Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5340116, 21 pages
https://doi.org/10.1155/2021/5340116

Research Article

BFR-SE: A Blockchain-Based Fair and Reliable Searchable Encryption Scheme for IoT with Fine-Grained Access Control in Cloud Environment

Hongmin Gao,[1] **Shoushan Luo**,[1] **Zhaofeng Ma**,[1] **Xiaodan Yan**,[2] **and Yanping Xu**[3]

[1]*Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Beihang University, Beijing 100191, China*
[3]*School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, Zhejiang Province 310018, China*

Correspondence should be addressed to Xiaodan Yan; yanxiaodan@buaa.edu.cn

Received 29 July 2021; Revised 1 September 2021; Accepted 29 October 2021; Published 24 November 2021

Academic Editor: Yan Huang

Due to capacity limitations, large amounts of data generated by IoT devices are often stored on cloud servers. These data are usually encrypted to prevent the disclosure, which significantly affects the availability of this data. Searchable encryption (SE) allows a party to store his data created by his IoT devices or mobile in encryption on the cloud server to protect his privacy while retaining his ability to search for data. However, the general SE techniques are all pay-then-use. The searchable encryption service providers (SESP) are considered curious but honest, making it unfair and unreliable. To address these problems, we combined ciphertext-policy attribute-based encryption, Bloom filter, and blockchain to propose a blockchain-based fair and reliable searchable encryption scheme (BFR-SE) in this paper. In BFR-SE, we constructed an attribute-based searchable encryption model that can provide fine-grained access control. The data owner stores the indices on SESP and stores some additional auxiliary information on the blockchain. After a data user initiates a request, SESP must return the correct and integral search results before the deadline. Otherwise, the data user can send an arbitration request, and the blockchain will make a ruling. The blockchain will only perform arbitrations based on auxiliary information when disputes arise, saving the computing resources on-chain. We analyzed the security and privacy of BFR-SE and simulated our scheme on the EOS blockchain, which proves that BFR-SE is feasible. Meanwhile, we provided a thorough analysis of storage and computing overhead, proving that BFR-SE is practical and has good performance.

1. Introduction

With the continuous development of Mobile Internet, 5G, and some other advanced technologies, especially the Internet of Things, people and machines are always generating massive amounts of data. Most IoT devices produce large amounts of data with a limited storage capacity, so the data owners like to use cloud storage services to reduce the burden of maintenance costs and local storage overhead. Cloud services provide users with great convenience, enabling users to access their data anytime and anywhere, instead of using a specific machine. But these data, especially the data generated by specific IoT devices such as smart homes and intelligent wristbands, often contains sensitive information to the user. To prevent the disclosure, users encrypt their data before uploading it to the cloud server [1–8]. However, encryption will weaken the ability of users to search for data.

Searchable encryption technology was first proposed by Song et al. [9], which allows a party to store his data in encryption on the cloud server to protect his privacy while retaining his ability to search for data. A searchable encryption scheme typically includes three participants: the data owner (DO), the data user (DU), and the cloud server. The DO encrypts his data together with the corresponding keywords and uploads them to the cloud server. The cloud server maintains these ciphertexts and provides search services for data users. A data user will initiate a search request using a search token generated based on the keywords, and

the matching search results will be sent to him by the cloud server. Finally, the data user can decrypt the ciphertext locally to obtain the data. The whole process will not expose any information related to the data itself. Nowadays, many researchers have proposed various searchable encryption algorithms, such as asymmetric searchable encryption [10, 11], multikeyword searchable encryption [12, 13], and fuzzy keyword searchable encryption [14, 15]. Most of the above studies focus on searchable encryption's privacy and performance in different scenarios and assume that the cloud server is curious but honest. However, this is not the case, which will cause problems in the fairness and reliability of searchable encryption:

(1) On the one hand, after the user pays, the cloud server cannot provide satisfactory search services, resulting in the user's economic losses. On the other hand, after the user obtains the desired search results, he will slander and deny the cloud server's service and deceptively refuse to pay the service fee

(2) The cloud server is not always reliable. To save costs, it may delete data that is not often used at ordinary times to save space. When users search, it will send part of the search results or even send fake data to users

According to the above point of view, in addition to the privacy of keywords and search algorithms' efficiency, practical searchable encryption is highly expected to be fair and reliable. To solve these problems, we urgently need such a searchable encryption scheme, in which the service provider is always to provide reliable search service, and the users pay for it. There is no credible third party in this scheme but will not cause any economic disputes. Fortunately, with the emergence and development of Bitcoin [16], as a decentralized cryptocurrency, its underlying technology blockchain can gracefully help us to achieve this goal. In this paper, we proposed a fair and reliable searchable encryption scheme (BFR-SE) based on blockchain. The main contributions of our research are as follows:

(1) We constructed an attribute-based searchable encryption algorithm (ABSE) and combined it with blockchain and Bloom filter to propose a fair and reliable searchable encryption scheme. While the DO stores the data indices in the SESP, some additional auxiliary information used for verification is uploaded to the blockchain. In the event of disputes between DU and SESP, the blockchain will arbitrate, and the dishonest participant will be punished financially

(2) BFR-SE supports users' multikeyword search for ciphertexts. By utilizing ABSE, the DO realizes fine-grained access control for their data search, which means that only the users whose attributes satisfy the specific policy can search and obtain the correct search results

(3) Not the same as other blockchain-based searchable encryption schemes, BFR-SE only stores a small amount of auxiliary information on-chain and performs possible arbitration when disputes occur, which dramatically saves storage and computing resources on-chain

(4) We simulated and implemented BFR-SE on the EOS blockchain and showed implementation details of smart contracts and algorithms. Together with the security analysis, it proves that our scheme is feasible

(5) We used 6 MacBook Pros to build an EOS private chain in a laboratory environment and simulated our scheme. The storage and computing overhead proves that BFR-SE is practical and has good performance

The rest of this paper is organized as follows: Section 2 consists of related works. Section 3 reviews some preliminary knowledge used throughout this paper. In section 4, we have an overview of our scheme. Section 5 describes specific implementation details. In Section 6, we analyze the security and performance. Finally, we present the conclusion and future direction.

2. Related Work

2.1. Verifiable Searchable Encryption. To ensure the reliability of searchable encryption and prevent the cloud server from returning partial or even wrong search results, users need to have the ability to verify the correctness of search results. Earliest in 2012, Chai and Gong [17] proposed a verifiable keyword search scheme, in which the cloud server needs to prove that the returned results are correct. Kurosawa and Ohtaki [18] proposed the first UC-secure verifiable symmetric searchable encryption, which can verify whether the search results are modified or deleted, and the computational cost of verification has a linear relationship with the number of files. Zhu et al. [19] constructed a verifiable fuzzy keyword search scheme to support dynamic data using Bloom filter and locality sensitive hash function. The single-keyword verifiable searchable encryption will return many irrelevant results that cause the waste of transmission bandwidth and computing resources, so the verifiable searchable encryption proposed by Azraoui et al. [20] supports multikeyword search or combined search. However, the above verifiable searchable encryption schemes are only suitable for a small number of users, and it is challenging to meet the user's dynamic requirements in the cloud environment. The number of users growing will cause the burden of key management and cannot achieve fine-grained access control. In 2014, Zheng et al. [21] proposed a novel cryptographic primitive named verifiable attribute-based keyword search. This primitive allows DO to control the search and outsource his encrypted data to the cloud server based on an access policy. Simultaneously, it allows legitimate users to outsource the search operation (usually expensive) to the cloud server and verify whether the cloud server loyalty performs it. Ameri et al. [22] combined hierarchical

identity-based multidesignated verifier signature (HIB-MDVS), hierarchical identity-based broadcast encryption (HIBBE) and Bloom filter to propose a generic construction for verifiable attribute-based keyword search. The VBKS scheme proposed by Sun et al. [23] can realize the revocation of user attributes and utilize proxy reencryption and lazy reencryption to transfer the heavy update work during attribute revocation to a semitrusted cloud server while supporting the multikeyword search.

As mentioned above on verifiable searchable encryption, the research enables users to verify the search results' correctness and integrity. However, as a more practical searchable encryption scheme, this is far from enough because when a dishonest server is detected, it cannot continue to punish the dishonest server without a third-party trusted organization, which cannot be genuinely reliable.

2.2. Blockchain-Based Searchable Encryption. In recent years, some researchers utilized blockchain to solve the fairness problem in searchable encryption. In 2017, Li et al. [24] used blockchain to construct symmetric searchable encryption (SSE-using-BC). In their scheme, users publicly store all data on the Bitcoin through transactions. As long as the participant does not execute honestly, he will lose his BTC. In their subsequent work [25], they also improved the scheme and adopted the Fabric blockchain, which significantly improved the performance. Hu et al. [26] explored the potential capabilities of the Ethereum blockchain and constructed a decentralized, privacy-protected search model. The scheme designed a financially fair smart contract to replace the centralized server so that all participants are treated equally and motivated to perform correct operations. Cai et al. [27] also used a smart contract to record encrypted search records on the blockchain and designed a fair protocol to deal with disputes and payment issues. They used a dynamic, efficient searchable encryption scheme, which retained the search capability and inspired the service provider to make a real effort. Tang [28] extended the original searchable encryption, storing some necessary information on-chain, in which blockchain only serves as a proper judicial function. If there is no dispute, it will perform little operations on-chain, reducing the blockchain's burden. Chen et al. [29] stored the indices and complex logical structure of EHRs on the blockchain. They believed that only utilizing blockchain for propagation can the data owner have complete control over their data. Blockchain ensures the integrity, unforgeability, and traceability of the indices. Jiang et al. [30] proposed a Bloom filter-enabled multikeyword search protocol with enhanced efficiency and privacy preservation. In the protocol, a low-frequency keyword is selected using the Bloom filter to filter the database when performing a multikeyword search.

In summary, although the above searchable encryption based on the blockchain can solve the fairness problem in the payment process, there are still some shortcomings:

(1) These schemes are products of the combination of blockchain and symmetric searchable encryption that can only achieve a single one-to-one scenario that is difficult for a large number of users and meeting dynamic requirements in a cloud environment, not to mention fine-grained access control

(2) The main idea of these schemes is to store index information of the encrypted data on-chain. Although encrypted, the symmetric searchable encryption is generally a deterministic function. It will be noticed when the user searches for the same keyword multiple times. It will lead to the establishment of some statistics, making it possible to infer some private information

(3) Both file storage and search algorithm execution are all processed on-chain, which increases the storage and computing overhead of blockchain. Compared with the traditional way, because the blockchain requires parallel storage and calculation of multiple miners, resource waste is bound to become noticeable. Individual schemes put this part off-chain but caused functional defects, such as the participants need a lot of offline communications

3. Preliminary

3.1. Bilinear Pairing. Let G_0 and G_1 be cyclic groups of order p and g be a generator of G_0. We call $e : G_0 \times G_0 \longrightarrow G_1$ is a bilinear paring if it is a map with the following properties:

(1) Bilinear: for all $g_1, g_2 \in G_0$ and $a, b \in Z_p$, there will be $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$

(2) Nondegenerate: there exists $g_0 \in G_0$, such that $e(g_0, g_0) \neq 1$

(3) Computable: there is an efficient algorithm to compute $e(g_1, g_2)$ for all g_1 and g_2

3.2. Linear Secret Sharing Scheme (LSSS). Let $P = \{P_1, \cdots, P_n\}$ be a set of participants, and (A, ρ) be an access structure. In the structure, A is an $l \times k$ matrix with ρ mapping its rows. An LSSS is composed of two polynomial-time algorithms:

(1) share$((A, \rho), s)$: to share a secret value s, it selects randomly $v_1, \cdots, v_{k-1} \in Z_p$. Let $\vec{\boldsymbol{v}} = (s, v_1, \cdots, v_{k-1})^T$, and A_i be the vector as the ith row of matrix A, then, the secret share $\sigma_i = A_i \vec{\boldsymbol{v}}$ belongs to party $\rho(i)$

(2) recover$(\omega, \{\sigma_i\}_{i \in \omega})$: it takes $\omega \in A$ and corresponding secret shares as inputs. If any $L \in \{i \mid \rho(i) \in \omega\}$ satisfies the access structure, a set of recovery coefficients $\{\mu_i\}_{i \in L}$ can be calculated so that $\sum_{i \in L} u_i \sigma_i = s$

3.3. Bloom Filter. Bloom filter is a space-efficient probabilistic data structure, proposed by Burton Howard Bloom in 1970 [31] that through an individual error rate in exchange for space-saving and query efficiency. A standard l_{BF}-bit Bloom filter includes a vector $\mathbf{V}$ with a length of l_{BF}, all bits of which are initialized to 0, k independent hash functions

$\{h_1, \cdots, h_k\}$, and each hash function has a uniform value in the range of $[0, l_{\mathrm{BF}} - 1]$. For each element $w_i (1 \le i \le n)$ in set $\mathbf{W} = \{w_1, \cdots, w_n\}$, set the corresponding position of $H_j(w_i)(1 \le j \le k)$ in the vector to 1. It is only necessary to determine whether an element all $H_j(w_i)$ in $\mathbf{V}$ are 1 or not to judge whether an element w is in the set $\mathbf{W}$. If not, there must be $w \notin W$; otherwise, there is a high probability $w \in \mathbf{W}$. (It should be noted that a high probability means that there is a false-positive rate of nonzero, but this possibility can be minimized by appropriate setting the value of l_{BF} and k.) A Bloom filter is composed of two algorithms:

(1) $\mathrm{BFGen}(\{h_1', \cdots, h_k'\}, \{w_1, \cdots, w_n\}) \longrightarrow \mathrm{BF}$: this algorithm generates an l_{BF}-bit Bloom filter by hashing $\{h_1, \cdots, h_k\}$ the data set $\mathbf{W} = \{w_1, \cdots, w_n\}$

(2) $\mathrm{BFVerify}(\mathrm{BF}, w, \{h_1', \cdots, h_k'\}) \longrightarrow (0, 1)$: this algorithm verifies whether the element w is in the set $\mathbf{W}$. If it returns 1, $w \in W$. Otherwise, $w \notin W$

3.4. Blockchain. The blockchain concept originated from Nakamoto's Bitcoin white paper [16], whose foundation is cryptography and P2P networks. Then, it organizes the data with a specific structure into blocks in a certain way and links these blocks into a chain in chronological order. Cryptography and consensus mechanisms together ensure the security and unforgeability of data. In short, as the underlying technology of cryptocurrency like Bitcoin, the blockchain is a trusted ledger with distributed computing capabilities that can process business credibly without a third-party organization.

3.5. Smart Contract. Initially, when it comes to blockchain, the only well-known applications are cryptocurrencies such as Bitcoin and Litecoin. What brings a qualitative change to the blockchain is that in 2013 Vitalik Buterin established the first public chain platform named Ethereum with a built-in Turing complete language [32] and inaugurated smart contract for the blockchain. Szabo defined smart contract as "a computerized transaction protocol that executes the terms of a contract" [33]. The smart contract in the blockchain is a piece of program code stored on the chain, which can be executed securely and reliably. On the one hand, the blockchain can utilize a programmable smart contract to implement more complex business logic. On the other hand, the blockchain can provide a trusted environment for executing a smart contract. The operating mechanism of the smart contract in the blockchain is shown in Figure 1.

As shown in the figure, the blockchain can be seen as a state machine triggered by transactions, and the ledger is a public world state starting from the Genesis Block. Users can create a transaction and broadcast it to the blockchain network from any node. All block producers will perform corresponding operations after receiving the transaction, and the consensus mechanism makes all nodes finally get a consistent result and update the world state.

Blockchain provides the following support for the execution of smart contract on-chain:

(i) Public status: every participant can inspect the smart contract's current world state on the public ledger

(ii) Timestamp server: the block height can be seen as a trusted timestamp that never stops

(iii) Trusted propagation channel: the sender can utilize the blockchain to spread the message, and the receiver will reliably receive the message shortly. The delivery traces will be recorded on-chain for auditing, and the records are credible and cannot be tampered with by anyone

3.6. Transactions of EOS. Account, address, and transaction are three essential components in the EOS blockchain [34]. Each user has an account that corresponds to multiple ECDSA key pairs expressed as (pk, sk), and each key pair represents different operation permission of the account. The private and public keys are used by users to sign and verify a transaction. Our definition of a transaction is consistent with our previous work [35, 36]:

$$\begin{aligned} \mathrm{Tx} = (&\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_U(\mathrm{Chain\_ID}, \mathrm{Tx})), \\ &\mathrm{Action}(\mathrm{Code}, \mathrm{Name}, \mathrm{Auth}_U, \mathrm{Data})), \end{aligned} \tag{1}$$

where $\mathrm{Ref}_{\mathrm{block}}$ refers to the height and id of a recently generated block, which prevents the transaction from being packaged on the fork chain and t is the expiration time of the transaction. $\mathrm{Sig}_U(\mathrm{Chain\_ID}, \mathrm{Tx})$ is the signature signed by the sender. Action is the operation performed by the transaction in which Code is the name of the smart contract, Name is the method to be invoked in the smart contract, Auth_U is used to verify whether the sender has the authority to call the method, and Data are the parameters. There may be multiple actions in one transaction. Smart contracts can also send actions to each other to call methods of other contracts, which is called inline communication, and the corresponding execution authority is the same as the original transaction.

3.7. Data Persistence of EOS. After the smart contract is executed, the occupied memory will be released, and all variable data in the program will be lost, so it is necessary to persist the data in smart contract. In the smart contact of Ethereum, data can only be stored in key-value pairs, which is difficult to meet more complex requirements. EOS imitates multi-index containers in Boost library and develops a C++ class: *eosio*::*multi_index* (from now on referred to as multi_index). Each multi_index can be regarded as a table in the traditional database. Each row of the table can store an object, and the object's attributes can be any C++ data type. Therefore, the table constructed by multi_index in EOS is no less flexible than traditional databases. A significant feature of multi_index is that a primary key can be set as the main index and 16 secondary indices. Users can obtain any of these indices and use the *emplace*, *erase*, *modify*, and *find* functions of the index to insert, delete, update, and select data.

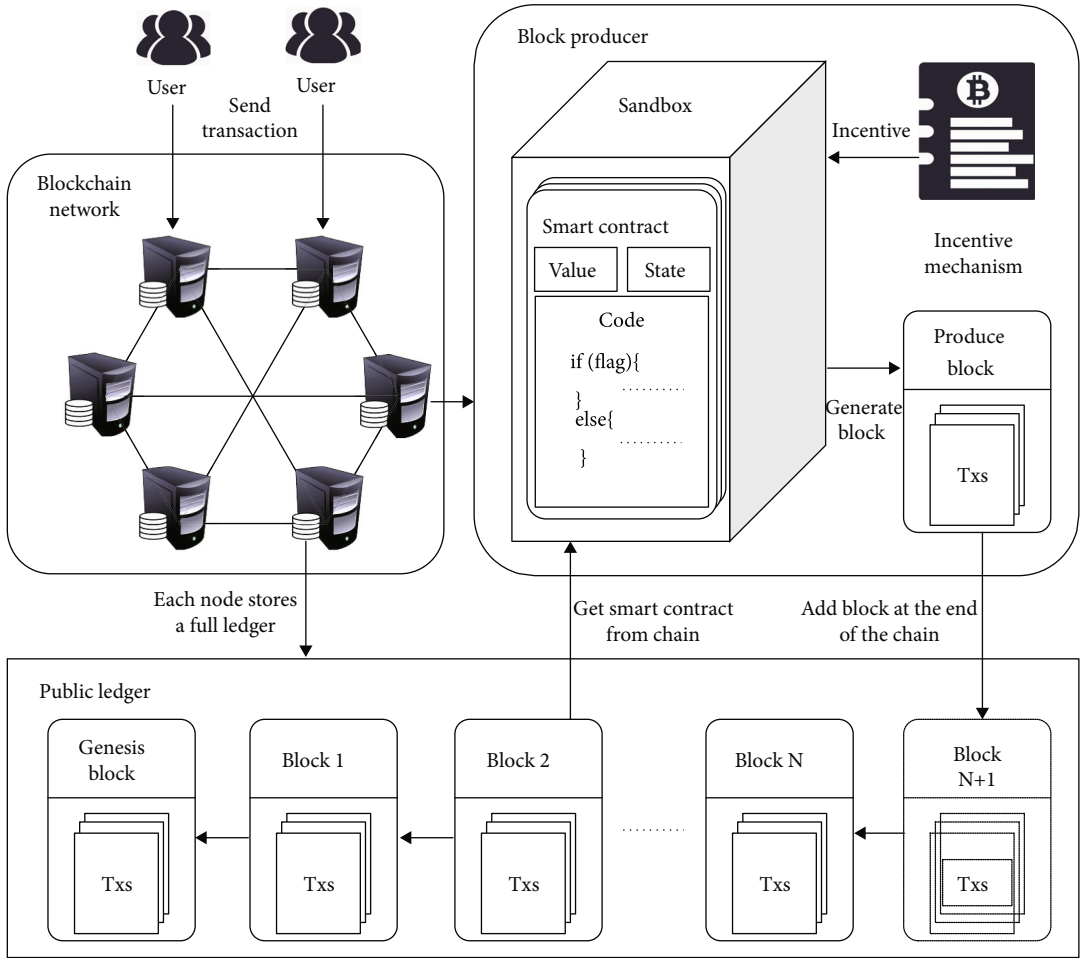

Figure 1: Operation mechanism of blockchain-based smart contract.

4. Overview of Proposed Scheme

This section will give an overview of our proposed scheme, including the system model and scheme design. The meanings of the symbols and abbreviations used in this paper are shown in Table 1.

4.1. System Model of BFR-SE. The scheme proposed in this paper is composed of four components: data owner (DO), data user (DU), searchable encryption service provider (SESP), and blockchain. The keywords and their corresponding index structure are encrypted and uploaded to the SESP after DO extracts the keyword set from the outsourced data set to prevent privacy disclosure. DO distribute keys for DUs through blockchain, and only the DUs whose attributes satisfy the access policy can search and obtain the original data. DU uses his private key to generate a search token according to the keywords he wants to retrieve. According to the search token provided by DU, SESP performs complex search calculation operations, then returns the search results to DU and obtains the revenue. The traces and additional evidence of each participant will be recorded on the blockchain, which cannot be destroyed or denied. DUs need to pay SESP for the service they use. If the SESP does not provide the correct result before the predetermined block height, DU can apply for arbitration, and the blockchain will make a judgment according to the auxiliary information and the additional evidence on-chain during the search. Then, the charge fee will be returned to the DU as compensation, together with a penalty on SESP. The specific functions and responsibilities of the four components are as follows:

(1) DO: the owner of the IoT device is also the owner of the data. Responsible for the system's initialization, including creating and deploying smart contracts in the scheme. DO needs to generate and distribute private keys for registered DUs according to their attributes. Besides, DO extracts keywords from the outsourced data files, generates the corresponding indices, and sets a reasonable access policy for the indices. DO is honest by default

(2) DU: according to the keywords he wants to retrieve, DU generates a search token with his private key and sends a search request to the SESP. At the same time, DU can judge whether the search results returned by SESP are correct or not

Table 1: The symbols and abbreviations involved in this paper.

