from the decentralized storage, and build a blockchain-based data retrieval framework with its objectives and threat model. Under the new framework, we present a Searchain protocol by employing OKSA, which was proposed in [14], and OMS. In addition, we evaluate the performance of the Searchain protocol by functionality analysis and implementation. So the conference version [14] is just one of four contributions in this work.

Organization. The rest of this paper is organized as follows. In Section 2, we review some related work. Section 3 and Section 4 describe the Searchain overview and OKSA, respectively. We present a Searchain protocol from OKSA in Section 5 and evaluate it in Section 6. The formal security proof of OKSA is given in Section 7. Finally, we conclude the paper in Section 8.

2. Related Work

Blockchain Technology. Decentralized cryptocurrency (e.g., Bitcoin [5]) has gained popularity and is also quoted as a glimpse in the future [15]. The cryptocurrency system builds on top of a novel technology named blockchain [16], which is essentially a distributed database of transactions. Digital information has been executed and shared among participating parties and allows public ledger of all transactions. A blockchain is composed of verifiable records for each single transaction ever made which is verified by consensus of a majority of the participants in the system. Blockchain technology is finding applications in wide range of non-financial areas besides current financial areas, such as decentralized proof of the existence of documents [17], decentralized IoT [18] and decentralized storage [19].

Blockchain-based storage has become a newly of growth engine in data sharing since it does not need a central service provider. We find that the data retrieval approaches are rarely studied in the blockchain-based system. To date, a personal data management system [20] was proposed with the assistance of the blockchain technology to ensure users own and control their data against the honest-but-curious services. A decentralized smart contract system named Hawk [15] was proposed to retain transaction privacy from the public's view, while no detailed retrieval algorithm was

given. A healthcare chain [21] was constructed to facilitate data interoperability in health information networks. However, these systems focus on the concepts with the corresponding frameworks instead of the concrete algorithms to guarantee data utilization and data secrecy. Also, the linkable transaction privacy is still a margin in the blockchain-based retrieval.

Public Key Encryption with Keyword Search. Boneh et al. introduced the notion of public key encryption with keyword search (PEKS) [11] to address the issue of the complicated key management in SSE and achieve search over the encrypted data in the asymmetric setting. Afterwards, combinable multi-keyword search schemes have been proposed to provide diverse search functionality, such as public-key encryption scheme with conjunctive keyword search (PECKS) [22, 23, 24] and public key encryption with temporary keyword search (PETKS) [25]. To improve the keyword privacy, secure channel free-PEKS (SCF-PEKS) schemes [26] and ciphertext retrieval against insider attacks (CR-IA) [27] were proposed to resist outsider attacks and insider attacks, respectively, and public key encryption with oblivious keyword search (PEOKS) [28] was proposed to permit authorized private search. We find that PEKS is unsuitable in the two-party database operation.

Oblivious Transfer. Originally, the notion of oblivious transfer was introduced by Rabin [29], which is a two-party protocol between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. $\mathcal{S}$ has two bits and $\mathcal{R}$ wishes to get one of them satisfying the followings properties: $\mathcal{S}$ does not know which bit $\mathcal{R}$ obtains, and $\mathcal{R}$ does not know any information about the bit that he did not obtain. In an OT system, the most general type is k-out-of-n oblivious transfer (OT_n^k), where $\mathcal{S}$ holds n messages and $\mathcal{R}$ retrieves k of them simultaneously, such that $\mathcal{S}$ does not know which messages $\mathcal{R}$ obtains. There have been many works on oblivious transfer, such as adaptive oblivious transfer [30, 31], oblivious transfer with fully simulatable security [32], oblivious transfer with universally composable security [33], oblivious transfer with access control [34] and priced oblivious transfer [35, 36]. Some proposed OT_n^k protocols, such as [37, 38], have ideal communication rounds.

Oblivious Keyword Search. Ogata and Kurosawa [13] introduced the notion of oblivious keyword search to address the user privacy issue in the keyword search, which was based on a two-party OT protocol between a supplier and a user. Their OKS employed the blind signature, where the ciphertext is generated with the master secret key of the supplier (denoted by msk) and some keyword, and each trapdoor is transferred from the supplier to the user using msk and the keyword token generated by the user. Rhee et al. [39] presented an oblivious conjunctive keyword search to allow search over boolean combinations of keywords. Freedman et al. [40] considered privacy concerns in keyword search using oblivious evaluation of pseudorandom functions. Zhu and Bao [41] addressed the OKS in the public database by using linear and non-linear oblivious polynomial evaluation. Camenisch et al. [28] proposed the public key encryption with oblivious keyword search (PEOKS) to build a public key encrypted database permitting private information retrieval (PIR), where computationally expensive zero-knowledge proof (ZKP) was employed.

Ordered Multisignatures. Ordered Multisignatures (OMS) was proposed in [42] to allow signers to attest to a common message as well as the order in which they signed. In

OMS, a group of signers sequentially form an aggregate by each adding their own signature to the aggregate-so-far. [43] proposed sequentially aggregate signed data based on uncertified claw-free permutations in the random oracle model, to minimize the total amount of transmitted data rather than just the signature length. [44] presented a practical synchronized aggregate signature scheme without interactive complexity assumption used in [42]. [45] constructed a provable secure OMS scheme in the standard model, which also improved the efficiency compared with original OMS [42].

3. Searchain Overview

3.1. Architecture

Figure 2 depicts an overview of the Searchain architecture. It includes transaction nodes with a peer-to-peer structure and a blockchain with all of ordered blocks. Their functions and the Searchain workflow are described as follows.

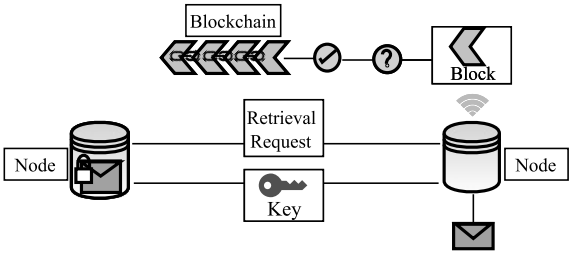


Figure 2: Searchain Architecture.

- Node. Nodes can share data in a peer-to-peer mode. A node plays the role of the supplier, the user or verifier. As a user, the node generates the data retrieval request, and accordingly creates a block and broadcasts it. As a supplier, the node who owns the data can respond the request to help retrieve the data. As a verifier, the node collects the unconfirmed block and approves it or not. No third party controls the data retrieval process.
- Block and Blockchain. A block provides a record for the current data retrieval information (or we call it as transaction record). The block can be added to the chain after being approved by most nodes in the network. We note that the optimization issue on the number of nodes in the block approval is out of the scope in this work.

Workflow. We assume that the transaction happens between two nodes, i.e., Node A and Node B, where Node A acts as a data supplier and Node B acts as a user. The goal of Node B is to retrieve the data owned by Node A. *Searchain* mainly consists of the following five phases.

Initialization Transaction. Node A generates some parameters for this transaction, where the public parameters are public while the secret key is only kept by Node A.

Node A negotiates a keyword set with Node B. Node A can control whether Node B is able to retrieve his data.

Data Sharing. Node A leverages the encryption module to handle the sensitive plaintext data before sharing with other nodes. Each plaintext data is associated with its respective keyword. The ciphertext data should provide both search capability and confidentiality, that is, a valid node can search and access the plain data. The ciphertext data is transmitted in the public network and available to any other nodes.

Retrieval Request. Node B, who submits a request to Node A for his/her retrieval target data. To hide the target of Node B, this request should be in an encrypted form, e.g., the encryption of some keyword. Meanwhile, a block which can record the data transaction is generated. After that, the request and the block are broadcasted to other nodes in the network.

Verification. The block needs to be approved for validity before it is added to the chain with an unchanged order. Node A verifies the validity of the request, that is, the keyword in this request is in the negotiated keyword set, and thereafter distributes a key to Node B. During the request verification, decryption key generation and distribution, Node A learns nothing about the retrieved data.

