6-2018

TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain

Yinghui ZHANG

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Jiangang SHU

Kan YANG

Dong ZHENG

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 26, 2018, accepted June 1, 2018, date of publication June 6, 2018, date of current version June 26, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2844400

TKSE: Trustworthy Keyword Search Over Encrypted Data With Two-Side Verifiability via Blockchain

YINGHUI ZHANG[1,2,3], (Member, IEEE), ROBERT H. DENG[2], (Fellow, IEEE), JIANGANG SHU[4], KAN YANG[5], (Member, IEEE), AND DONG ZHENG[1,3]

[1]National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[2]School of Information Systems, Singapore Management University, Singapore 188065
[3]Westone Cryptologic Research Center, Beijing 100070, China
[4]Department of Computer Science, City University of Hong Kong, Hong Kong
[5]Department of Computer Science, University of Memphis, Memphis, TN 38152, USA

Corresponding authors: Yinghui Zhang (yhzhaang@163.com) and Jiangang Shu (jgshu2-c@my.cityu.edu.hk)

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802000, in part by the AXA Research Fund, in part by the National Natural Science Foundation of China under Grants 61772418, 61472472, and 61402366. The work of Y. Zhang was supported by the New Star Team of Xi'an University of Posts and Telecommunications under Grant 2016-02.

ABSTRACT As a very attractive computing paradigm, cloud computing makes it possible for resource-constrained users to enjoy cost-effective and flexible resources of diversity. Considering the untrustworthiness of cloud servers and the data privacy of users, it is necessary to encrypt the data before outsourcing it to the cloud. However, the form of encrypted storage also poses a series of problems, such as: How can users search over the outsourced data? How to realize user-side verifiability of search results to resist malicious cloud servers? How to enable server-side verifiability of outsourced data to check malicious data owners? How to achieve payment fairness between the user and the cloud without introducing any third party? Towards addressing these challenging issues, in this paper, we introduce TKSE, a trustworthy keyword search scheme over encrypted data without any third party, trusted or not. In TKSE, the encrypted data index based on digital signature allows a user to search over the outsourced encrypted data and check whether the search result returned by the cloud fulfills the pre-specified search requirements. In particular, for the first time, TKSE realizes server-side verifiability which protects honest cloud servers from being framed by malicious data owners in the data storage phase. Furthermore, blockchain technologies and hash functions are used to enable payment fairness of search fees without introducing any third party even if the user or the cloud is malicious. Our security analysis and performance evaluation indicate that TKSE is secure and efficient and it is suitable for cloud computing.

INDEX TERMS Blockchain, cloud computing, fair payment, searchable encryption, verifiability.

I. INTRODUCTION

Cloud computing has attracted an increasing number of individuals and organizations to outsource their data to third-party platforms for infinite computing and storage capacity available on demand with inexpensiveness and convenience [1], [2]. Considering the untrustworthiness of cloud servers and the data privacy of users, it is necessary to encrypt the data before outsourcing it to the cloud. However, the direct use of traditional encryption technologies deprives users of search capability and thus results in a poor user experience. In order to preserve the search functionality, searchable encryption technologies have been developed in two representative settings including the symmetric-key setting [3] and the public-key setting [4]. In this paper, we focus on the former, that is, searchable symmetric encryption (SSE), which is usually more practical than the latter in terms of efficiency. However, many important and challenging issues have not been well studied in SSE, such as *user-side verifiability* and *server-side verifiability* as illustrated in Figure 1, *fair payment*, and *no trusted third party*.

In fact, user-side verifiability takes into consideration that the cloud server may be malicious, that is, the cloud server may only return part of search results or maliciously return incorrect results. The issue of user-side verifiability is firstly

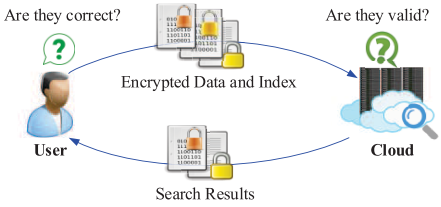


FIGURE 1. The scenario of two-side verifiability in SSE.