No.	Symbol	Description
1	DO	The data owner
2	DU	The data user
3	SESP	The searchable encryption service provider
4	MSK	System master key
5	PK	System public parameters
6	$\varepsilon = (\varepsilon.\mathrm{Enc}, \varepsilon.\mathrm{Dec})$	An asymmetric encryption algorithm like ECC
7	$(Pk_{\mathrm{com}}, Sk_{\mathrm{com}})$	A pair of keys for algorithm ε
8	$(Pk_{\mathrm{sig}}, Sk_{\mathrm{sig}})$	A pair of keys for an algorithm like ECDSA
9	S	All general attribute set
10	ω	The attributes set of a specific DU
11	$\mathscr{P}$	Access policy
12	Sk_{ω}	The private attribute key of DU whose attributes set is ω
13	FID	The set of file identities
14	KW	The set of keywords
15	CKW	The set of ciphertext for keywords
16	I	The set of indices between keywords and data files
17	CI	The set of ciphertext for indices
18	AI	Auxiliary information
19	TOK	Search token
20	COMM	DO's commitment to his search request

(3) SESP: it has powerful computing capabilities and stores the ciphertext of indices provided by DO. It will perform complex search calculations on the indices according to DU's search token, then return the search results to DU. SESP may be malicious and may return partial or even wrong search results to save resources and defraud profits

(4) Blockchain: it stores the auxiliary information of the indices and intermediate evidence information. In the event of a dispute, arbitration can be conducted according to this information, and the malicious parties can be punished economically. In the absence of a third-party authoritative and trusted organization, the blockchain is the cornerstone of trust in the scheme. Additionally, blockchain can provide a reliable broadcast channel for each participant, which can be used by each party for information dissemination

The system model of our proposed scheme is shown in Figure 2.

Our scheme's searchable encryption algorithm is inspired by the scheme named VABKS (verifiable attribute-based keyword search) proposed by Zheng et al. [21]. We have optimized and extended it to support a multi-keyword search. The detailed description of each step in the flowchart is as follows:

(i) DO creates and deploys smart contracts on the blockchain. BFR-SE includes two smart contracts: PMContract and SEContract

(ii) DO generates the system master key and public parameters, as well as a pair of signature keys. Then, DO publishes the public parameters and public key for the signature to the smart contract, while the system master key and the private key for signature keep secret

(iii) SESP is registered in the SEContract, and a definite amount of deposit is required when registering. If SESP has fraudulent behavior, it will deduct part of the deposit as punishment

(iv) DU applies for registration in the PMContract, and he needs to provide his EOS account and a public key of ECC in which the EOS account is used for receiving compensation when SESP is dishonest

(v) DO generates the attribute key for DU according to his attributes set, then uses his public key to encrypt it and broadcast it to the blockchain. The ciphertext of the attribute key will be stored in the PMContract

(vi) DU obtains the ciphertext of his attribute key from PMContract and decrypts it locally by the corresponding private key

(vii) DO encrypts his data files and outsources them. (It can be uploaded to the cloud server, or IPFS, which is beyond the scope of this paper.) The returned address and the corresponding decryption key

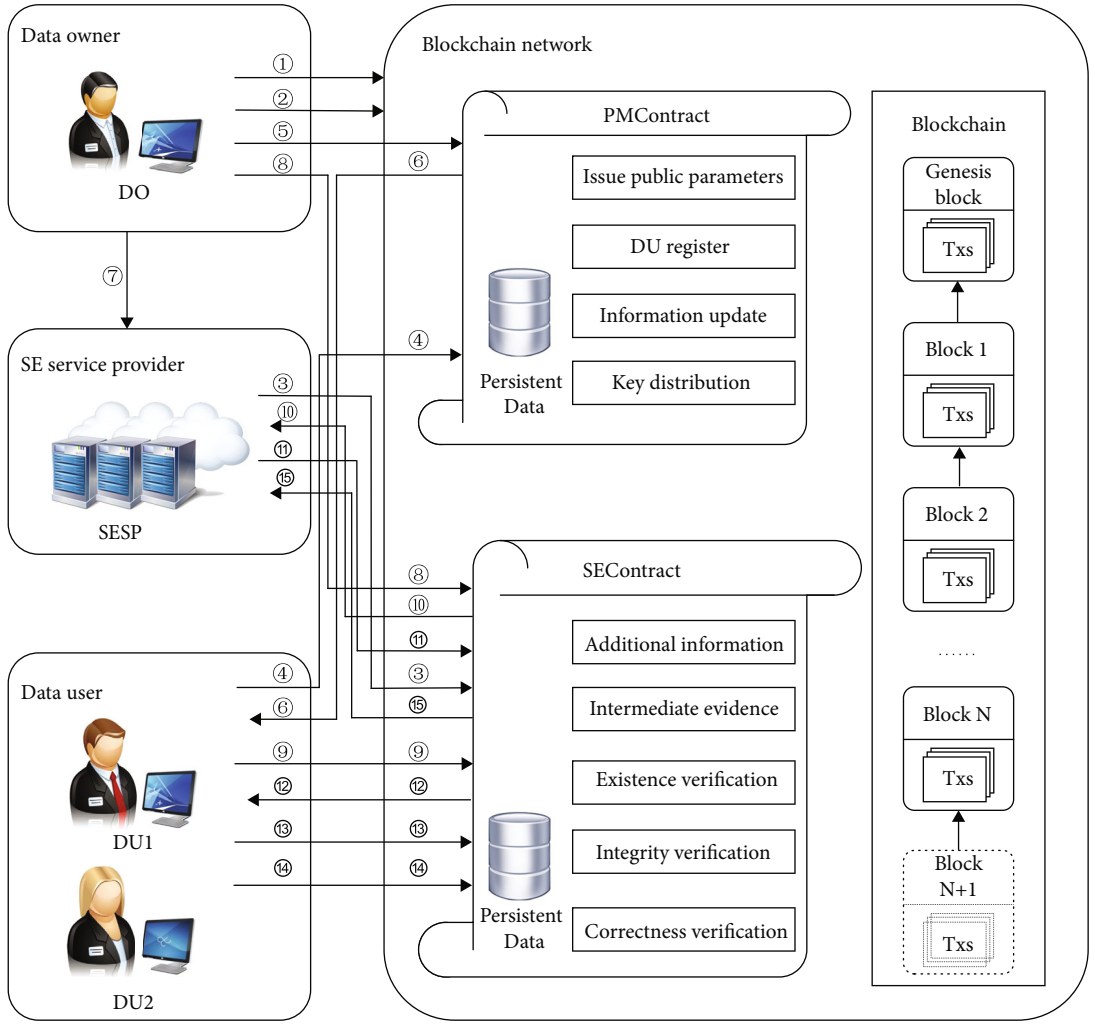

Figure 2: The system model of our scheme.

constitute the identity of the shared data. DO extracts the keywords set from the data files and builds indices for the keywords matching data files' identification, then generates auxiliary information simultaneously. DO uploads the indices to SESP

(viii) DO uploads the auxiliary information to SEContract

(ix) DU uses his private attribute key locally to generate the corresponding search token and additional commitment for his search request. Before searching, DU can check the amount of SESP deposit in PMContract. If the deposit is not enough to pay the penalty when doing evil, DU can choose not to continue

(x) SESP obtains the search request from SEContract, uses DU's search token and the indices uploaded by DO to execute the search algorithm, and returns the search results

(xi) SESP uploads search results to SEContract. It should be noted that SESP needs to complete the search and upload the results before the preagreed block height

(xii) DU gets his search results from SEContract

(xiii) If SESP does not provide any results before the specified block height, DU can withdraw his service fee through SEContract without any loss

(xiv) If DU believes that the search results returned by SESP are incorrect, DU could initiate a request for arbitration, and the blockchain will determine whether SESP has performed correctly

(xv) SESP gets his deposit from the contract

4.2. Detail Design of BFR-SE. BFR-SE consists of the following phase: initialization phase, apply and register phase, build index phase, token generation phase, search phase, verification phase, and withdraw phase. This section will describe each phase's detailed design in BFR-SE and give the corresponding relationship with the process steps in the flowchart.

(1) Initialization phase

The main work of the initialization phase is that DO creates smart contracts and deploys them on the blockchain. Then, DO generates the system master key MSK and public parameters PK locally. The core algorithm of this phase is $\text{Setup}(1^\lambda) \longrightarrow (\text{MSK}, \text{PK}, Sk_{\text{sig}}, Pk_{\text{sig}})$, which is run by DO locally. The algorithm's input is a security parameter 1^λ, and the outputs are MSK, PK, and a pair of keys for signature. After that, the DO publishes PK and Pk_{sig} to smart contracts and keep MSK, and Sk_{sig} secret locally. The corresponding steps in the system flowchart are ① and ②.

(2) Apply and Register phase

The apply and register phase's primary work is to complete the registration of SESP and DUs, including that DO distributes private attribute key for each DU. SESP needs to transfer a certain amount of system tokens to SEContract when applying for registration. If the amount of deposit is less than fines when doing evil, DU can choose not to use the search service. When DU applies for registration, it needs to provide a public key of ECC. After that, the DO generates a private attribute key for the DU according to his attributes set. The core algorithm is $\text{KeyGen}(\text{MSK}, \text{PK}, \omega) \longrightarrow Sk_\omega$, which is run by DO locally. The algorithm inputs MSK, PK, the attribute set ω of DU, and DU's public key Pk_{com} and outputs the private attribute key Sk_ω of DU. Then, DO uses the public key provided by DU when applying for registration to encrypt Sk_ω and obtain the ciphertext of Sk_ω, $CSk_\omega = \varepsilon.\text{Enc}_{Pk_{com}}(Sk_\omega)$. DO uploads CSk_ω to the SEContracts, so that DU can securely obtain and decrypt it to get his private attribute key Sk_ω. The corresponding steps in the system flowchart are ③, ④, ⑤, and ⑥.

(3) Build index phase

The main work of the build index phase is that DO encrypts the sharing data and outsource it. Take IPFS as an example, we can use the returned address and key to identify the data. After that, DO extracts the keywords set from the sharing data, builds indices for all the sharing data with the same keyword set, and generates the auxiliary information. DO sends the indices to SESP and uploads the auxiliary information to SEContract. The corresponding steps in the system flowchart are ⑦ and ⑧. It consists of the following three subalgorithms:

(a) $\text{EncryFile}(\{F_\eta\}_{1\le\eta\le d}) \longrightarrow (\{\text{FID}_\eta\}_{1\le\eta\le d})$

This algorithm is run by DO. For each element in $\{F_\eta\}_{1\le\eta\le d}$ where d denotes the number of the sharing data, DO encrypts it by the key key_η and outsource it. Taking IPFS as an example, the returned address of F_η is href_{F_η}, and the sharing data's identity is $\text{FID}_\eta = \text{IDGen}(\text{key}_\eta, \text{href}_\eta)$. The algorithm's final output is the identity set $\text{FID} = \{\text{FID}_\eta\}_{1\le\eta\le d}$. IDGen is an encryption function module defined by DO, which is not the focus of this paper, so the flowchart does not present it.