Data Retrieval. Upon receiving a valid key, Node B can search and access its interested data with other secret information.

3.2. Objectives

Searchain works with each block being verified publicly. Considering the structure features of the blockchain-based storage system and the properties of keyword search over encrypted data, *Searchain* aims to satisfy the objectives, i.e., Decentralizing, Rule Independence, Transaction Order-preserving, Secrecy and Retrieval Privacy.

- Decentralizing. Searchain provides peer-to-peer data retrieval in the distributed storage without relying on a third party (e.g., the storage server).
- Rule Independence. Searchain should provide each retrieval transaction to be executed independently. On one hand, it sets rules about a transaction (business logic) that are tied to the transaction itself instead of at the entire database level.
- Transaction Order-preserving. Searchain should guarantee transactions to be executed orderedly. That is, each transaction block can be linked into the previous chain with a non-broken order.
- Secrecy. Searchain provides the secrecy of the to-be-search data and its associated keywords.
- Retrieval Privacy. Under a case that Node B wants to retrieve data from Node A, Searchain supports the Node B's retrieval privacy. That is, Node A does not know the retrieved plain-cipher data and its associated keyword although assuring the authorization of this keyword.

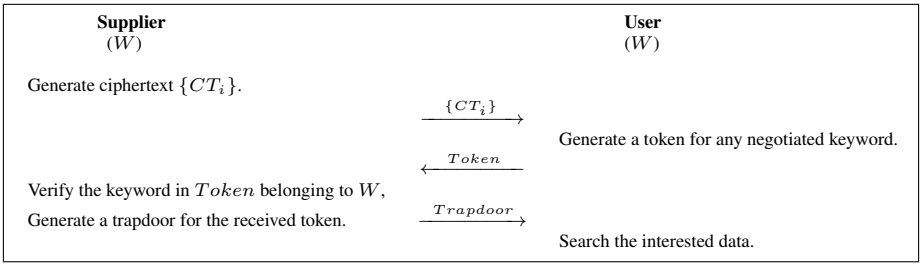


Figure 3: OKSA Framework.

3.3. Threat Model

This paper focuses on *decentralized storage* that achieves peer-to-peer data retrieval while tying the records into the blockchain. This blockchain is an ordered sequence of all retrieval transactions. We assume that each transaction maps to a unique block that is added to the previous chain, such that it provides an indelible and transparent record. We also assume that each plaintext data is encrypted into the corresponding ciphertext that is searchable. In addition, we allow all adversaries to access the system public parameter and the transferred data stream in the public channel.

We consider that there exists an "honest-but-curious" adversary that follows all prescribed protocols and attempts to infer the retrieved data based on the two attack modes: (i) The *chosen plaintext attack* models the case that an adversary knows some of the ciphertexts of arbitrary plaintexts and guesses the used plaintext. (ii) The *chosen keyword attack* models the case that an adversary knows some of the ciphertexts of arbitrary keywords and guesses the used keyword.

Our threat model makes the following assumptions. First, all keywords are short strings and allowed to guess by brute-force attacks. Second, we default the target node itself not to be the adversary, that is, the adversary cannot identify the secret key with associated information. Third, we do not consider any collusion of entities with different roles. Here we disallow private, peer-to-peer communication between the nodes except via the system's built-in communication, for great savings in complexity and for reduced security risk. This means that malicious nodes are unable to transmit data to others.

4. Oblivious Keyword Search with Authorization

In this section, we build blocks of OKSA, which follows the supplier-user mode in OKS [13]. We systematically study the keyword authorization problem in the oblivious keyword search, where the supplier has an agreed authorized keyword set with each user. In OKSA, the user generates a keyword token for any keyword in the authorized keyword set and thereafter the supplier generates the trapdoor with the received token, his secret key and the authorized keyword set. Figure 3 presents the OKSA framework and its detailed algorithms are described as follows.

4.1. Algorithm Definition

Definition 1. *An oblivious keyword search with authorization scheme consists of the following polynomial time randomized algorithms.*

Setup. The supplier $\mathcal{T}$ takes a security parameter λ and an integer n as input, and outputs the public parameter pp and the master public/secret key pair (mpk, msk) to establish the system. Note we assume pp is implicitly included in the following algorithm. $\mathcal{T}$ negotiates a keyword set W with each user, where $|W| \leq n$.

Commit. $\mathcal{T}$ takes a message m_i, a keyword w_i and the master public key mpk as input, and outputs the ciphertext CT_i, where each message m_i has its own unique keyword w_i. $\mathcal{T}$ commits all ciphertexts $\{CT_i\}$ to the user $\mathcal{U}$.

Transfer.

Transfer_1. $\mathcal{U} \rightarrow \mathcal{T}$: $\mathcal{U}$ takes the authorized keyword set W, a specified keyword $w_i' \in W$ and the master public key mpk as input, and outputs the keyword token $\mathbf{P}(w_i')$, the secret key of the user sk and the proof information for accountability Σ. Then $\mathcal{U}$ sends $(\mathbf{P}(w_i'), \Sigma)$ to $\mathcal{T}$. Here, $\mathbf{P}(w_i')$ is computed from sk, w_i', W, mpk. Σ helps $\mathcal{T}$ to verify the accountability, that is, the received token is used to generate a trapdoor for only one keyword in the authorized keyword set.

Transfer_2. $\mathcal{T}$: $\mathcal{T}$ takes the received keyword token $\mathbf{P}(w_i')$, the authorized keyword set W and the master public key msk as input. It verifies the accountability by checking $|\mathbf{P}(w_i')| = 1$.

Transfer_3. $\mathcal{T} \rightarrow \mathcal{U}$: Once the verification passes, $\mathcal{T}$ outputs a trapdoor T to $\mathcal{U}$.

Transfer_4. $\mathcal{U}$: $\mathcal{U}$ takes CT_i, T, sk as input and outputs m_i if $w_i = w_i'$, otherwise, $\perp$.

Correctness. An oblivious keyword search with authorization is correct if the user obtains the message of his choice when all of entities follow the protocol steps above. Also, passing the verification of accountability means that the trapdoor generated from the received token will be for only one specific keyword and this specific keyword is in the authorized keyword set.

4.2. Security Notions

Based on [11] and [13], we define security requirements to be user privacy, indistinguishability and accountability. User privacy guarantees that the supplier $\mathcal{T}$ does not learn the search keyword from the user's token in the i-th Transfer sub-phase. Indistinguishability guarantees that a malicious user $\mathcal{U}$ cannot distinguish the message and keyword from the ciphertext. Accountability guarantees that the trapdoor the user asks for is for only one keyword in the authorized keyword set.

- (User Privacy.) Given $(\mathbf{P}(w), \Sigma)$ and two keywords w_0, w_1, it is hard to distinguish whether $w = w_0$ or w_1.
- (Indistinguishability.) Given two message-keyword tuples $(m_0, w_0), (m_1, w_1)$ and a ciphertext CT for (m, w), it is hard to distinguish $(m, w) = (m_0, w_0)$ or $(m, w) = (m_1, w_1)$.

- (Accountability.) Given $(\mathbf{P}(W), W, sk)$ satisfying $|W| > 1$, it is hard to generate $(\mathbf{P}(W), \Sigma)$ that passes the verification.

Based on the above requirements, we define the security models via the following games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. More formally,

User Privacy Game.

Setup. $\mathcal{C}$ runs the Setup algorithm to generate mpk and sends it to $\mathcal{A}$.
Challenge. $\mathcal{A}$ gives two keywords w_0, w_1 to $\mathcal{C}$. $\mathcal{C}$ responds by choosing a coin $\theta \in \{0, 1\}$, setting $w = w_\theta$ and generating $(\mathbf{P}(w), \Sigma)$.
Guess. $\mathcal{A}$ outputs θ' and wins the game if $\theta' = \theta$.

We define $\mathcal{A}$'s advantage as $\mathsf{Adv} = |\Pr[\theta' = \theta] - 1/2|$.

Definition 2. *We say that an OKSA scheme satisfies user privacy if there exists no probabilistic polynomial time adversary to win the above user privacy game with a non-negligible advantage.*

Indistinguishability Game.