addressed in [5] and [6]. However, these two schemes cannot support server-side verifiability and fair payment without any trusted third party. Furthermore, server-side verifiability takes into consideration that the data owner may be malicious, that is, the data owner may maliciously outsource invalid data in the data storage phase and fraudulently claim compensation later. This concern has not been addressed and even has received little attention in the literature. Last but not least, most of the previous schemes are bank-dependent. Specifically, either the payment issue is not considered or the default traditional payment mechanism is exploited in which a trusted third party (TTP) such as a trustworthy bank has to be introduced for payment fairness. Payment fairness can promote the honest behaviors of users and cloud servers [7]. If a malicious behavior is detected based on the user-side verifiability (resp. server-side verifiability), the data owner (resp. cloud server) should get adequate compensation from the cloud server (resp. data owner) no matter what the cloud server (resp. data owner) does. Therefore, fair payment without any third party is a meaningful and challenging task and it remains in SSE.

In order to throughly address the aforementioned challenging issues in cloud computing, we propose TKSE, a **T**rustworthy **K**eyword **S**earch scheme over **E**ncrypted data without needing any third party. TKSE is proven secure and our performance evaluation shows its efficiency. In particular, TKSE is characterized by the following desirable features.

- *Keyword Search over Encrypted Data.* The encrypted data index based on the Elliptic Curve Digital Signature Algorithm (ECDSA) allows a user to search over the outsourced encrypted data.
- *User-side Verifiability.* In TKSE, a data owner can embed search requirements into the output script of a joint transaction such that the transaction can be redeemed by the cloud server if and only if the output script evaluates to true based on the returned search result. Therefore, TKSE enables the data owner to resist malicious cloud servers and user-side verifiability is realized.
- *Server-side Verifiability.* Similar to user-side verifiability, the public verification of digital signature enables the cloud server to check the validness of the outsourced encrypted data from the data owner in the data storage phase. Thus, malicious data owners can be detected by the cloud server, which realizes server-side verifiability.
- *Fair Payment and No TTP.* Based on hash functions and ECDSA, TKSE is compatible with blockchains such as the Bitcoin blockchain and the Ethereum blockchain. The global consensus and distributed nature of a blockchain enable a fair payment mechanism in TKSE without introducing any TTP.

The rest of the paper is organized as follows. The related work is discussed in Section II. Some preliminaries are reviewed in Section III. We then formulate problems in Section IV. In Section V, we describe the proposed TKSE scheme in detail. Security analysis and performance evaluation are given in Section VI. Finally, concluding remarks are made in Section VII.

II. RELATED WORK

In recent years, cloud computing technologies have gotten rapid developments and a line of studies have been done on security issues in cloud computing, such as access control [8]–[13] and privacy protection [14]–[19]. As a typical service in cloud computing, cloud storage needs both data security and search functionality. To realize keyword search on remote encrypted data, Song *et al.* [3] proposed the first searchable encryption scheme in the symmetric setting, in which each keyword is encrypted independently under a two-layer encryption construction. Boneh *et al.* [4] proposed the first public-key based searchable encryption scheme. Li *et al.* proposed [20] the first fuzzy keyword search over encrypted data in cloud computing. Curtmola *et al.* [21] first considered the adaptive security of SSE and presented two index-based schemes. Li *et al.* [22]m proposed a lightweight framework for privacy-aware data queries in cloud computing. However, all the above schemes assume that the cloud server is honest and the returned search result is correct without considering user-side verifiability. In order to resist selfish cloud servers, Kurosawa and Ohtaki [5] first proposed a SSE scheme with user-side verifiability based on [21]. User-side verifiability has also been realized in SSE [6], fuzzy SSE [23], attribute-based keyword search [24], and database update [25], [26]. In these schemes, however, malicious data owners may outsource invalid encrypted data to the cloud server for fraudulent compensation in the future. Similar issues also exist in secure deduplication and data integrity in cloud computing [27]–[29]. Server-side verifiability can be used to tackle this problem but it has not been found in the literature. In addition, all the above schemes are bank-dependent. To be specific, either the payment issue is not taken into account or a TTP is inevitably introduced for fair payment.