(b) $\text{IndexGen}(\text{KW}, \text{FID}) \longrightarrow I$

DO runs this algorithm, and the main work is to establish the corresponding indices based on sharing data and the relevant keywords. At first, DO need to extract the keywords KW_η for each F_η in $\{F_\eta\}_{1\le\eta\le d}$, then $\text{KW} = \text{KW}_1 \cup \text{KW}_2 \cdots \cup \text{KW}_d$. For $\forall \text{KW}_\tau = \{\text{kw}_1, \cdots, \text{kw}_m\}$ and $\text{KW}_\tau \in \text{KW}$ $(\text{KW}_\tau \neq \varnothing, 1 \le \tau \le n)$, if the corresponding data set is FID_τ, then use all the elements in FID_τ as leaf nodes to generate a Merkle Tree, and the root is MerkleRoot_τ. We defined that $I_\tau = (\text{KW}_\tau, \text{MerkleRoot}_\tau, \text{FID}_\tau)$, and the final indices of the keywords set KW matching the sharing data FID is $I = \{I_\tau\}_{1\le\tau\le n}$ in which n is the number of indices.

(c) $\text{Encrypt}(I, \mathscr{P}, \text{PK}) \longrightarrow (\text{CI}, \text{AI})$

DO runs this algorithm and the primary work is to encrypt the indices by a specific access policy to generate the ciphertext of indices CI and the auxiliary information AI. The inputs are the indices I, the access policy $\mathscr{P}$, and the public system parameters PK, while the outputs are CI and AI. DO encrypts the keywords of each I_τ in I to get the ciphertext of the keywords CKW_τ. DO signs CKW_τ and MerkleRoot_τ by his private key to ensure the integrity of the index. DO uploads the ciphertext $\text{CI} = \{\text{CI}_\tau\}_{1\le\tau\le n}$ to SESP, and the corresponding auxiliary information AI is uploaded to SEContract. For example, with four files, the data structure of an index is shown in Figure 3.

(4) Token generation phase

The main work of the token generation phase is that DU uses his private attribute key to call the trapdoor function to generate a search token and commitment for the searching keywords. $\text{TokenGen}(Sk_\omega, \text{KW}_{\text{search}}) \longrightarrow (\text{TOK}, \text{COMM})$ is the core algorithm whose inputs are the private attribute key Sk_ω of DU, and the keywords set $\text{KW}_{\text{search}}$ retrieved, and outputs are search token TOK and the commitment COMM. The commitment is to prove that DU did search for the keywords set provided by him when arbitrating. After that, DU uploads TOK and COMM to SEContract and pays the fee simultaneously. The corresponding steps in the system flowchart is ⑨.

(5) Search phase

The search phase's primary work is to use the search token to retrieve the ciphertext of indices uploaded by DO and return the successful matching results. The core algorithm of this phase is $\text{TEST}(\text{TOK}, \text{CI}_\tau) \longrightarrow \{0, 1\}$, which is run by SESP locally. This algorithm's inputs are the search token TOK and the ciphertext of index CI_τ, and the output is 0 or 1. If the output is 1, then the match is booming, and the search result is $\text{CI}_{\text{result}}$. SESP needs to upload $\text{CI}_{\text{result}}$ to SEContract before the preagreed time. Otherwise, DU can claim back the charge fee. The corresponding steps in the system flowchart are ⑩, ⑪, ⑫, and ⑬.

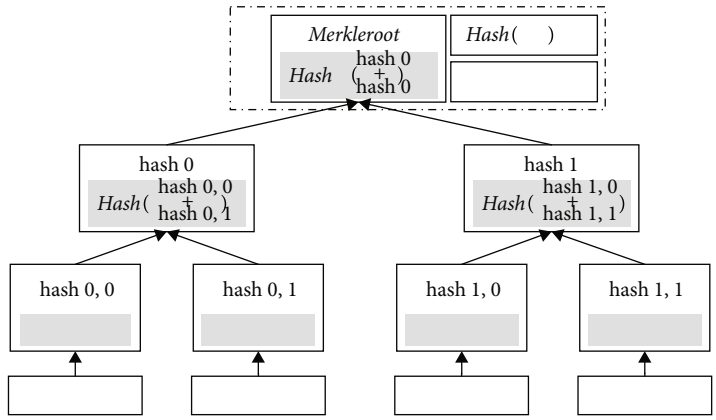

Figure 3: An example of the data structure for an index.

(6) Verification phase

The verification phase's primary work is to verify the search results returned by SESP according to the results and the auxiliary information uploaded by DO, including the verification of existence, integrity, and correctness. If the verification result is that the SESP has done evil, economic punishment will be imposed on SESP. This algorithm is executed by the blockchain, corresponding to step ⑭ in the flowchart. It can be subdivided into three subphases as follows:

(a) ExistenceVerify(AI, random, $\{H_2(W_j')\}_{1\le j\le m}$, COMM) $\longrightarrow \{0,1\}$

When the search result returned by SESP is *null*, the algorithm can verify the existence of the sharing data searched by DU. The algorithm's inputs are the auxiliary information AI, a random number related to the search token and $\{H_2(W_j')\}_{1\le j\le m}$, and the commitment corresponding to the search request. The output is 0 or 1.

(b) IntegrityVerify($\mathrm{CI}_{\mathrm{result}}$) $\longrightarrow \{0,1\}$

This algorithm can verify the integrity of the search results returned by SESP and prevent SESP from returning only partial or even forged results. The input of this algorithm is the search result $\mathrm{CI}_{\mathrm{result}}$, and the output is 0 or 1.

(c) CorrectnessVerify(TOK, $\mathrm{CI}_{\mathrm{result}}$) $\longrightarrow \{0,1\}$

This algorithm can verify the correctness of the search results returned by SESP and prevent SESP from returning the wrong results. This algorithm's inputs are the search token TOK and the search result $\mathrm{CI}_{\mathrm{result}}$. The output is 0 or 1.

(7) Withdraw phase

The main work of this phase is that each participant withdraws their coins from smart contract. The SESP's coin includes the deposit at the time of registration and DU's payment for using the search service. The coin of DU is mainly compensation, which comes from the penalty of SESP. It should be noted that each fee needs a freeze period, during which DU can apply for arbitration on the blockchain. Only after the freezing period has passed can SESP withdraw this fee from the contract. The corresponding steps in the system flowchart are ⑮.

5. Implementation Details of Proposed Scheme

To achieve our goal, we constructed an attribute-based searchable encryption algorithm that supports multikeyword search and combined with the EOS blockchain platform to realize our fair and reliable scheme. This section will elaborate on the details of our smart contracts deployed on EOS and the concrete construction of BFR-SE.

5.1. Smart Contract Design. In order to make the logic clearer, we divide the smart contract in the scheme into two parts, PMContract and SEContract. We use *_self* to represent the account of smart contract itself and *_self.asset* to represent the balance in the contract. Let require_auth be a function that represents which account's permission is needed to continue. We will describe the two smart contracts in detail in this section.

5.1.1. Participant Management Contract (PMContract). The PMContract is composed of five interfaces: *SetSPK*, *Register*, *GetPK*, *SetSK*, and *GetSK*. We initialize PMContract as follows:

Let three-tuple $(\mathrm{Account}_{\mathrm{user}}, \mathrm{Pk}_{\mathrm{com}}, CSk_{\omega})$ denote a DU and create a multi_index named *table_user*, in which $\mathrm{Account}_{\mathrm{user}}$ is an EOS account of DU, Pk_{com} is a public key of DU, and CSk_{ω} is the private attribute key of DU. Let $\mathrm{Account}_{\mathrm{user}}$ be the primary key of *table_user*, whose corresponding index is *account_idx*. Let PK denote the system public parameters.

(1) *SetSPK*: when PMContract receives action (PMContract, *SetSPK*, Auth, (*pk*)), this function will be triggered to execute. It can only be invoked by DO to set and update the public system parameters

(2) *Register*: when PMContract receives action (PMContract, Register, Auth, (A_{user}, Pk_{com})), this function will be triggered to execute. It is invoked by DU to apply for registration in the system. The detail of this function can be seen in Algorithm 1

(3) *GetPK*: when PMContract receives action (PMContract, GetPK, Auth, (A_{user})), this function will be triggered to execute

(4) *SetSK*: when PMContract receives action (PMContract, SetSK, Auth, (A_{user})), this function will be triggered to execute. It is used for DO distributing private attribute keys to DUs. The detail of this function can be seen in Algorithm 2

(5) *GetSK*: when PMContract receives action (PMContract, GetSK, Auth, (A_{user})), this function will be triggered to execute

5.1.2. Searchable Encryption Contract (SEContract). The SEContract consists of 13 functions: *SetPK*, *SetAI*, *Deposit*, *SearchRequest*, *SendResult*, *ExistenceVerify*, *IntegrityVerify*, *CorrectnessVerify*, *GetFeeSESP*, *IsVerifyRound*, *IsResultReady*, *Compensate*, and *CommitVerify*. We initialize SEContract as follows:

Let six-tuple ($Account_{user}$, SerialNum, TOK, COMM, Height, Coin) be a search request initiated from DU and create a multi_index named *search_table*, in which $Account_{user}$ is the account of DU, SerialNum is the serial number of the search request, TOK is the search token, COMM is the commitment of DU for the request, Height is the block height when the request is initiated, and Coin is the charge fee paid by the user for the service. Let $Account_{user}$ be the primary key of *search_table*, and the corresponding index is *search_idx*. Let three-tuple ($Account_{user}$, SerialNum,CI_{result}) be a result returned from SESP and create a multi_index named result_table, in which $Account_{user}$ and SerialNum are to match the search request in *search_table*, and CI_{result} denotes the search result. Let $Account_{user}$ be the primary key of *result_table*, and the corresponding index is *result_idx*. Let $Account_{sesp}$ be the account of SESP and Deposit be the balance of SESP in the contract. Let d represent the fee that the user needs to pay for each search and the amount of penalty when SESP does evil. Let *round_height* represent the time required for search round and verification round, BF be the auxiliary information which is a Bloom filter, and PK_{Sig} be the public key for the signature of DO.

(1) *SetPK*: when SEContract receives action (SEContract, SetPK, Auth, (*pk*)), this function will be triggered to execute

(2) *SetAI*: when SEContract receives action (SEContract, SetAI, Auth, (*AI*)), this function will be triggered to execute

Input: A_{user}, Pk_{com}
Output: void
1 **require_auth**(A_{user})
2 u = *account_idx*.find(A_{user})
3 **if** (u == *null*) **then**
4 u.Pk_{com}= Pk_{com}
5 account_idx.modify(u)
6 **else**
7 u.$Account_{user}$= A_{user}
8 u.Pk_{com}= Pk_{com}
9 *account_idx*.emplace(u)
10 **end**

Algorithm 1: Register.

(3) *Deposit*: when SEContract receives action (SEContract, Deposit, Auth, (A_{sesp}, coin)), this function will be triggered to execute. The detail of this function can be seen in Algorithm 3

(4) *SearchRequest*: when SEContract receives action (SEContract,SearchRequest,Auth, (A_{user},Sn,TOK, COMM)), this function will be triggered to execute. The detail of this function can be seen in Algorithm 4

(5) *SendResult*: when SEContract receives action (SEContract, SendResult, Auth, (A_{user}, Sn, CI_{result})), this function will be triggered to execute. It can only be invoked by SESP. The detail of this function can be seen in Algorithm 5

(6) *ExistenceVerify*: when SEContract receives action (SEContract, ExistenceVerify, Auth, (A_{user},random, $\{H_2(W_j')\}_{1\le j\le m}$)), this function will be triggered to execute. The detail of this function can be seen in Algorithm 6

(7) *IntegrityVerify*: when SEContract receives action (SEContract, IntegrityVerify, Auth, (A_{user})), this function will be triggered to execute. The detail of this function can be seen in Algorithm 7

12 **end**

(8) *CorrectnessVerify*: when SEContract receives action (SEContract, CorrectnessVerify, Auth, (A_{user})), this function will be triggered to execute. The detail of this function can be seen in Algorithm 8

(9) *GetFeeSESP*: when SEContract receives action (SEContract, GetFeeSESP, Auth, (A_{user})), this function will be triggered to execute. The detail of this function can be seen in Algorithm 9

```
Input: A_user,CSk_ω
Output: bool
require_auth(_self)
u = account_idx.find(A_user)
if u == null then
    return false
else
    u.CSk_ω=CSk_ω
    account_idx.modify(A_user)
    return true
end
```

Algorithm 2: SetSK.