Setup. $\mathcal{C}$ runs the Setup algorithm to generate mpk and sends it to $\mathcal{A}$.
Phase 1. $\mathcal{A}$ makes the trapdoor query for w and $\mathcal{C}$ responds with the trapdoor T.
Challenge. $\mathcal{A}$ gives two same length message-keyword tuples $(m_0, w_0), (m_1, w_1)$ to $\mathcal{C}$ with the restriction that w_0, w_1 have not been issued the trapdoor queries in **Phase 1**. $\mathcal{C}$ responds the challenge ciphertext CT^* for randomly choosing $\theta \in \{0, 1\}$.
Phase 2. $\mathcal{A}$ issues more trapdoor queries with the same restriction in **Challenge**, $\mathcal{C}$ responds as **Phase 1**.
Guess. $\mathcal{A}$ outputs θ' and wins the game if $\theta' = \theta$.

We define $\mathcal{A}$'s advantage as $\mathsf{Adv} = |\Pr[\theta' = \theta] - 1/2|$.

Definition 3. *We say that OKSA has indistinguishability against chosen keyword attack if there exists no probabilistic polynomial time adversary to win the above game with a non-negligible advantage.*

Accountability Game.

In OKSA, the verification of accountability is to assure that the trapdoor is for only one authorized keyword. It captures the attack that an adversary $\mathcal{A}$ can forge a proof for a valid keyword token $\mathbf{P}(W')$, where W' is a subset of the authorized keyword set W with $1 < |W'| < |W| \leq n$. Here, the validness means that $\mathcal{A}$ knows W', W, sk of computing $\mathbf{P}(W')$.

Setup. $\mathcal{C}$ runs the Setup algorithm to generate mpk and sends it to $\mathcal{A}$.
Challenge. $\mathcal{A}$ outputs $(\mathbf{P}(W'), W, W', sk)$ and 1 for challenge, where $\mathbf{P}(W')$ is generated from W, W', sk, mpk and $|W'| > 1$.
Win. $\mathcal{A}$ outputs $(\mathbf{P}(W'), \Sigma)$ and wins the game if $(\mathbf{P}(W'), \Sigma)$ passes the verification algorithm.

We define $\mathcal{A}$'s advantage as Adv in computing $(\mathbf{P}(W'), \Sigma)$.

Definition 4. *We say that OKSA has accountability if there exists no polynomial time adversary to win the above game with a non-negligible advantage.*

4.3. Construction

In this section, we propose an oblivious keyword search with authorization protocol. Our protocol allows the user to obliviously obtain an authorized trapdoor by submitting a keyword token adaptively. It features with constant size communication cost between the supplier and the user. The proposed scheme achieves that $\mathcal{T}$ can generate the trapdoor for any keyword in the authorized keyword set but cannot guess which one it is. Like OKS, OKSA is played between a supplier $\mathcal{T}$ and a user $\mathcal{U}$, and it consists of three phases: Setup, Commit and Transfer as follows.

Setup. $\mathcal{T}$ takes as input a security parameter λ, an integer n. It chooses a bilinear map system $\mathcal{PG} = (p, \mathbb{G}, \mathbb{G}_T, e)$ [46] and a cryptographic hash function $H : (\{0,1\}, \mathbb{G}_T) \rightarrow \{0,1\}^\ell$. It also randomly selects $g, h \in \mathbb{G},\ \alpha, x \in \mathbb{Z}_p$ and computes $g^\alpha, h_i = h^{\alpha^i}$ for $i = 1, 2, \cdots, n$. The public parameter is denoted as $pp = (\mathcal{PG}, H, g, h)$, and the master public/secret key pair is

$$mpk = (g^\alpha, h_1, h_2, \cdots, h_n),\ msk = \alpha.$$

$\mathcal{T}$ publishes pp, mpk to all and keeps msk private.

Commit. The universal keyword space is denoted as $\mathcal{KS}$ with the size n. Each message has its associated keyword. Given a message $m_i \in \{0,1\}^\ell$ and a keyword $w_i \in \mathcal{KS}$, $\mathcal{T}$ chooses $r_i \in_R \mathbb{Z}_p$ and computes the encrypted message CT_i as

$$CT_i = \left(c_{1i} = g^{r_i(\alpha+w_i)}, c_{2i} = H\left(0, e\left(g,h\right)^{r_i}\right), c_{3i} = H\left(1, e\left(g,h\right)^{r_i}\right) \oplus m_i\right).$$

$\mathcal{T}$ commits all the ciphertexts $\{CT_i\}$ to $\mathcal{U}$.

Remark. There is a little difference between our OKSA and traditional OKS [13]. In OKS, the ciphertext is based on the master secret key and only the supplier, who holds msk, can generate it. In our OKSA, the ciphertext is based on the master public key and anyone in the system can generate it.

Transfer.

We suppose $\mathcal{T}$ negotiates a unique keyword set W with each user, where $W \subseteq \mathcal{KS}$ and the size of W is denoted as $|W| = k \leq n$.

Transfer_1. $\mathcal{U} \rightarrow \mathcal{T}$: Given the authorized keyword set W, a keyword $w_i \in W$ and the master public key mpk, $\mathcal{U}$ picks $s \in_R \mathbb{Z}_p$ as his secret key $sk = s$ and computes the token $\mathbf{P}(w_i)$ and the proof Σ as

$$\mathbf{P}(w_i) = h^{s\prod_{w_j \in W, j \neq i}(\alpha+w_j)}, \Sigma = \left(\Sigma_1 = h^{\frac{\alpha+w_i}{s}},\ \Sigma_2 = \Sigma_1^{\alpha^{n-1}}\right).$$

Then $\mathcal{U}$ sends $(\mathbf{P}(w_i), \Sigma)$ to $\mathcal{T}$.

Transfer_2. $\mathcal{T}$: Given the tuple $(\mathbf{P}(w_i), W, \Sigma)$ and the master public key mpk, $\mathcal{T}$ checks the accountability by the following equations,

$$e\left(\Sigma_2, h^\alpha\right) = e\left(\Sigma_1, h^{\alpha^n}\right), e\left(h, h^{\prod_{w_i \in W}(\alpha+w_i)}\right) = e\left(\mathbf{P}(w_i), \Sigma_1\right).$$

If both equations hold, $\mathcal{T}$ accepts the received keyword token is for the trapdoor for one keyword, and we denote it as $|\mathbf{P}(w_i)| = 1$; otherwise, aborts.

Transfer_3. $\mathcal{T} \rightarrow \mathcal{U}$: Given msk and W, $\mathcal{T}$ computes the trapdoor T as

$$T = \mathbf{P}(w_i)^{\frac{1}{\prod_{w_j \in W}(\alpha + w_j)}}.$$

Then $\mathcal{T}$ returns the trapdoor T to $\mathcal{U}$.