To solve the payment fairness issue in cloud computing, blockchain technologies have been taken into account [30], [31]. Andrychowicz *et al.* [32] proposed secure multiparty lottery protocols based on the Bitcoin blockchain, in which a bitcoin-based timed commitment scheme serves as the main ingredient. In order to realize more general computation, Andrychowicz *et al.* [33] proposed a two-party computation protocol, which is not compatible with the

Bitcoin blockchain because it modifies the Bitcoin specification to resist malleability attacks. Similar ideas were developed independently by Bentov and Kumaresan [34]. Li *et al.* [35] proposed a blockchain-based SSE scheme, which requires the miners of the blockchain to do some particular tasks, such as decryption, and hence violates the Bitcoin specification. Many other blockchain-based SSE schemes can be found in [36]–[39], in which the server-side verifiability remains to be addressed. Besides, blockchain technologies have also been used in outsourcing computation [40], accountable storage [41] and is combined with attribute-based signature [42] and homomorphic signature [43]. It is noticeable that all these schemes are not compatible with the Bitcoin blockchain, where the scripts are very limited [44]. For example, the string concatenation is currently disabled in Bitcoin scripts.

III. PRELIMINARIES

A. NOTATIONS

In Table 1, we summarize important notations used in TKSE for ease of reference.

TABLE 1. Notations used in TKSE.

Notation	Meaning
User	The data owner and consumer, a.k.a. the user.
CSP	The cloud service provider.
$x \in_R X$	An element x is randomly chosen from the set X.
N	The number of documents.
$[m]$	The set $\{i \in \mathbb{Z}^+ \mid 1 \leq i \leq m\}$ with $m \in \mathbb{Z}^+$.
r_S	A secret of the cloud service provider.
h_S	The hash value $H(r_S)$, where H is a hash function.
t	A time-lock.
$\mathcal{T}$	A data tree.
ℓ	The height of $\mathcal{T}$.
$\mathcal{D}$	A set of documents $\mathcal{D} = \{D_i\}_{i\in[N]}$.
$\mathcal{C}$	The ciphertext data set of $\mathcal{D}$ and $\mathcal{C} = \{C_i\}_{i\in[N]}$. It is also known as encrypted data set.
$\mathcal{I}$	The index set with respect to $\mathcal{D}$.
$\mathcal{W}$	A set of keywords with respect to $\mathcal{D}$.
$\mathcal{D}_\omega$	The set of documents which contain a keyword $\omega \in \mathcal{W}$.
$\mathcal{C}_\omega$	The ciphertext data set of $\mathcal{D}_\omega$.
(pk_A, sk_A)	An ECDSA public and secret key pair of A.
σ_{root}	The ECDSA signature of the root of $\mathcal{T}$.
$\max_B$	The maximal delay between broadcasting a transaction and including it on the blockchain.

B. BLOCKCHAIN

As an essential technology behind Bitcoin [30] and Ethereum [31], the idea of blockchain is that the longest chain is accepted as the proper one. In TKSE, the bitcoin-based timed commitment scheme [32] is used to resist malicious behaviors, which is also adopted by [40] and [41]. The commitment scheme is denoted by CS(User, CSP, d, t, s) and is executed between User and CSP. Here, User and CSP represent the user and the cloud service provider in TKSE. They act as the committer and the receipt in the commitment scheme, respectively. Concretely, CSP commits to a secret s and has to open the commitment before a specific time t to get his/her deposit of value d ₿ back. Otherwise, the deposit will be redeemed by User. The commitment scheme consists of three phases: the commitment phase CS.Commit(CSP, User, d, t, s), the opening phase CS.Open(CSP, User, d, t, s) and the punishment phase CS.Fine(CSP, User, d, t, s). Note that the punishment phase is performed only if the opening phase is not correctly performed. Please refer to [32] for more details.

C. SPECIFICATION OF TWO-SIDE VERIFIABLE SSE

A two-side V-SSE scheme is composed of seven polynomial time algorithms (KeyGen, Encrypt, BuildIndex, Trapdoor, Search, Verify, Decrypt) as below. The specification is the same as the one given in [5] except that index generation is separated from data encryption for clarity and the index is publicly verifiable.