(10) *IsVerifyRound*: this is a private function and can only be called internally by the contract itself. The detail of this function can be seen in Algorithm 10

(11) *IsResultReady*: this is a private function and can only be called internally by the contract itself. The detail of this function can be seen in Algorithm 11

(12) *Compensate*: this is a private function and can only be called internally by the contract itself. The detail of this function can be seen in Algorithm 12

(13) *CommitVerify*: this is a private function and can only be called internally by the contract itself. The core of this function is the algorithm TEST in the search phase. For detailed implementation, see the concrete construction of BFR-SE in the next section

5.2. Concrete Construction of BFR-SE. This section shows the concrete construction of BFR-SE, including the algorithms to be executed at each phase and how each participant interacts with the EOS blockchain. Our initialization is as follows:

Let G_0 and G_1 be cyclic groups of order p, and g be a generator of G_0. Let $e: G_0 \times G_0 \longrightarrow G_1$ be a bilinear pairing, $S = \{1, \cdots, l\}$ be the set of all attributes, $\{h_1{}', \cdots, h_k{}'\}$ be k general and distinct hash functions. $H_1: \{0,1\}* \longrightarrow G$ and $H_2: \{0,1\}* \longrightarrow Z_p$ are also two hash functions.

(1) $\text{Setup}(1^\lambda)$

Firstly, DO picks $a, b, c \longleftarrow Z_p$ randomly. For each attribute $\{i \mid i \in S\}$, compute that $\{h_i = H_1(i) \mid i \in S\}$.

The public system parameters are PK:

$$\text{PK} = \left\{ g^a, g^b, g^c, \{h_i\}_{i \in S} \right\}. \tag{2}$$

The system master key is MSK:

$$\text{MSK} = (a, b, c). \tag{3}$$

DO randomly selects a key pair of ECDSA which be denoted $(Sk_{\text{sig}}, Pk_{\text{sig}})$, then keeps MSK and Sk_{sig} secret and sends the following two transactions to the blockchain:

$$\begin{aligned} &(\text{Ref}_{\text{block}}, t, \text{Sig}_{\text{DO}}(\text{Chain\_ID}, \text{Tx}), \\ &\text{Action}(\text{PMContract}, \text{SetSPK}, \text{Auth}_{\text{DO}}, (\text{PK}))), \end{aligned} \tag{4}$$

$$\begin{aligned} &(\text{Ref}_{\text{block}}, t, \text{Sig}_{\text{DO}}(\text{Chain\_ID}, \text{Tx}), \\ &\text{Action}\left(\text{SEContract}, \text{SetPK}, \text{Auth}_{\text{DO}}, \left(\text{Pk}_{\text{Sig}}\right)\right)), \end{aligned} \tag{5}$$

(2) $\text{KeyGen}(\text{MSK}, \omega) \longrightarrow Sk_\omega$

At first, DO sends the following transaction to the blockchain to obtain the DU's public key:

$$\begin{aligned} &(\text{Ref}_{\text{block}}, t, \text{Sig}_{\text{DO}}(\text{Chain\_ID}, \text{Tx}), \\ &\text{Action}(\text{PMContract}, \text{GetPK}, \text{Auth}_{\text{DO}}, (\text{A}_{\text{user}}))) \end{aligned} \tag{6}$$

Let the attribute set of DU be ω and $\omega \in S$, then randomly pick $t \longleftarrow Z_p$ and compute $K_1 = g^{(ac-t)/b}$, $K_2 = g^t$. For each $i \in \omega$, it computes that $K_{3,i} = h_i^t$. The private attribute key of DU is Sk_ω:

$$Sk_\omega = \left(K_1, K_2, \left\{K_{3,i}\right\}_{i \in \omega}\right). \tag{7}$$

Then, encrypt it with the public key of DU:

$$CSk_\omega = \varepsilon.\text{Enc}_{Pk_{com}}(Sk_\omega). \tag{8}$$

DO sends the following transaction to the blockchain:

$$\begin{aligned} &(\text{Ref}_{\text{block}}, t, \text{Sig}_{\text{DO}}(\text{Chain\_ID}, \text{Tx}), \\ &\text{Action}(\text{SEContract}, \text{SetSK}, \text{Auth}_{\text{DO}}, (\text{A}_{\text{user}}, \text{CSk}_\omega))), \end{aligned} \tag{9}$$

(3) $\text{Encrypt}(I, (A_{l \times k}, \rho), PK) \longrightarrow (\text{CI}, \text{AI})$

For $\forall I_\tau \in I$, $I_\tau = (\text{KW}_\tau, \text{MerkleRoot}_\tau, \text{FID}_\tau)$, randomly picks $r, s \longleftarrow Z_p$ and computes $C_0 = g^{cr}$, $C_1 = g^{bs}$. Let $(A_{l \times k}, \rho)$ be an access structure. It randomly chooses $y_2, y_3, \cdots, y_k \in Z_p$ and sets $\vec{\mathbf{v}} = (s, v_2, \cdots, v_k)^T$. For each $i = 1$ to l, it calculates $\sigma_i = A_i \times \vec{\mathbf{v}}$. Then, randomly picks $r_1, \cdots, r_l \in Z_p$ and performs the following calculations for each attribute:

$$C_{2,i} = g^{a\sigma_i} h_i^{-r_i}, C_{3,i} = g^{r_i}. \tag{10}$$

Let m be the size of the keyword set KW_τ. For each W_j

```
Input: A_sesp, coin
Output: void
require_auth(A_sesp)
if Account_sesp== null then
    Account_sesp=A_sesp
end
send action (eosio.token, transfer, Auth, (A_sesp,_self, coin))
Deposit = Deposit + coin
```

Algorithm 3: Deposit.

```
Input: A_user, Sn,TOK, COMM
Output: void
require_auth(A_user)
s = search_idx.find(A_user)
send action (eosio.token, transfer, Auth, (A_user,_self, d))
if s == null then
    s.Account_user = A_user
    s.SerialNum = Sn
    s.TOK = TOK
    s.COMM = COMM
    s.Coin = d
    s.Height = getCurrentHeight()
    search_idx.emplace(s)
else if (getCurrentHeight() > (s.Height+2 × round_height)) then
    s.SerialNum = Sn
    s.TOK = TOK
    s.COMM = COMM
    s.Coin = d
    s.Height = getCurrentHeight()
    search_idx.modify(s)
end
```

Algorithm 4: SearchRequest.

```
Input: A_user, Sn, CI_result
Output: void
require_auth(A_sesp)
rlt=result_idx.find(A_user)
if rlt!=null then
    rlt.SerialNum=Sn
    rlt.CI_result=CI_result
    result_idx.modify(rlt)
else
    rlt.A_user=A_user
    rlt.SerialNum=Sn
    rlt.CI_result=CI_result
    result_idx.Emplace(rlt)
end
```

Algorithm 5: SendResult.

in KW_τ, $1 \le j \le m$, perform the following calculation:

$$C_{4,j} = g^{a(r+s)/m} g^{brH_2(W_j)}. \quad (11)$$

The ciphertext of KW_τ will be

$$\mathrm{CKW}_\tau = \left\{C_0, C_1, \left\{C_{2,i}, C_{3,i}\right\}_{i\in(1,\cdots,l)}, \left\{C_{4,j}\right\}_{j\in(1,\cdots,m)}\right\}. \quad (12)$$

DO signs the ciphertext of the index by his private key:

$$\mathrm{Sig}_\tau = \mathrm{SigGen}\left(\mathrm{Sk}_{\mathrm{Sig}}, \mathrm{MerkleRoot}_\tau \| \mathrm{SHA256}(\mathrm{CKW}_\tau)\right). \quad (13)$$

Let CI_τ be the ciphertext corresponding to KW_τ:

$$\mathrm{CI}_\tau = (\mathrm{CKW}_\tau, \mathrm{MerkleRoot}_\tau, \mathrm{Sig}_\tau, \mathrm{FID}_\tau). \quad (14)$$

Set $\mathrm{HKW}_\tau = \mathrm{SHA256}(H_2(W_1)\|, \cdots, \|H_2(W_m))$, $W_j \in KW_\tau, 1 \le j \le m$.

Input: A_{user}, *random*, $\{H_2(W_j{}')\}_{1\le j\le m}$
Output: void
1 bool $IsVr = IsVerifyRound(A_{\text{user}})$
2 bool $IsRd = IsResultReady(A_{\text{user}})$
3 **if** *IsVr== true* && *IsRd==false* **then**
4 $Compensate(A_{\text{user}})$
5 **else if** *IsVr== true* && *IsRd==true* **then**
6 **if** $CommitVerify(A_{\text{user}}, random, \{H_2(W_j{}')\}_{1\le j\le m})$ == true **then**
7 bool $isExist = BFVerify(BF, \text{SHA256}(\{H_2(W_j{}')\}_{1\le j\le m}))$
8 **if** *isExist == false* **then**
9 $Compensate(A_{user})$
10 **end**
11 **end**

Algorithm 6: ExistenceVerify.

Input: A_{user}
Output: void
1 bool IsVr = IsVerifyRound(A_{user})
2 bool IsRd = IsResultReady(A_{user})
3 **if** IsVr== true && IsRd==false **then**
4 Compensate(A_{user})
5 **else if** IsVr== true && IsRd==true **then**
6 rlt = result_idx.find(A_{user})
7 bool isMK = VerifyMerkleRoot(rlt.MerkleRoot$_\tau$, rlt.FID)
8 bool isSig = SigVerify(rlt.MerkleRoot$_\tau$|| SHA256(rlt.CKW$_\tau$), rlt.Sig$_\tau$, PK$_{\text{Sig}}$))
9 **if** isMK && isSig **then**
10 **throw**
11 else
12 Compensate(A_{user})
13 end
14 end

Algorithm 7: IntegrityVerify.

Input: A_{user}
Output: void
1 bool $IsVr = IsVerifyRound(A_{\text{user}})$
2 bool $IsRd = IsResultReady(A_{\text{user}})$
3 **if** *IsVr== true* && *IsRd==false* **then**
4 $Compensate(A_{\text{user}})$
5 **else if** *IsVr== true* && *IsRd==true* **then**
$s = search\_idx.\text{find}(A_{\text{user}})$
6 $rlt = result\_idx.\text{find}(A_{\text{user}})$
7 bool $isCv = Test(s.TOK, rlt.\text{CKW}_\tau)$
9 **if** *isCv* == false **then**
12 $Compensate(A_{\text{user}})$
13 **end**
14 **end**

Algorithm 8: CorrectnessVerify.