Transfer_4. $\mathcal{U}$: Given $CT_i, T, sk, \mathcal{U}$ executes the searching operation by

$$c_{2i} = H\left(0, e\left(c_{1i}, T\right)^{\frac{1}{s}}\right).$$

If the above equation holds, $\mathcal{U}$ continues the decryption operation by

$$m_i = c_{3i} \oplus H\left(1, e\left(c_{1i}, T\right)^{\frac{1}{s}}\right).$$

Correctness. Given the master public secret key pair (mpk, msk) from running Setup algorithm and token/proof tuple $(\mathbf{P}(w_i), \Sigma)$, the correctness of the accountability is verified by the following equations.

$$\begin{aligned} e\left(\Sigma_2, h^{\alpha}\right) &= e\left(\Sigma_1^{\alpha^{n-1}}, h^{\alpha}\right) = e\left(\Sigma_1, h^{\alpha^n}\right), \\ e\left(h, h^{\prod_{w_i \in W}(\alpha + w_i)}\right) &= e\left(h^{s\prod_{w_j \in W, j \neq i}(\alpha + w_j)}, h^{\frac{\alpha + w_i}{s}}\right) = e\left(\mathbf{P}(w_i), \Sigma_1\right). \end{aligned}$$

Given a ciphertext CT_i from running the Commit algorithm, a trapdoor T from running the Transfer algorithm and the secret key of the user sk, the correctness of searching and decryption can be verified by

$$\begin{aligned} H\left(0, e\left(c_{1i}, T\right)^{\frac{1}{s}}\right) &= H\left(0, e\left(g^{r_i(\alpha + w_i)}, h^{\frac{s}{\alpha + w_i}}\right)^{\frac{1}{s}}\right) = H\left(0, e\left(g, h\right)^{r_i}\right) = c_{2i} \\ c_{3i} \oplus H\left(1, e\left(c_{1i}, T\right)^{\frac{1}{s}}\right) &= H\left(1, e\left(g, h\right)^{r_i}\right) \oplus m_i \oplus H\left(1, e\left(g^{r_i(\alpha + w_i)}, h^{\frac{s}{\alpha + w_i}}\right)^{\frac{1}{s}}\right) \\ &= H\left(1, e\left(g, h\right)^{r_i}\right) \oplus m_i \oplus H\left(1, e\left(g, h\right)^{r_i}\right) = m_i. \end{aligned}$$

4.4. Security

OKSA achieves *User Privacy, Indistinguishability* and *Accountability*. The formal security proof will be presented later in Section 7.

5. Searchain Design

This section shows a Searchain protocol from OKSA and OMS. OKSA allows the user to obliviously retrieve data from the supplier without a third party. We employ the OMS into the block generation, which records data sharing information of the current transaction. All blocks can thereafter be added into the chain, where OMS provides the attestation for the order of the data retrieval transaction. This chain guarantees the

transparency and order of the record. Searchain hides the retrieved plain-cipher data but verifies the authorization of the keyword in the decentralized storage. The used notations are summarized in Table 1.

Table 1: Notations Used in Searchain.

Notation	Description
W	A authorized keyword set negotiated by two nodes.
w_i	A keyword from the authorized keyword set.
m_i	Plaintext data associated with the keyword w_i.
CT	Ciphertext data shared by a node (i.e., supplier).
Req	A request for retrieving data associated with some keyword.
σ'	A block, used to record information in the current retrieving transaction.
Key	A key, used to retrieve the plain data associated with some keyword from the cipher data.

Before presenting the Searchain protocol, we review the OMS scheme [42].

Definition 5. *An OMS scheme* $\mathbf{OMS} = (\mathsf{OPg}, \mathsf{OKg}, \mathsf{OSign}, \mathsf{OVf})$ *consists of four algorithms.*

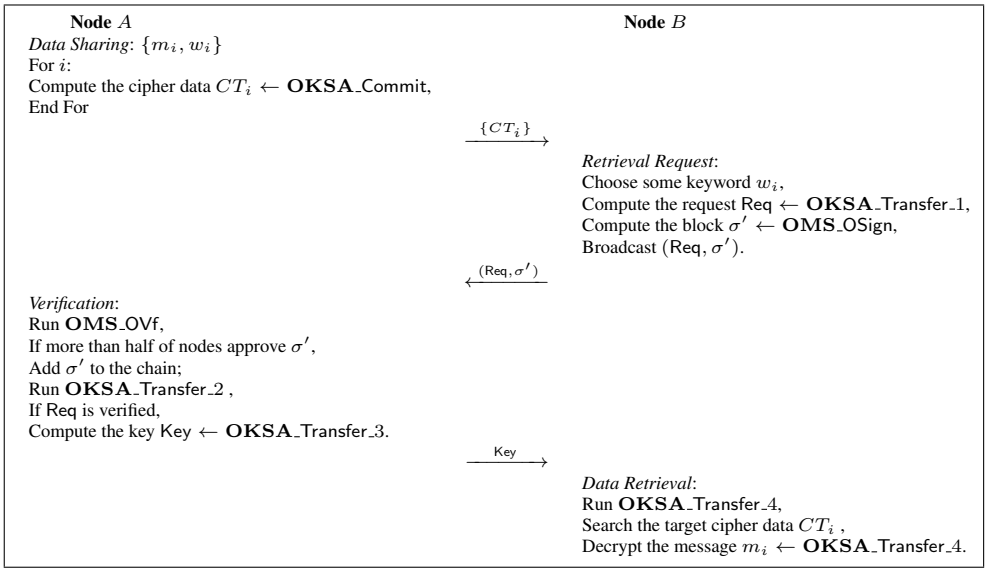
OPg. The parameter generation algorithm returns some global information for the scheme.
OKg. The key generation algorithm inputs global information and returns a public-private key-pair (pk, sk).
OSign. The signing algorithm inputs the secret key sk, a message m, an OMS-so-far $sigma$ and a list of $i-1$ public keys $L = (pk_1, \cdots, pk_{i-1})$, and returns a new OMS σ', or $\perp$ if the input is deemed invalid.
OVf. The verification algorithm inputs a list of public keys $(pk_1, \cdots, pk_n)$, a message m and an OMS σ', and returns a bit.

We intuitively illustrate the Searchain protocol in Figure 4 and describe its details as follows. In the Searchain protocol, we just consider an independent transaction between two nodes. We say that the block will be added into the chain once approved by more than half of nodes in this system. Nodes will adopt the same verification algorithm for the same block.

5.1. Initialization Transaction

We assume that this data retrieval transaction happens between Node A and Node B. Node A pre-sets the basic parameters (we can also regard them as the transaction rule), which are denoted as $\mathcal{R} = \{\mathcal{PG}, g, h, H, \mathcal{KS}\}$. Each node accepts the rule and operations used in the protocol also follow the rule. Given the transaction rule $\mathcal{R}$, Node A and Node B generates the public/secret key pair, respectively.

T.1 Node A runs the $\mathbf{OKSA}\_\mathsf{Setup}$ algorithm to generate its public/secret key (pk_A, sk_A), where $pk_A = \mathbf{OKSA}\_mpk,\ sk_A = \mathbf{OKSA}\_msk$.
T.2 Node B runs $\mathbf{OMS}\_\mathsf{OKg}$ algorithm to generate its public/secret key (pk_B, sk_B), where $pk_B = \mathbf{OMS}\_pk,\ sk_B = \mathbf{OMS}\_sk$.
T.3 Node A and Node B negotiate a keyword set $W \subseteq \mathcal{KS}$.



We note that other nodes also verify the block σ'. Only when approved by more than half of nodes, this block are denoted as passing the verification and added into the chain.

Figure 4: Searchain Protocol.

5.2. Data Sharing

If Node A wants to shares its data with other nodes, it firstly needs to encrypt the data as well as its associated keyword before transferring the ciphertext data to the network.

S.1 For the message $m_i \in \{0,1\}^\ell$ and its associated keyword $w_i \in W$, Node A runs the **OKSA**_Commit algorithm and generates the ciphertext CT_i.

S.2 Node A sends out all his encrypted data $\{CT_i\}$ to some location in the public network.

5.3. Retrieval Request

Since aiming to retrieve the data owned by Node A, Node B needs to generate a request, which is for Node B's interest while hiding that. Also, Node B creates a block to record this transaction. Before asking the request, Node B can find available storage location of $\{CT_i\}$ through the pre-set interface. In this paper, we give no consideration about how to set interface and find the ciphertext location.

R.1 Node B selects a keyword $w_i \in W$ based on its interest.

R.2 Node B runs the **OKSA**_Transfer_1 and generates a request Req for keyword w_i. This request is verifiable and used to ask Node A for the retrieving key.

R.3 Node B runs the **OMS**_OSign(Req) and generates σ'. We remark that the block should include σ' as well as data location, merkle root and so on. Concrete operations about data location and merkle root are out of the scope, therefore, we just regard σ' as the block for this transaction in this work.

R.4 Node B broadcasts Req, σ' to the whole network.

5.4. Verification

The verification includes transaction validity and the request authentication. During this phase, Node A executes no decryption operation to catch the keyword w_i in the request.