- KeyGen(1^λ) $\rightarrow K$: Given a security parameter, it outputs a secret key K.
- Encrypt($K, \mathcal{D}$) $\rightarrow \mathcal{C}$: Given K, a document set $\mathcal{D}$, it outputs a ciphertext data set $\mathcal{C}$.
- BuildIndex($K, \mathcal{D}, \mathcal{C}, \mathcal{W}$) $\rightarrow \mathcal{I}$: Given K, $\mathcal{D}$, $\mathcal{C}$ and a keyword set $\mathcal{W}$, it outputs an index set $\mathcal{I}$. Note that $\mathcal{C}$ and $\mathcal{W}$ can be generated based on K and $\mathcal{D}$. We explicitly take them as inputs for the simplicity of exposition.
- Trapdoor(K, ω) $\rightarrow T_\omega$: Given K and a keyword ω, it outputs a trapdoor T_ω associated with ω.
- Search($\mathcal{I}, \mathcal{C}, T_\omega$) $\rightarrow (\mathcal{C}_\omega, \mathsf{tag}_\omega)$: Given $\mathcal{I}$, $\mathcal{C}$ and T_ω, it outputs a ciphertext data set $\mathcal{C}_\omega$ and a tag set tag_ω associated with ω.
- Verify($T_\omega, \mathcal{C}_\omega, \mathsf{tag}_\omega$) $\rightarrow$ `1` or `0`: Given $T_\omega, \mathcal{C}_\omega$ and tag_ω, it outputs `1` if the search result is valid and `0` otherwise.
- Decrypt($K, \mathcal{C}$) $\rightarrow \mathcal{D}$: Given $K, \mathcal{C}$, it outputs $\mathcal{D}$.

In Section V, the design of TKSE is described in detail. We will see that an underlying two-side V-SSE scheme is used in TKSE, in which KeyGen, Encrypt and Decrypt are based on a symmetric encryption algorithm, and verifiable BuildIndex is based on a digital signature.

IV. PROBLEM FORMULATION

A. SYSTEM MODEL

As shown in Figure 2, the entities involved in TKSE include a data owner, a data consumer, a cloud service provider (i.e., CSP) and a blockchain. In TKSE, we focus on searchable symmetric encryption. In other words, the data owner and the data consumer are essentially the same entity (i.e., User) because they share a secret key. Suppose User intends to outsource some documents to CSP while keeping data confidentiality, searching ability and verifiability. The main procedures are presented in Figure 2. Obviously, TKSE consists of three phases including data storage phase, data search phase and user claim phase. In the data storage phase, User generates encrypted data and index by procedure (1) and outsources them to CSP. Based on procedure (2), data storage is enforced by CSP after ensuring its validness. Procedures (3.1) and (3.2) are used by User to confirm the enforcement of data storage. On the other hand, data search is enabled by procedures (4), (5.1),

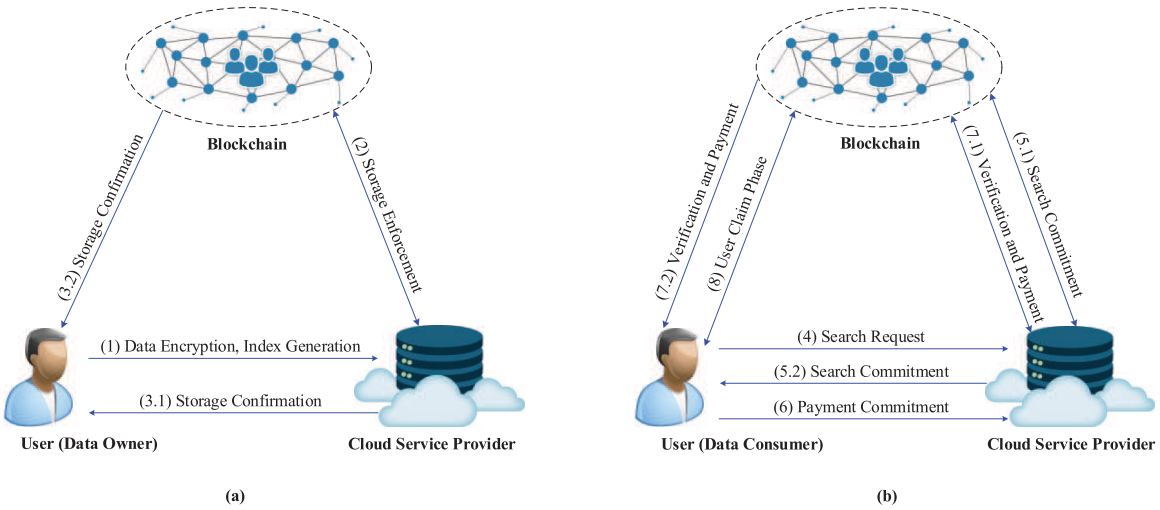


FIGURE 2. The system model of TKSE. (a) Data Storage Phase. (b) Data Search and User Claim Phase.