Finally, the ciphertext of the indices and the auxiliary information will be as follows:

$$\text{CI} = \{\text{CI}_\tau\}_{1\le\tau\le n}$$
$$\text{AI} = \text{BFGen}\left(\left\{h_1', \cdots, h_k'\right\}, \{\text{HKW}_1, \cdots, \text{HKW}_n\}\right) \longrightarrow \text{BF}. \tag{15}$$

After that, DO uploads CI to SESP, and the auxiliary information AI is uploaded to the blockchain by sending the following transaction:

$$(\text{Ref}_{\text{block}}, t, \text{Sig}_{\text{DO}}(\text{Chain\_ID}, \text{Tx}), \text{Action}(\text{SEContract}, \text{SetAI}, \text{Auth}_{\text{DO}}, (\text{AI})), \tag{16}$$

(4) $\text{TokenGen}(Sk_\omega, \text{KW}_{\text{search}}) \longrightarrow (\text{TOK}, \text{COMM})$

```
Input: A_user
Output: void
1 s = search_idx.find(A_user)
2 if (getCurrentHeight() > s.Height+2 × round_height)& &
(s.Coin>0) then
3       Deposit = Deposit + s.Coin
4       s.Coin =0
5       search_idx.modify(s)
6 end
```

Algorithm 9: GetFeeSESP.

Firstly, DU obtains his ciphertext of private attribute key CSk_ω by sending the following transaction:

$$\begin{aligned}&(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{user}}(\mathrm{Chain\_ID}, \mathrm{Tx}),\\ &\mathrm{Action}(\mathrm{SEContract}, \mathrm{GetSK}, \mathrm{Auth}_{\mathrm{user}}, (\mathrm{A}_{\mathrm{user}})).\end{aligned} \tag{17}$$

DU decrypts it to get the private attribute key:

$$Sk_\omega = \varepsilon.\mathrm{Dec}_{Sk_{com}}(CSk_\omega). \tag{18}$$

Then, DU randomly picks $\pi \longleftarrow Z_p$. Let m be the size of $\mathrm{KW}_{\mathrm{search}}$ and compute it as follows:

$$\begin{aligned}\mathrm{tok}_1 &= g^{a\pi}\prod_{j=1}^{m} g^{b\pi H_2(W_j')},\\ \mathrm{tok}_2 &= g^{c\pi},\\ \mathrm{tok}_3 &= K_1^\pi = g^{(ac-t)\pi/b},\\ \mathrm{tok}_4 &= K_2^\pi = g^{\pi t},\end{aligned} \tag{19}$$

For each $i \in \omega$, it computes $H_i = K_{3,i}^\pi = h_i^{t\pi}$.Set $\mathrm{tok}_5 = \{K_{3,i}^\pi\}_{i\in\omega} = \{h_i^{\pi t}\}_{i\in\omega}$, and the search token will be *TOK*:

$$\mathrm{TOK} = \{\mathrm{tok}_1, \mathrm{tok}_2, \mathrm{tok}_3, \mathrm{tok}_4, \mathrm{tok}_5\}. \tag{20}$$

It calculates the commitment of the search request as follows:

$$\begin{aligned}\mathrm{HKW} &= \mathrm{SHA256}\left(H_2\left(W_1'\right)\middle\|, \cdots, \middle\|H_2\left(W_m'\right)\right),\\ \mathrm{COMM} &= \mathrm{SHA256}(\pi\|\mathrm{HKW}).\end{aligned} \tag{21}$$

Finally, DU picks $Sn \longleftarrow Z_q$ randomly as the serial number of the search request and sends the following transaction:

$$\begin{aligned}&(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{user}}(\mathrm{Chain\_ID}, \mathrm{Tx}),\\ &\mathrm{Action}(\mathrm{SEContract}, \mathrm{SearchRequest}, \mathrm{Auth}_{\mathrm{user}}, (A_{\mathrm{user}}, \mathrm{Sn}, \mathrm{TOK}, \mathrm{COMM})),\end{aligned} \tag{22}$$

(5) $\mathrm{TEST}(\mathrm{TOK}, \mathrm{CI}_\tau) \longrightarrow \{0, 1\}$

After SESP receives the TOK from DU, it will compare each row of CI. Firstly, it is judged whether the number of keywords m in CI_τ is the same as that in TOK. If different, compare the next row.

Assuming that the attribute set of DU satisfies the access policy, set μ_i be the recovery coefficient of the ith row in $A_{l\times k}$ and calculate as follows:

$$E = \prod_{i\in\omega}(e(C_{2,i}, \mathrm{tok}_4)e(C_{3,i}, H_i))^{\mu_i}. \tag{23}$$

Then, determine whether the following two formulas (4) and (5) are equal. If it is, it returns 1. Otherwise, it returns 0.

$$\begin{aligned}&e(C_0, \mathrm{tok}_1)e(\mathrm{tok}_3, C_1)E,\\ &e\left(\prod_{j=1}^{m} C_{4,j}, \mathrm{tok}_2\right).\end{aligned} \tag{24}$$

If the result is found successfully, then $\mathrm{CI}_{\mathrm{result}} = \mathrm{CI}_\tau$; otherwise, $\mathrm{CI}_{\mathrm{result}} = \mathrm{null}$. SESP sends the following transaction to the blockchain.

$$\begin{aligned}&(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{SESP}}(\mathrm{Chain\_ID}, \mathrm{Tx}),\\ &\mathrm{Action}(\mathrm{SEContract}, \mathrm{Send\ Result}, \mathrm{Auth}_{\mathrm{SESP}}, (A_{\mathrm{user}}, \mathrm{Sn}, \mathrm{CI}_{\mathrm{result}})).\end{aligned} \tag{25}$$

(6) Verification

If DU believes that there are problems with the search results returned by SESP during the verification round, DU can initiate a request to the blockchain within this period and request the blockchain to make a judgment.

(a) Existence

When the result is null, DU can send the following transaction to the blockchain:

$$\begin{aligned}&(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{user}}(\mathrm{Chain\_ID}, \mathrm{Tx}),\\ &\mathrm{Action}\left(\mathrm{SEContract}, \mathrm{ExistenceVerify}, \mathrm{Auth}_{\mathrm{user}}, \left(A_{\mathrm{user}}, \pi, \left\{H_2\left(W_j'\right)\right\}_{1\le j\le m}'\right)\right).\end{aligned} \tag{26}$$

After the contract receives the transaction, it will first verify the DU's previous commitment to prevent DU from

```
Input: A_user
Output: bool
1 s = search_idx.find(A_user)
2 if (getCurrentHeight() > (s.Height + round_height)) &&
      (getCurrentHeight() < (s.Height+2 × round_height)) then
3     return true
4 else
5     return false
6 end
```

Algorithm 10: IsVerifyRound.

```
Input: A_user
Output: bool
1 s = search_idx.find(A_user)
2 rlt = result_idx.find(A_user)
3 if s.SerialNum == rlt.SerialNum && rlt.CI_Result! = null then
4     return true
5 else
6     return false
7 end
```

Algorithm 11: IsResultReady.

```
Input: A_user
Output: void
1 s=search_idx.find(A_user)
2 send action (eosio.token, transfer, Auth, (_self, A_user, s.Coin+d))
3 s.Coin=0
4 search_idx.modify(A_user)
5 Deposit = Deposit - d
```

Algorithm 12: Compensate.

submitting keywords that are different from that in the search request. It will be calculated as follows:

$$\begin{aligned} \mathrm{HKW}' &= \mathrm{SHA256}\left(H_2\left(W_1{}'\right)\|, \cdots, \|H_2\left(W_m{}'\right)\right) \\ \mathrm{COMM}' &= \mathrm{SHA256}\left(\pi \| \mathrm{HKW}'\right) \end{aligned} \tag{27}$$

Blockchain determines whether COMM' and COMM in the contract are equal. If they are the same, the calculation will continue as follows:

$$g^{a\pi} g^{b\pi \sum_{j=1}^{m'} H_2\left(W_j\right)} = \mathrm{tok}_1 \tag{28}$$

If the above equation is tenable, this DU is honest. After that, compute the following formula to verify the existence of the search result.

$$\mathrm{BFVerify}\left(\mathrm{BF}, \mathrm{HKW}', \left\{h_1{}', \cdots, h_k{}'\right\}\right) \longrightarrow (0, 1). \tag{29}$$

(b) Integrity

When DU believes that the result returned by SESP is not complete or corrupted, he can send the following transaction to the blockchain.

$$\begin{aligned} &\left(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{user}}(\mathrm{Chain\_ID}, \mathrm{Tx}),\right. \\ &\left.\mathrm{Action}(\mathrm{SEContract}, \mathrm{IntegrityVerify}, \mathrm{Auth}_{\mathrm{user}}, (A_{\mathrm{user}}))\right). \end{aligned} \tag{30}$$

(c) Correctness

When DU believes that the result returned by SESP does not include the keywords searched for, he can send the following transaction:

$$(\mathrm{Ref}_{\mathrm{block}}, t, \mathrm{Sig}_{\mathrm{user}}(\mathrm{Chain\_ID}, \mathrm{Tx}), \mathrm{Action}(\mathrm{SEContract}, \mathrm{CorrectnessVerify}, \mathrm{Auth}_{\mathrm{user}}, (A_{\mathrm{user}})). \tag{31}$$

6. Security and Performance Analysis of Proposed Scheme

6.1. Security and Privacy Analysis of BFR-SE

6.1.1. Security and Privacy of ABSE. The ABSE model we constructed in BFR-SE is inspired by the attribute-based search model VABKS of Zheng et al. [21]. Furthermore, we have expanded it to support the multikeyword search. VABKS is proved to be secure, and the complete proof process can refer to the security analysis in [21]. The security of VABKS relies on the decisional linear assumption.

We focus on the fairness and reliability of searchable encryption, and the security of ABSE is not the main work of this paper. So, we will mainly analyze the correctness of ABSE and briefly discuss its security and privacy.

(1) Correctness. Let μ_i be the recovery coefficient of the ith row in $A_{l\times k}$ and calculate as follows:

$$E = \prod_{i\in\omega}(e(C_{2,i}, \mathrm{tok}_4)e(C_{3,i}, H_i))^{\mu_i} = \prod_{i\in\omega} e(g,g)^{\sigma_i \pi t \mu_i} = e(g,g)^{\pi t \sum_{i\in\omega}\mu_i\sigma_i}. \tag{32}$$

If the attribute set ω of DU satisfies the access policy $(A_{l\times k}, \rho)$, it will get s by calculating $\sum_{i\in\omega}\mu_i\sigma_i$, and $E = e(g,g)^{\pi t s}$.

Then, calculate as follows:

$$e\left(\prod_{j=1}^{m} C_{4,j}, \mathrm{tok}_2\right) = e(g,g)^{arc\pi + as c\pi + br c\pi \sum_{j=1}^{m} H_2(W_j)},$$

$$e(C_0, \mathrm{tok}_1) = e(g,g)^{cra\pi + crb\pi \sum_{j=1}^{m} H_2(W_j')},$$

$$e(\mathrm{tok}_3, C_1) = e(g,g)^{ac\pi s - t\pi s}. \tag{33}$$

If the keywords set KW' in the search token is the same as the keywords set KW in the index, it will have

$$e(C_0, \mathrm{tok}_1)e(\mathrm{tok}_3, C_1)E == e\left(\prod_{j=1}^{m} C_{4,j}, \mathrm{tok}_2\right). \tag{34}$$

(2) Security and Privacy. From a security point of view, all attribute-based cryptographic algorithms need to resist collusion attacks. In the ABSE used in BFR-SE, we pick a random number t for each DU at the key generation phase, and the attribute-based part of the private key $\{K_{3,i}\}_{i\in\omega}$ is related to it. So, different DUs cannot combine their respective attributes to launch a collision attack.