V.1 Any node runs the $\mathbf{OMS}\_\mathsf{OVf}$ algorithm to verify the block. Once more than half of nodes in the network approve this transaction, the block σ' is added to the chain.

V.2 Node A runs the $\mathbf{OKSA}\_\mathsf{Transfer}\_2$ algorithm to check whether the request is for a negotiated keyword.

V.3 Once accepting it, Node A runs the $\mathbf{OKSA}\_\mathsf{Transfer}\_3$ algorithm to generate a key Key to Node B. This key helps Node B to search and access the data associated with w_i from Node A.

5.5. Data Retrieval

Upon receiving a valid key T for w_i, Node B executes the decryption to obtain the data associated with w_i.

U.1 Node B runs the $\mathbf{OKSA}\_\mathsf{Transfer}\_4$ to search CT_i from all the ciphertexts $\{CT_i\}$.

U.2 Node B runs the $\mathbf{OKSA}\_\mathsf{Transfer}\_4$ to access the data m_i for w_i.

6. Performance Evaluation

We deploy a static environment composed of 10 nodes, without adding or revoking nodes. The block will be seen as to be valid once 6 nodes approve it, respectively. We employ the OKSA construction in Section 4.3 and OMS construction [42] to instantiate our Searchain protocol. We conduct the algorithm implementation on a Windows system with an Intel(R) Core(TM) i5-2400 CPU@3.10GHz 4 cores 8G Memory. We exploit PBC (Version 0.5.12) and OpenSSL (Version 1.0.1c) libraries, where the language is C. We select a symmetric elliptic curve α-curve [47, 48], where the base field size is 512-bit and the embedding degree is 2. The α-curve has a 160-bit group order, which means p is a 160-bit length prime. We quantify the transfer bandwidth between two nodes and computation overhead of different phases in one transaction, where there are 10 plain messages. Since there are no previously comparable schemes, we only consider Searchain in the following experiments.

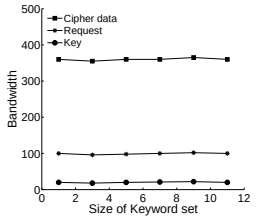


Figure 5: Transfer Bandwidth (bytes).

Transfer Bandwidth. We test the bandwidth from three kinds of parameters, including cipher data, request and key and exclude the response bandwidth from other nodes for block approval. We select a hash function whose output is 64-bits and the plain data with the same-length string. We vary the size of the negotiated keyword set and test the corresponding parameter size. The experiment results are shown in Figure 5. It is clear to see that each parameter is almost of size constant with the increase of the size of the negotiated keyword set, respectively. Therefore, the bandwidth for transfer is independent of the keyword set, as well.

Computation Overhead. We measure the computation overhead in different phases. The detailed settings and necessary assumptions depend on the corresponding phases.

- *Data Sharing* phase. We test the computation time to generate the cipher data. Encryption for each message is an independent process, so the measurements run ten times for ten pre-set messages, respectively. Figure 6 presents the data encryption speed versus the size of the authorized keyword set. We can observe that the computation time in *Data Sharing* phase is independent of the size of the keyword set.
- *Retrieval Request* phase. We test the computation time to generate the request and the block Req, σ'. Our measurements rely on that the size of the keyword set is i in the i-th retrieval transaction and that the size of the keyword set varies from 1 to 10. We show the measured results in Figure 7. From Figure 7, the curve of the request generation speed keeps a linear growing trend with the size of the keyword set.
- *Verification* phase. We measure the verification speed, which includes the block approval, the request accountability and key generation. In our experiment, we make an assumption for block verification, that is, the block is valid as long as 6 nodes approve it. Figure 8 shows the verification speed versus the size of the keyword set. We can see that this phase costs linear-size-increasing computation time when the keyword set includes more keywords.
- *Data Retrieval* phase. We test the time to retrieve one message. As the Searchain protocol in Section 4, the computation operations should be mainly contributed by the data search and decryption. The result is presented in Figure 9. When the size of the keyword set increases, there is no explicit changing engendered in the time to retrieve the message. This tallies with the Searchain protocol, which has an independent retrieval algorithm of the keyword set.

7. Security Analysis

7.1. Assumptions

We define two hard problems to provide foundation for the security of OKSA, i.e., (f, n)*-DHE Problem* and (f, q)*-MSE-DDH Problem.* Since (f, n)*-DHE Problem* has been proposed and analyzed in [49, 50], we only give its description and omit its intractability analysis. We refer readers to the corresponding references for details.

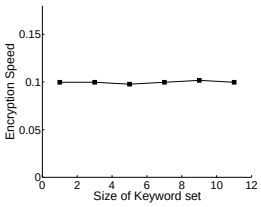


Figure 6: Computation Time in Data Sharing Phase (ms).

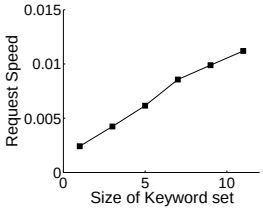


Figure 7: Computation Time in Retrieval Request Phase (ms).

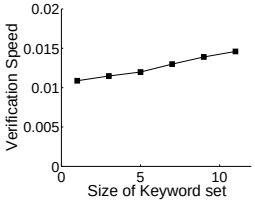


Figure 8: Computation Time in Verification Phase (ms).

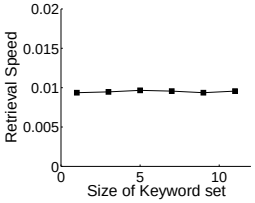


Figure 9: Computation Time in Data Retrieval Phase (ms).

Definition 6. (f, n)**-DHE Problem.** *Let $\mathbb{G}$ be a group of prime order p, $h \in \mathbb{G}$ and $a \in \mathbb{Z}_p$. Given $h, h^a, \cdots, h^{a^n}$, output $(f(x), h^{f(a)})$, where $f(x) \in \mathbb{Z}_p[x]$ is a polynomial function with deg $f(x) > n$.*

Then we introduce a new hard problem named (f, q)*-MSE-DDH Problem*, which is slightly modified from MSE-DDH problem while still preserving its hardness. Our (f, q)-MSE-DDH problem is a special instance of general Diffie-Hellman exponent assumptions in [46], and its intractability will be analyzed later on.

Definition 7. (f, q)**-MSE-DDH Problem.** *Let $\mathcal{PG}$ be a bilinear map group system and g_0, h_0 be the generators of the group $\mathbb{G}$. We assume two pairwise co-prime polynomials f and q with degree 1 and $n-1$, respectively, where n is an integer. Given $g_0, g_0^{\alpha}, g_0^{r}, h_0^{f(\alpha)}, \cdots, h_0^{\alpha^{n-2}f(\alpha)}, h_0^{f(\alpha)q(\alpha)}, \cdots, h_0^{\alpha^n f(\alpha)q(\alpha)}$, and $Z \in \mathbb{G}_T$, the goal is to distinguish $Z = e(g_0, h_0)^{rq(\alpha)}$ or a random group element in $\mathbb{G}_T$.*

Intractability Analysis of (f, q)-MSE-DDH Problem.

The (f, q)*-MSE-DDH Problem* can be reformulated as D, E, F. Since g_0, h_0 are generators in group $\mathbb{G}$, we suppose $h_0 = g_0^{\beta}$,

$$\begin{aligned} D &= \big(1, \alpha, r, \beta f(\alpha), \cdots, \beta\alpha^{n-2}f(\alpha), \beta f(\alpha)q(\alpha), \cdots, \beta\alpha^n f(\alpha)q(\alpha), \big) \\ E &= 1, \\ F &= r\beta q(\alpha). \end{aligned}$$

We need to show that F is independent of (D, E), i.e. no coefficients $\{x_{i,j}\}$ and y_1 exist such that $F = \Sigma x_{i,j} d_i d_j + \Sigma y_1 e_1$, where the polynomials d_i, d_j are listed in D and e_1 is listed in E above. By making all possible products of two polynomials from D which are multiples of $r\beta$ to F', we want to prove that no such linear combination F' leads to F,

$$F' = \big(r\beta f(\alpha), \cdots, r\beta\alpha^{n-2} f(\alpha), r\beta f(\alpha)q(\alpha), \cdots, r\beta\alpha^{n} f(\alpha)q(\alpha), \big)$$

Any such linear combination associated with $r\beta$ can be written

$$r\beta f(\alpha)A(\alpha) + r\beta f(\alpha)q(\alpha)B(\alpha) = r\beta q(\alpha),$$

where $A(\alpha), B(\alpha)$ are polynomials with degree deg $A \leq n-2$ and deg $B \leq n$.

If $B(\alpha) \neq 0$, we have deg $f(\alpha)q(\alpha)B(\alpha) \geq n$. Since deg $(q(\alpha) - f(\alpha)A(\alpha)) \leq n-1$, we have $B(\alpha) = 0$. We simplify the above equation as $f(\alpha)A(\alpha) = q(\alpha)$, so $f(\alpha)|q(\alpha)$, which contradicts that $f(\alpha)$ and $q(\alpha)$ are comprime. Therefore, there exist no coefficients $\{x_{i,j}\}, y_1$ such that $F = \Sigma x_{i,j} d_i d_j + \Sigma y_1 e_1$ holds, (f, q)-*MSE-DDH Problem* is intractable.

7.2. Security Proof of OKSA

We formally analyze the security of our OKSA protocol, which is under the security model defined in Section 4.2. The security reduction is based on the hard problems defined in Section 7.1. We show the detailed proof process as follows.

Theorem 1. *The proposed scheme satisfies the unconditional keyword privacy of the token from the user under the User Privacy game.*

PROOF. Let W be the authorized keyword set and $(\mathbf{P}(w), \Sigma)$ be generated from $w = w_0$. We have the keyword token and proof as

$$\mathbf{P}(w) = h^{s\prod_{w_j \in W, w_j \neq w_0}(\alpha + w_j)}, \Sigma = \left(\Sigma_1 = h^{\frac{\alpha+w_0}{s}},\ \Sigma_2 = h^{\frac{\alpha^{n-1}(\alpha+w_0)}{s}}\right).$$

For any distinct keyword w_1, let $s' \in \mathbb{Z}_p$, we implicitly set $s' = s \cdot \frac{\alpha+w_1}{\alpha+w_0}$. We find that the keyword tokens are identical, i.e., $\mathbf{P}(w_0) = \mathbf{P}(w_1)$, which can be verified as

$$\mathbf{P}(w_0) = h^{s\prod_{w_j \in W, w_j \neq w_0}(\alpha + w_j)} = h^{s'\prod_{w_j \in W, w_j \neq w_1}(\alpha + w_j)} = \mathbf{P}(w_1).$$

Suppose $\Sigma' = (\Sigma'_1, \Sigma'_2)$. The proofs of accountability are also identical, i.e., $\Sigma_1 = \Sigma'_1$ and $\Sigma_2 = \Sigma'_2$, which can be verified as

$$\Sigma_1 = h^{\frac{\alpha+w_0}{s}} = h^{\frac{\alpha+w_1}{s'}} = \Sigma'_1,\ \Sigma_2 = h^{\frac{\alpha^{n-1}(\alpha+w_0)}{s}} = h^{\frac{\alpha^{n-1}(\alpha+w_1)}{s'}} = \Sigma'_2.$$

We have $(\mathbf{P}(w_0), \Sigma) = (\mathbf{P}(w_1), \Sigma')$. Since s is randomly chosen from $\mathbb{Z}_p$, we have s' is also universally random in $\mathbb{Z}_p$. The distributions of $(\mathbf{P}(w), \Sigma)$ for both w_0 and w_1 are identical and therefore $\mathcal{A}$ has no advantage in guessing the keyword in $\mathbf{P}(w)$. This completes the proof of **Theorem 1**.

Theorem 2. *The proposed scheme is semantically secure and indistinguishable under the Indistinguishability game in the random oracle model if the* (f, q)-MSE-DDH Problem *is hard.*

PROOF. Suppose there exists an adversary $\mathcal{A}$ who can break the indistinguishability. We can construct an algorithm $\mathcal{B}$ that solves the (f, q)-*MSE-DDH Problem*. That is, given an instance of (f, q)-*MSE-DDH Problem* and $Z \in \mathbb{G}_T$, the goal of $\mathcal{B}$ is to distinguish $Z = e(g_0, h_0)^{rq(\alpha)}$ or a random group element in $\mathbb{G}_T$. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Setup. We assume the universal keyword space as $\mathcal{KS} = \{w_1, w_2, \cdots, w_n\}$. $\mathcal{B}$ chooses w_θ from $\mathcal{KS}$ and its corresponding message is denoted as m_θ. It implicitly sets polynomials $f(\alpha) = \alpha + w_\theta$, $q(\alpha) = \prod_{w_j \in \mathcal{KS}, w_j \neq w_\theta}(\alpha + w_j)$. It also sets $g = g_0, h = h_0^{f(\alpha)q(\alpha)}$ and computes $h_i = h_0^{\alpha^i f(\alpha)q(\alpha)}$. The public parameter is denoted as $pp = \left(g_0, h_0^{f(\alpha)q(\alpha)}, \mathcal{PG}\right)$. $\mathcal{B}$ sends the master public key mpk to $\mathcal{A}$, where

$$mpk = (g_0^\alpha, h_1, h_2, \cdots, h_n).$$

H-Query. $\mathcal{B}$ maintains a hash list $L(a_i, X_i, h^i)$, which is initially empty. Upon receiving an H query for (a_i, X_i), if (a_i, X_i) is in the list L, $\mathcal{B}$ returns the corresponding h^i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sets the hash value h^i as follows.

$$h^i = H(a_i, X_i) = \begin{cases} b_0^i, & \text{if } a_i = 0, \\ b_1^i, & \text{if } a_i = 1, \end{cases}$$

where b_0^i, b_1^i are randomly chosen from $\{0,1\}^\ell$. Then $\mathcal{B}$ adds (a_i, X_i, h^i) to the list and returns h^i to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ chooses a keyword set $W \subseteq \mathcal{KS}$, where $|W| \leq n$. When asking for the trapdoor query for a keyword $w_i \in W$, $\mathcal{A}$ randomly chooses $s \in \mathbb{Z}_p$ as the secret key $sk = s$, and sends (w_i, s) to $\mathcal{B}$.

- If $w_i = w_\theta$, abort.
- If $w_i \neq w_\theta$, $\mathcal{B}$ responds $T = h_0^{sq_i(\alpha)f(\alpha)}$ to $\mathcal{A}$, where $q_i(\alpha) = \frac{q(\alpha)}{\alpha + w_i}$. The trapdoor can be verified

  $$T = \mathbf{P}(w_i)^{\frac{1}{\prod_{w_j \in W}(\alpha + w_j)}} = h^{\frac{s\prod_{w_j \in W, j \neq i}(\alpha + w_j)}{\prod_{w_j \in W}(\alpha + w_j)}} = h_0^{\frac{sf(\alpha)q(\alpha)}{\alpha + w_i}} = h_0^{sq_i(\alpha)f(\alpha)}.$$

  It is easy to see that $h_0^{q_i(\alpha)f(\alpha)}$ can be computed from elements $h_0^{f(\alpha)}$, $h_0^{\alpha f(\alpha)}$, $\cdots$, $h_0^{\alpha^{n-2}f(\alpha)}$ in the instance.

Challenge. $\mathcal{A}$ sends two tuples $(m_0, w_0), (m_1, w_1)$ to $\mathcal{B}$ for challenge, where the trapdoor for w_0 or w_1 has not been queried.

- If $w_\theta \notin \{w_0, w_1\}$, abort.