From the perspective of privacy, DO encrypts his indices and stores them on SESP without revealing any information. Moreover, DO has fine-grained access control on the search function. For DU, a random number π is used every time generating a search token, making the search token generated will not be the same even if DU searches for the same keywords multiple times. The adversary cannot analyze DUs' privacy by collecting the traces of search requests.

6.1.2. Fairness and Reliability of BFR-SE

(1) Fairness. In this paper, we proposed a pay-per-use searchable encryption scheme. We believe that both SESP and DU are not always credible, and the dishonest behavior of either party may cause economic disputes. In our scheme, the participant with substantial computing power needs to pay a certain amount of deposit before becoming SESP. We divide each search of DU into two rounds: search round and verification round. When a DU initiates a search request, he needs to transfer the fee to smart contract.

DUs can initiate a request for arbitration in the verification round when SESP does not return any result, return partial results, or return incorrect results. The blockchain will arbitrate the search results. If it determines that SESP has acted dishonestly, SESP will be subject to a financial penalty, and the fine will compensate DU. Although DU may expose a little bit of their information when applying for arbitration, they will get financial compensation.

For SESP, if he can provide the correct results before the pre-agreed time, he can take away his profits after the freezing period. When DU initiates a search request, he promises the set of keywords retrieved and stores it on-chain, ensuring that DU will not submit a different set of keywords during the verification round to defraud the SESP deposit. Moreover, a particular fine can be introduced to DU so that DU cannot apply for arbitration without any certainty.

In summary, BFR-SE is fair to both SESP and DU.

(2) Reliability. In our scheme, we draw on the idea of verifiable searchable encryption in which DU can verify the results return by SESP from three aspects, including existence, integrity, and reliability. Therefore, the reliability of verifiable searchable encryption is also available in our scheme. Unlike the verifiable searchable encryption, we utilize the blockchain to introduce a reward and punishment mechanism for the scheme. When DU finds any problem with the results returned by SESP, they apply for arbitration during the verification round. If the SESP is indeed dishonest, the blockchain will punish SESP financially and compensate DU.

Table 2: Functional comparison between BFR-SE and other related searchable encryption schemes.

	BFR-SE	Ref. [20]	Ref. [21]	Ref. [29]	Ref. [30]
Fairness	Yes	No	No	Yes	Yes
Reliability	Strong	Weak	Weak	No	No
Privacy protection	Strong	Strong	Strong	Weak	Weak
Multikeyword search	Yes	Yes	No	No	Yes
Suitable for multiusers	Yes	No	Yes	Yes	Yes
Fine-grained access control	Yes	No	Yes	No	No
Practicability	Yes	Yes	Yes	No	No

Therefore, BFR-SE is more reliable than previous schemes.

6.1.3. Verifiability of Search Results

(1) Existence. BFR-SE uses Bloom filter to verify the existence of the search result. DO stores all the information corresponding to the keyword set in the indices into the Bloom filter. If the result returned by SESP is null, which means that there is no matching data for the search request, DU could verify it using the keywords searched for and Bloom Filter. If the verification result is 1, the keywords searched for having a high probability of existence. Although there is a specific false-positive rate, the research in Ref. [37] shows that the calculation method of this false-positive rate is as follows:

$$\left(1-\left(1-\frac{1}{l_{BF}}\right)^{kn}\right)^{k} \approx \left(1-e^{-kn/l_{BF}}\right)^{k}. \tag{35}$$

It can be seen that by setting the values of l_{BF} and k, the false-positive rate can be reduced. For example, when $k = (\ln 2) l_{\mathrm{BF}}/n$, it can get a minimum false-positive rate of $(0.6185)^{l_{BF}/n}$. Therefore, DUs can verify the existence of search results in BFR-SE.

(2) Integrity. For each row of indices in our scheme, DO uses his private key to sign the keywords and the MerkleRoot obtained from the data-related information as leaf nodes. Each row of the indices uploaded by DO contains the signature. DUs can use the public key of DO to verify whether the keywords and MerkleRoot are damaged. For the returned search results, DU can verify the data's integrity according to whether the MerkleRoot can be constructed. Once the data-related information as leaf nodes are destroyed, or the SESP only returns partial results, the MerkleRoot cannot be constructed. Therefore, DUs can verify the integrity of search results in BFR-SE.

(3) Correctness. If the existence and integrity are verified, then DU can get the ciphertext of the keyword set of the search results. DU only needs to use this ciphertext and his search token as inputs and repeatedly execute the TEST function to verify the search results. Therefore, DUs can verify the correctness of search results in BFR-SE.

6.2. Security and Privacy Analysis of BFR-SE

6.2.1. Functional Comparison. We compared BFR-SE with previous verifiable searchable encryption and blockchain-based searchable encryption schemes from the following aspects: fairness, reliability, privacy protection, whether it supports multikeyword search, whether it is suitable for multi-user situation, whether it supports fine-grained access control for the search function, and practicability.

From the comparison in Table 2, comparing verifiable searchable encryption and previous blockchain-based searchable encryption schemes, the following conclusions can be drawn:

(1) The former and earlier related studies did not consider the fairness of searchable encryption. With the emergence of blockchain, these blockchain-based schemes all meet the requirements of fairness

(2) The former supports DUs to verify search results and has a certain degree of reliability, but because there is no subsequent sufficient punishment, the reliability is weak. However, the recent searchable encryption schemes based on blockchain have not been considered reliable

(3) The former stores the indices and DU's search records on SESP, which can better protect DU's privacy on the premise that SESP is credible. The latter stores the indices and search records on blockchain and uses a deterministic encryption algorithm. Since the information on-chain is public, even if the keywords are encrypted, when the search records are large enough, it is not impossible to analyze some privacy of DU

(4) The performance of the former depends on the capabilities of SESP and has high practicability. The latter is mostly based on low-performance blockchain platforms such as Bitcoin and Ethereum, and the design is not particularly perfect, so there are still problems in performance and security. In real enterprise applications, it does not have practicality

Compared with these two types of schemes, BFR-SE designs a relatively perfect reward and punishment mechanism with blockchain. If SESP is dishonest, it will pay the price. Therefore,

our scheme has both fairness and reliability. The indices are still stored on SESP in our scheme, and the blockchain only plays the role of arbitration when there are disputes, which makes BFR-SE more efficient than the previous blockchain-based schemes. Our constructed ABSE is not a deterministic encryption algorithm. The random number makes the search token different, even for the same keyword set. Therefore, BFR-SE has strong privacy protection capabilities. We have extended the work of Ref. [21] to support a multikeyword search. The combination of attribute-based searchable encryption and blockchain makes BFR-SE meet multiuser scenarios in a distributed environment quickly and enables DO to have fine-grained access control on his sharing data. BFR-SE uses EOS blockchain, which is the current high-performance public chain. Moreover, it considers more practical scenarios, such as SESP and DU may be dishonest. So, our scheme has better practicality.

6.2.2. Storage Analysis. BFR-SE is a fair and reliable searchable encryption scheme based on the EOS blockchain. Since the storage resource on the blockchain is very valuable, and the acquisition of RAM in the EOS blockchain requires the user to mortgage the system token, it is necessary to analyze the size of the data stored on-chain.

In the beginning, we define some symbols. We set $|G_0|$, $|G_1|$ to represent the bit length of an element in group G_0 and G_1, respectively. Let $|Z_p|$ be the bit length of an element in field Z_p, $|Sk_{\text{sig}}|$ be the bit length of the signature, $|S|$ be the number of all attributes, $|U|$ be the number of DUs, |TOK| be the bit length of search token, |COMM| be the bit length of commitment, $|\text{CI}_{\text{result}}|$ be the bit length of the result returned by SESP. Let m be the size of the keyword set.

According to the experiment simulation in our scheme, we set $|G_0| = |G_1| = 1024$ bits, $|Z_p| = 128$ bits, $|Sk_{\text{sig}}| = 576$ bits, $|Sk_{\text{com}}| = |Sk_{\text{sig}}| = 256$ bits $|Pk_{\text{com}}| = |Pk_{\text{sig}}| = 272$ bits. The length of *Account*, *SerialNum*, *blockheight*, *Deposit*, and *Coin* are all 64 bits. The implementation of our Bloom filter refers to the C++ code on Github, which address is https://github.com/bbondy/bloom-filter-cpp.git. We set the length of the bloom filter to 20 KB and there are 5000 rows of ciphertext indices, so $l_{\text{BF}}/n = (20 \times 8 \times 1024)/5000 = 32.768$ and the number of hash functions in Bloom Filter will be $k = (\ln 2)\, l_{\text{BF}}/n = 22$. Then, the false-positive will be 1.45×10^{-7}. In our scheme, three operations that will interact with blockchain to store data in the smart contract, which are as follows:

(1) Initialization

DO uploads the public system parameters and his public key for the signature to smart contract. The storage cost is as follows:

$$3|G| + |G||S| + |Pk_{\text{Sig}}|. \tag{36}$$

(2) Registration

The information on-chain mainly includes registration-related information uploaded by DU and SESP. The DU registration includes the information submitted by DU and the private key distributed by DO. The SESP registration includes the account and the deposit. The specific storage overhead is as follows:

$$(|\text{account}| + |Pk_{\text{com}}| + 2|G| + |S||G|)|U| + (|\text{account}| + |\text{Deposit}|). \tag{37}$$

(3) Search

The information on-chain mainly includes auxiliary information uploaded by DO, search requests initiated by DUs, and search results returned by SESP. The specific storage overhead is as follows:

$$l_{\text{BF}} + (|\text{account}| + |\text{SerialNum}| + |\text{TOK}| + |\text{COMM}| + |\text{height}| + |\text{coin}|)|U| + (|\text{account}| + |\text{SerialNum}| + |\text{CI}_{\text{result}}|)|U|. \tag{38}$$

The storage overhead of BFR-SE varies with the number of attributes is shown in Figure 4.

For simplicity, the figure only shows that the storage overhead varies with the number of attributes when there are only 10 DUs and 50 keywords. From the figure, we can see that the storage overhead is mainly spent in the search phase, while the initialization and registration phase are negligible. As the number of DUs and keywords grows, storage overhead will also increase linearly. However, since we only store some information necessary for verification on-chain, compared with other blockchain-based schemes [24, 26–30] that store all index information on-chain, the storage overhead is reduced a lot.

Users can obtain the RAM in EOS by collateralizing system tokens, and the current price is 42EOS/MB. DO can purchase RAM according to the scale of his system. Unlike Ethereum transactions that need to consume ETH as gas, the tokens mortgaged when acquiring RAM in EOS can still be redeemed at the original price. Above all, BFR-SE is feasible and practical.

6.2.3. Performance Analysis. Before analyzing the performance, we define two primary operations' computational cost: P for bilinear pairing and E for power exponent operation. Here we ignore the computational overhead of operations such as hash functions because they are very efficient than the above two. In our proposed scheme, the computational overhead of the primary operations is shown in Table 3.

There are many studies on the analysis mentioned above [38–43], so we will not repeat them too much. Like storage resource, the computing resource on-chain is also very valuable. If the interaction with the blockchain is too frequent or the computational overhead is too large, it will have a terrible impact on system performance. So, we mainly focus on the execution time of BFR-SE on-chain.

We used 6 nodes to build an EOS private chain in a laboratory environment. The 6 nodes we chose were all

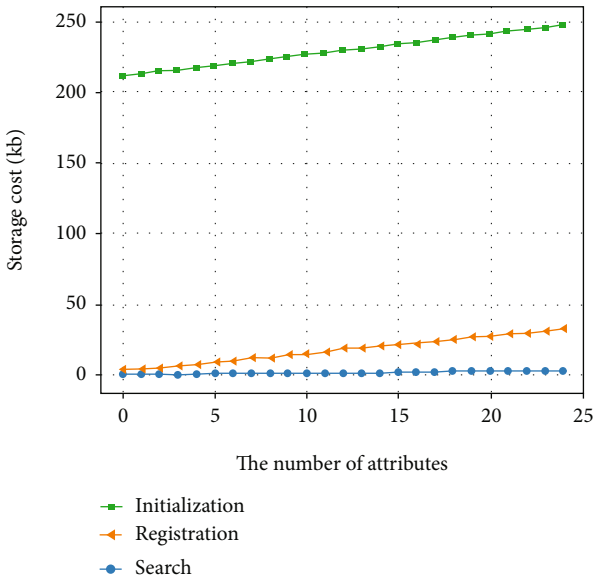

Figure 4: The storage overhead of BFR-SE varies with the number of attributes.

Table 3: The computational overhead of main operations in BFR-SE.

Operation	Computational overhead
Initialization	$(3+\|S\|)E$
Key generation	$(2+\|S\|)E$
Keyword encryption	$(2+3\|S\|+m)E$
Search token generation	$(4+\|S\|+m)E$
Search	$4P$

MacBook Pro (2017) with Intel (R) Core (TM) i5 CPU that clocks at 3.1 GHz and has 8 GB of RAM. The version of the EOS blockchain we chose is v2.0.7. The computational overhead of other blockchain-based schemes [24, 26, 27, 30] is all second level, which is obviously not better than ours, and will not be analyzed anymore. We compared BFR-SE with the scheme in Ref. [25], as shown in Figure 5.

In BFR-SE, most of the interactions with the blockchain are to upload data to smart contract, such as initialization and registration. The computational overhead of this part can be ignored. The main computational overhead of BFR-SE occurs when the blockchain arbitrates. It can be seen from the figure that as the number of indices to increase, the computational overhead on-chain of BFR-SE has always remained at a stable level, about 40 ms, while the scheme in Ref. [25] will grow. It is because, in our scheme, all the time-consuming operations are executed off-chain. The EOS block producers' configuration in the Mainnet is much higher than that of the MacBook used in our simulation environment. When our contracts are deployed on the Mainnet, the performance will be even more outstanding. The EOS blockchain generates a block in 0.5 seconds, and the transaction will be confirmed soon after execution.

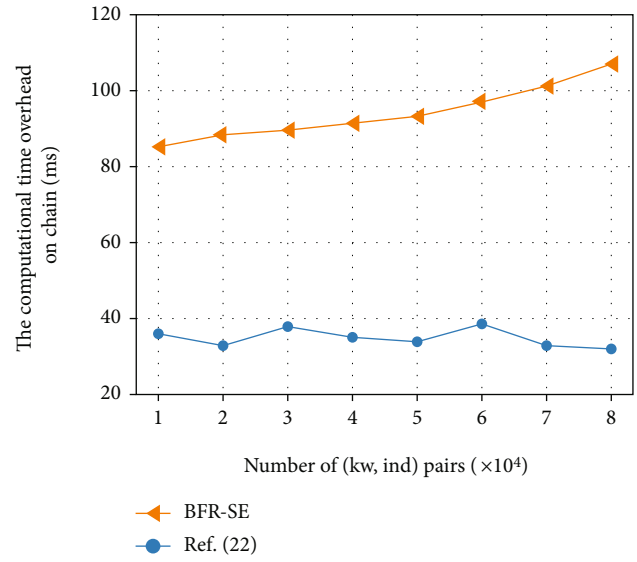

Figure 5: The computational overhead of BFR-SE and Ref. [25] varies with the number of indices.

Therefore, compared with Ref. [25], BFR-SE has a better performance.

7. Conclusion

To achieve a fair and reliable searchable encryption scheme, we constructed an attribute-based searchable encryption ABSE that supports multiple keywords search and designed an exclusive reward and punishment mechanism by using blockchain. In our scheme, DO sends the ciphertext of indices to SESP and uploads the auxiliary information to the blockchain. SESP must return the correct search results before a preagreed block height, and the charge fee paid from DU will be frozen for a period during which DU could initiate an arbitration request to the blockchain if he disagrees with the results. As the cornerstone of trust, blockchain will punish the dishonest party economically, ensuring the scheme's absolute fairness and reliability [44–53]. Besides, ABSE can be used by DO to have fine-grained access control on the search function. Experiments and analyses show that our scheme is feasible and has better performance. However, our scheme still has many shortcomings. For example, our scheme uses an index structure, and the signature guarantees the integrity of the indices, but this significantly reduces the flexibility of the scheme, especially when adding or updating the index of sharing data. Simultaneously, due to an attribute-based encryption algorithm, topics such as the revocation or update of permission are also one of the directions that need to be studied in the future. We will continue to refine our approach in conjunction with some other research [37, 54–57].

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Fundamental of China under Grants 61272519, 61170297, 61472258, and 61802094, China Postdoctoral Science Foundation under Grant 2021M690305, and National Natural Science Foundation of Zhejiang Province under Grant LY20F020012.

References

[1] Y. Xu, X. Yan, Y. Wu, Y. Hu, W. Liang, and J. Zhang, "Hierarchical bidirectional RNN for safety-enhanced B5G heterogeneous networks," *IEEE Transactions on Network Science and Engineering*, 2021.

[2] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing.*, vol. 15, no. 4, pp. 577–590, 2018.

[3] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[4] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *The 39th IEEE International Conference on Distributed Computing Systems*, pp. 144–153, 2019.

[5] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[6] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.

[7] X. Zhou, X. Yang, J. Ma, and K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet of Things Journal*, 2021.

[8] X. Yan, J. Zhang, H. Elahi, M. Jiang, and H. Gao, "A personalized search query generating method for safety-enhanced vehicle-to-people networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5296–5307, 2021.

[9] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P*, pp. 44–55, 2000.

[10] B. Dan, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 2-6, 2004.

[11] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions," in *CRYPTO'05: Proceedings of the 25th annual international conference on Advances in Cryptology*, pp. 205–222, Berlin, Heidelberg, 2005.

[12] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[13] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multikeyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM 2014- IEEE Conference on Computer Communications*, pp. 2112–2120, 2014.

[15] S. Tahir, S. Ruj, Y. Rahulamathavan, M. Rajarajan, and C. Glackin, "A new secure and lightweight searchable encryption scheme over encrypted cloud data," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 4, pp. 530–544, 2019.

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

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

[18] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," in *FC 2012: Financial Cryptography and Data Security*, vol. 7397, pp. 285–298, 2012.

[19] X. Zhu, Q. Liu, and G. Wang, "Verifiable dynamic fuzzy search over encrypted data in cloud computing," in *The 15th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2015)*, vol. 9530, pp. 655–666, 2015.

[20] M. Azraoui, K. Elkhiyaoui, M. Onen, and R. Molva, "Publicly verifiable conjunctive keyword search in outsourced databases," in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 619–627, 2015.

[21] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE INFOCOM 2014- IEEE Conference on Computer Communications*, pp. 522–530, 2014.

[22] M. H. Ameri, M. R. Asaar, J. Mohajeri, and M. Salmasizadeh, *A generic construction for verifiable attribute-based keyword search schemes*, IACR Cryptology ePrint Archive, 2015.

[23] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1187–1198, 2016.

[24] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using blockchain," 2017, https://arxiv.org/abs/1711.01030.

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

[26] 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*, pp. 792–800, 2018.

[27] 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, 2021.

[28] Q. Tang, "Towards blockchain-enabled searchable encryption," in *Information and Communications Security*, pp. 482–500, Springer International Publishing, Cham, 2020.

[29] 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.

[30] S. Jiang, J. Cao, J. A. McCann et al., "Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 405–410, 2019.

[31] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[32] V. Buterin, *Ethereum: a next-generation smart contract and decentralized application platform*, white paper, 2013.

[33] N. Szabo, "Smart contracts," 1994, http://szabo.best.vwh.net/smart.contracts.idea.html.

[34] D. Larimer, "Eos.io technical white paper," 2017, https://steemit.com/eos/@eosio/eos-io-technical-white-paper.

[35] H. Gao, Z. Ma, S. Luo, and Z. Wang, "BFR-MPC: a blockchain-based fair and robust multi-party computation scheme," *IEEE Access*, vol. 7, pp. 110439–110450, 2019.

[36] H. Gao, Z. Ma, S. Luo, Y. Xu, and Z. Wu, "BSSPD: a blockchain-based security sharing scheme for personal data with fine- grained access control," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, Article ID 6658920, 20 pages, 2021.

[37] J. Zhang, Y. Yan, Z. Cheng, and W. Wang, "Lightweight attention pyramid network for object detection and instance segmentation," *Applied Sciences*, vol. 10, no. 3, pp. 883–899, 2020.

[38] X. Zhou, X. Xu, W. Liang et al., "Intelligent small object detection based on digital twinning for smart manufacturing in industrial CPS," *IEEE Transactions on Industrial Informatics*, 2022.

[39] Y. Xu, Z. Liu, C. Zhang, J. Ren, Y. Zhang, and X. Shen, "Blockchain-based trustworthy energy dispatching approach for high renewable energy penetrated power systems," *IEEE Internet of Things Journal*, 2021.

[40] X. Zhou, Y. Li, and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 912–921, 2021.

[41] X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, and C. Li, "Learning URL embedding for malicious website detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6673–6681, 2020.

[42] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2021.

[43] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, "A method of information protection for collaborative deep learning under GAN model attack," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 871–881, 2021.

[44] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1202–1213, 2021.

[45] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Transactions on Cloud Computing*, 2021.

[46] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1421–1432, 2021.

[47] Y. Xu, Q. Zeng, G. Wang, C. Zhang, J. Ren, and Y. Zhang, "An efficient privacy-enhanced attribute-based access control mechanism," *Concurrency & Computation: Practice & Experience*, vol. 32, no. 5, pp. 1–10, 2020.

[48] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, "Deep-learning-enhanced multitarget detection for end-edge-cloud surveillance in smart IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12588–12596, 2021.

[49] C. Zhang, Z. Ni, Y. Xu, E. Luo, L. Chen, and Y. Zhang, "A trustworthy industrial data management scheme based on redactable blockchain," *Journal of Parallel and Distributed Computing*, vol. 152, pp. 167–176, 2021.

[50] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

[51] J. Zhang, M. Z. A. Bhuiyan, X. Yang et al., "AntiConcealer: reliable detection of adversary concealed behaviors in EdgeAI assisted IoT," *IEEE Internet of Things Journal*, 2021.

[52] J. Zhang, M. Z. A. Bhuiyan, Y. Xu, A. K. Singh, D. F. Hsu, and E. Luo, "Trustworthy target tracking with collaborative deep reinforcement learning in EdgeAI-aided IoT," *IEEE Transactions on Industrial Informatics*, 2022.

[53] G. Yu, X. Zha, X. Wang et al., "Enabling attribute revocation for fine-grained access control in blockchain-IoT systems," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1213–1230, 2020.

[54] X. Yan, Y. Xu, X. Xing, B. Cui, Z. Guo, and T. Guo, "Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6182–6192, 2020.

[55] Y. Wang, T. Li, H. Qin et al., "A brief survey on secure multiparty computing in the presence of rational parties," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 6, pp. 807–824, 2015.

[56] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.

[57] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology, ser. EUROCRYPT'11*, pp. 568–588, Berlin, Heidelberg, 2011.