- If $w_\theta \in \{w_0, w_1\}$, $\mathcal{B}$ checks whether $(0, Z)$ and $(1, Z)$ are in the list L. If yes, obtains the corresponding hash value and denotes them as b_0^* and b_1^*. Otherwise, $\mathcal{B}$ chooses $b_0^*, b_1^* \in_R \{0,1\}^\ell$ and sets

$$H(0, Z) = b_0^*,\ H(1, Z) = b_1^*.$$

  Then $\mathcal{B}$ adds $(0, Z, b_0^*)$ and $(1, Z, b_1^*)$ to the list L. $\mathcal{B}$ responds $\mathcal{A}$ with the challenge ciphertext $CT^* = (c_1 = g_0^r, c_2 = b_0^*, c_3 = b_1^* \oplus m_\theta)$.

If $Z = e(g_0, h_0)^{rq(\alpha)}$, one can verify it by implicitly setting $r = r_i f(\alpha)$

$$c_1 = g^{r_i(\alpha + w_\theta)} = g_0^{r_i f(\alpha)} = g_0^r,$$

$$c_2 = H(0, e(g,h)^{r_i}) = H\left(0, e(g_0, h_0)^{rq(\alpha)}\right) = H(0, Z) = b_0^*,$$

$$c_3 = H(1, e(g,h)^{r_i}) \oplus m_\theta = H\left(1, e(g_0, h_0)^{rq(\alpha)}\right) \oplus m_\theta = H(1, Z) \oplus m_\theta = b_1^* \oplus m_\theta.$$

Therefore, CT^* is a valid challenge ciphertext.

If Z is a random element in $\mathbb{G}_T$, the challenge ciphertext CT^* will be random from the $\mathcal{A}$'s view.

Phase 2. $\mathcal{A}$ continues to ask trapdoor queries for w_i with restrictions $w_i \neq w_0, w_1$. $\mathcal{B}$ responds as **Phase 1**.

Guess. $\mathcal{A}$ outputs a guess θ' of θ.

This completes the description of our simulation. We will analyze the advantage of $\mathcal{B}$ to solve the hard problem. Suppose that the total number of trapdoor query is q_T and the size of the keyword space is n. According to the above simulation, we have the probability that $\mathcal{B}$ does not abort is $\Pr[\neg\text{abort}] = (1 - \frac{1}{n})(1 - \frac{1}{n-1}) \cdots (1 - \frac{1}{n-q_T+1}) = \frac{n-q_T}{n}$. Since w_0, w_1 have not been queried in Phase 1 and Phase 2, we have $q_T \leq n-2$. Then $\Pr[\neg\text{abort}] \geq \frac{2}{n}$. Assume that $\mathcal{A}$'s advantage to break the security game is at least ϵ, then we have $\epsilon_{reduction} = \Pr[\theta' = \theta | Z = e(g_0, h_0)^{rq(\alpha)}] - \Pr[\theta' = \theta | Z \text{ is random}] = \frac{1}{2} + \epsilon - \frac{1}{2} = \epsilon$. Therefore, $\mathcal{B}$'s advantage to solve the (f, q)-*MSE-DDH Problem* is at least $\epsilon_{\mathcal{B}} = \Pr[\neg abort] \cdot \epsilon_{reduction} = \frac{2}{n}\epsilon$.This completes the proof of **Theorem 2**.

Theorem 3. *The proposed scheme captures the accountability under the Accountability game if the* (f, n)-DHE Problem *is hard.*

PROOF. Suppose there exists an adversary $\mathcal{A}$ who can break the security of accountability. We construct an algorithm $\mathcal{B}$ that solves the (f, n)-*DHE Problem*. Given a challenge instance of (f, n)-*DHE Problem*, $\mathcal{B}$ interacts with the adversary as the follows.

Setup. $\mathcal{B}$ sets $\alpha = a$, we have $h_1 = h^a, \cdots, h_n = h^{a^n}$, which are from the (f, n)-*DHE* instance. $\mathcal{B}$ chooses a hash function H as in the real scheme and the public

parameters can be denoted as $pp = (\mathcal{PG}, H, g, h)$. $\mathcal{B}$ sends mpk to $\mathcal{A}$, where

$$mpk = (g^a, h_1, h_2, \cdots, h_n).$$

Challenge. The adversary chooses two keyword sets W, W' with restriction $|W'| > 1, |W| \leq n$, and selects a random number $s \in \mathbb{Z}_p$ as the secret key of the user $sk = s$. $\mathcal{A}$ outputs $(\mathbf{P}(W'), W, W', sk)$ and 1 for challenge, where the token is denoted as

$$\mathbf{P}(W') = h^{s \prod_{w_j \in W - W'}(a+w_j)}.$$

Win. The adversary $\mathcal{A}$ outputs $(\mathbf{P}(W'), \Sigma)$ and wins the game if $(\mathbf{P}(W'), \Sigma)$ passes the verification algorithm.

In this case, the proof for accountability should be denoted as

$$\Sigma = \left(\Sigma_1 = h^{\frac{1}{s}\prod_{w_j \in W'}(a+w_j)},\ \Sigma_2 = \Sigma_1^{a^{n-1}} = h^{\frac{1}{s}a^{n-1}\prod_{w_j \in W'}(a+w_j)}\right).$$

Then the token and its proof can pass the verification as

$$e\left(\Sigma_2, h^a\right) = e\left(\Sigma_1, h^{a^n}\right),\ e\left(h, h^{\prod_{w_i \in W}(a+w_i)}\right) = e\left(\mathbf{P}(W'), \Sigma_1\right).$$

Let $f(x) = \frac{1}{s}x^{n-1}\prod_{w_j \in W'}(x + w_j)$, then $\Sigma_2 = h^{f(a)}$. We have $f(x)$ is a polynomial function with deg $f(x) > n$. $\mathcal{B}$ outputs $(f(x), \Sigma_2)$ as the solution to the (f, n)-*DHE Problem*. This completes the proof of **Theorem 3**. Hence we obtain $|W'| = 1, |\mathbf{P}(W')| = 1$.

7.3. Security Analysis of Searchain

According to the proposed Searchain protocol, Node A shares data with Node B without relying on a third-party. We assume that Node B specifies the keyword w_i and retrieves the data m_i associated with w_i and the block σ' is verified by more than half of nodes. In the current transaction, all parameters for data retrieval is set between Node A and Node B. The security of the proposed Searchain protocol can be directly obtained from the security of OKSA and the security of OMS.

Firstly, *Indistinguishability* of OKSA guarantees secrecy protection of m_i and w_i in Searchain, which cannot be obtained without the corresponding secret key of the user. Secondly, *Privacy* and *Accountability* of OKSA provides oblivious retrieval of Searchain. In Searchain, Node A is able to know the relationship $w_i \in W$ but not to know what is the specific keyword w_i in an oblivious way. As well as, Node A learns nothing about the retrieved plain-cipher data (m_i, CT_i). Thirdly, *Unforgeability* and *Ordering* of OMS [42] guarantee the impossibility to re-order the positions of blocks in the chain.

Acknowledgments.

This work is supported by NSFC (Grant Nos. 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

8. Conclusion

Motivated by the privacy concern of the data retrieval in the decentralized storage, we proposed Searchain, a blockchain-based keyword search mechanism that aims to assure private search over authorized keywords with unchanged retrieval order. The core design of Searchain is oblivious keyword search with authorization (OKSA), which supports keyword authorization. Searchain adopts OKSA to build the retrieval protocol so that the node who owns the data can verify the authorization of the keyword in the retrieval request but learns nothing about it. By further employing ordered multisignatures (OMS) into block generation, Searchain remains an ordered retrieval transaction. We evaluated Searchain by algorithm implementations, where the results showed its cost-efficiency.

References

[1] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, D. X. Song, Provable data possession at untrusted stores, in: CCS 2007, ACM, 2007, pp. 598–609.

[2] F. Armknecht, J. Bohli, G. O. Karame, F. Youssef, Transparent data deduplication in the cloud, in: CCS 2015, ACM, 2015, pp. 886–900.

[3] C. Qin, J. Li, P. P. C. Lee, The design and implementation of a rekeying-aware encrypted deduplication storage system, TOS 13 (1) (2017) 9:1–9:30.

[4] J. Li, C. Qin, P. P. C. Lee, X. Zhang, Information leakage in encrypted deduplication via frequency analysis, in: DSN 2017, Vol. To appear, IEEE Computer Society, 2017.

[5] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, `https://bitcoin.org/bitcoin.pdf` (2009).

[6] T. McConaghy, R. Marques, A. Müller, D. de Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, A. Granzotto, BigchainDB: A scalable blockchain database, `https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf` (2016).

[7] M. T. Özsu, P. Valduriez, Principles of Distributed Database Systems, 3rd Edition, Springer, 2011.

[8] D. X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: S&P 2000, IEEE Computer Society, 2000, pp. 44–55.

[9] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, M. Steiner, Highly-scalable searchable symmetric encryption with support for boolean queries, in: CRYPTO 2013, Vol. 8042 of LNCS, Springer, 2013, pp. 353–373.

[10] S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, M. Steiner, Outsourced symmetric private information retrieval, in: CCS 2013, ACM, 2013, pp. 875–888.

[11] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: EUROCRYPT 2004, Vol. 3027 of LNCS, Springer, 2004, pp. 506–522.

[12] C. Fan, V. S. Huang, Provably secure integrated on/off-line electronic cash for flexible and efficient payment, IEEE Trans. Systems, Man, and Cybernetics, Part C 40 (5) (2010) 567–579.

[13] W. Ogata, K. Kurosawa, Oblivious keyword search, J. Complexity 20 (2-3) (2004) 356–371.

[14] P. Jiang, X. Wang, J. Lai, F. Guo, R. Chen, Oblivious keyword search with authorization, in: ProvSec 2016, Vol. 10005 of LNCS, Springer, 2016, pp. 173–190.

[15] A. E. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: S&P 2016, IEEE Computer Society, 2016, pp. 839–858.

[16] M. Crosby, Nachiappan, P. Pattanayak, S. Verma, V. Kalyanaraman, Blockchain technology, Tech. rep., Sutardja Center, Berkeley, University of California (2015).

[17] Proof of existence, `https://proofofexistence.com/`.

[18] Filament, `https://filament.com/`.

[19] Storj, `https://storj.io/`.

[20] G. Zyskind, O. Nathan, A. Pentland, Decentralizing privacy: Using blockchain to protect personal data, in: SPW 2015, IEEE Computer Society, 2015, pp. 180–184.

[21] K. Peterson, R. Deeduvanu, P. Kanjamala, K. Boles, A blockchain-based approach to health information exchange networks, `https://www.healthit.gov/sites/default/files/12-55-blockchain-based-approach-final.pdf` (2016).

[22] E. Ryu, T. Takagi, Efficient conjunctive keyword-searchable encryption, in: AINA 2007, Vol. 1, IEEE Computer Society, 2007, pp. 409–414.

[23] J. Bethencourt, D. X. Song, B. Waters, New constructions and practical applications for private stream searching (extended abstract), in: S&P 2006, IEEE Computer Society, 2006, pp. 132–139.

[24] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: TCC 2007, Vol. 4392 of LNCS, Springer, 2007, pp. 535–554.

[25] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, H. Shi, Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions, in: CRYPTO 2005, Vol. 3621 of LNCS, Springer, 2005, pp. 205–222.

[26] H. S. Rhee, J. H. Park, W. Susilo, D. H. Lee, Improved searchable public key encryption with designated tester, in: ASIACCS 2009, IEEE Computer Society, 2009, pp. 376–379.

[27] P. Jiang, Y. Mu, F. Guo, X. Wang, Q. Wen, Online/offline ciphertext retrieval on resource constrained devices, Comput. J. 59 (7) (2016) 955–969.

[28] J. Camenisch, M. Kohlweiss, A. Rial, C. Sheedy, Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data, in: PKC 2009, Vol. 5443 of LNCS, Springer, 2009, pp. 196–214.

[29] M. O. Rabin, How to exchange secrets with oblivious transfer, Tech. Rep. Technical Report TR-81, Aiken Computation Laboratory, Harvard University (2005).

[30] C. Chu, W. Tzeng, Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries, in: PKC 2005, Vol. 3386 of LNCS, Springer, 2005, pp. 172–183.

[31] K. Kurosawa, R. Nojima, Simple adaptive oblivious transfer without random oracle, in: ASIACRYPT 2009, Vol. 5912 of LNCS, Springer, 2009, pp. 334–346.

[32] J. Camenisch, G. Neven, A. Shelat, Simulatable adaptive oblivious transfer, in: EUROCRYPT 2007, Vol. 4515 of LNCS, Springer, 2007, pp. 573–590.

[33] M. Green, S. Hohenberger, Universally composable adaptive oblivious transfer, in: ASIACRYPT 2008, Vol. 5350 of LNCS, Springer, 2008, pp. 179–197.

[34] J. Camenisch, M. Dubovitskaya, G. Neven, Oblivious transfer with access control, in: CCS 2009, ACM, 2009, pp. 131–140.

[35] W. Aiello, Y. Ishai, O. Reingold, Priced oblivious transfer: How to sell digital goods, in: EUROCRYPT 2001, Vol. 2045 of LNCS, Springer, 2001, pp. 119–135.

[36] J. Camenisch, M. Dubovitskaya, G. Neven, Unlinkable priced oblivious transfer with rechargeable wallets, in: FC 2010, Vol. 6052 of LNCS, Springer, 2010, pp. 66–81.

[37] Y. Chen, J. Chou, X. Hou, A novel k-out-of-n oblivious transfer protocols based on bilinear pairings, IACR Cryptology ePrint Archive 2010 (2010) 27.

[38] F. Guo, Y. Mu, W. Susilo, Subset membership encryption and its applications to oblivious transfer, IEEE Trans. Information Forensics and Security 9 (7) (2014) 1098–1107.

[39] H. S. Rhee, J. W. Byun, D. H. Lee, J. Lim, Oblivious conjunctive keyword search, in: WISA 2005, Vol. 3786 of LNCS, Springer, 2005, pp. 318–327.

[40] M. J. Freedman, Y. Ishai, B. Pinkas, O. Reingold, Keyword search and oblivious pseudorandom functions, in: TCC 2005, Vol. 3378 of LNCS, Springer, 2005, pp. 303–324.

[41] H. Zhu, F. Bao, Oblivious keyword search protocols in the public database model, in: ICC 2007, IEEE, 2007, pp. 1336–1341.

[42] A. Boldyreva, C. Gentry, A. O'Neill, D. H. Yum, Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing, in: CCS 2007, ACM, 2007, pp. 276–285.

[43] G. Neven, Efficient sequential aggregate signed data, in: EUROCRYPT 2008, Vol. 4965 of LNCS, Springer, 2008, pp. 52–69.

[44] J. H. Ahn, M. Green, S. Hohenberger, Synchronized aggregate signatures: new definitions, constructions and applications, in: CCS 2010, ACM, 2010, pp. 473–484.

[45] N. Yanai, M. Mambo, E. Okamoto, An ordered multisignature scheme under the CDH assumption without random oracles, in: ISC 2013, Vol. 7807 of LNCS, Springer, 2013, pp. 367–377.

[46] D. Boneh, X. Boyen, E. Goh, Hierarchical identity based encryption with constant size ciphertext, in: EUROCRYPT 2005, Vol. 3494 of LNCS, Springer, 2005, pp. 440–456.

[47] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo, L. Zhou, On emerging family of elliptic curves to secure internet of things: ECC comes of age, IEEE Transactions on Dependable and Secure Computing 14 (3) (2017) 237–248.

[48] Z. Liu, J. Groschdl, Z. Hu, K. Jrvinen, H. Wang, I. Verbauwhede, Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the internet of things, IEEE Transactions on Computers 66 (5) (2017) 773–785.

[49] F. Guo, Y. Mu, W. Susilo, V. Varadharajan, ACISP 2013, Vol. 7959 of LNCS, Springer, 2013, pp. 219–234.

[50] F. Guo, Y. Mu, W. Susilo, V. Varadharajan, `http://www.uow.edu.au/~fuchun/publications/ACISP13.pdf`, full Version (2013).