(5.2), (6), (7.1) and (7.2). The search request is based on procedure (4) and the search commitment is realized in procedures (5.1) and (5.2). The payment commitment is based on procedure (6). Procedures (7.1) and (7.2) are used to realize verification and payment. Finally, procedure (8) allows User to claim compensation due to search payment if the search result is invalid in the data search phase. In TKSE, a public blockchain is considered, such as the Bitcoin blockchain and the Ethereum blockchain.

B. ADVERSARY MODEL AND DESIGN GOALS

In TKSE, both User and CSP can be malicious and they are of mutual distrust. Specifically, the following malicious behaviors should be taken into consideration.

- **Case 1.** Malicious CSP may try to learn information on secret keys and original documents of User from trapdoor and ciphertext data.
- **Case 2.** Malicious CSP may forge search results or return partial results.
- **Case 3.** Malicious User may defraud CSP for compensation by outsourcing invalid ciphertext data and index to CSP in the data storage phase.
- **Case 4.** Malicious CSP aims to obtain search fees from User without providing search results satisfying the requirements of User.
- **Case 5.** Malicious User wants to get valid search results from CSP without paying the search fee.

Considering the above attacks, the design goals of TKSE are as follows.[1]

- **Privacy.** In TKSE, it is impossible for CSP to learn any information on secret keys and original documents. This goal is against the attack in **Case 1**.
- **User-side Verifiability.** It is the same as the property *reliability* in [5]. This goal is against the attack in **Case 2**. That is, CSP cannot forge valid search results or return partial results.
- **Server-side Verifiability.** This goal is against the attack in **Case 3** and it can protect CSP from being framed by malicious User.
- **Fair Payment.** This goal is against attacks in **Case 4** and **Case 5**. Specifically, fairness ensures that User can get valid search results only if corresponding search fees are paid to CSP. On the other hand, CSP obtains search fees from User only if satisfactory search results are returned.

In addition, considering that blockchain technologies are used in TKSE, the following attractive properties should be achieved.

- **No TTP.** It means that payment fairness is realized without introducing any TTP.
- **Compatibility.** TKSE should be compatible with mainstream blockchains such as Bitcoin and Ethereum. Note that if TKSE is compatible with the Bitcoin blockchain, then it is compatible with Ethereum blockchains [44], [45].

C. DEFINITION OF TKSE

TKSE is composed of four phases including the *initialization phase*, the *data storage phase*, the *data search phase* and the *user claim phase*. The first three phases are compulsory and the *user claim phase* is performed by User when CSP is malicious. The definition of TKSE is as follows.

1) INITIALIZATION

User and CSP initialize parameters to be used in the subsequent phases, such as their own secret keys and unredeemed transactions on the blockchain.

[1] As shown in Section VI, TKSE achieves privacy and user-side verifiability in the universal composability framework similar to [5].

2) DATA STORAGE PHASE

The outsourcing of data storage is implemented in this phase. Four procedures, data encryption, index generation, storage enforcement and storage confirmation, are sequentially performed as follows.

- **Data Encryption:** For confidentiality, User encrypts $\mathcal{D}$ to get $\mathcal{C}$ which is outsourced to CSP.
- **Index Generation:** User generates $\mathcal{I}$ and sends it together with $\mathcal{C}$ to CSP.
- **Storage Enforcement:** CSP firstly checks and stores $\mathcal{I}$ and $\mathcal{C}$. Then, CSP generates a digital signature according to $\mathcal{C}$ and stores the signature on the blockchain. Finally, CSP sends such information to User that helps User to get the signature from the blockchain.
- **Storage Confirmation:** Upon getting the signature from the blockchain, User thinks that $\mathcal{C}$ has been stored by CSP.

3) DATA SEARCH PHASE

In this phase, CSP searches his/her storage according to the requirements of User. It can be ensured that CSP earns search fees from User if and only if User obtains valid search results. Four sequential sub-phases are performed, including search request, search commitment, payment commitment, and verification and payment.

- **Search Request:** User generates a search request based on $\mathcal{W}$ and sends it to CSP. Besides, the compensation and penalty of CSP in the case of invalid search should be specified by User and CSP.
- **Search Commitment:** This phase is used by CSP to commit to that if search results cannot meet the requirements specified in *Search Request* and malicious CSP refuses to compensate User in the subsequent phase *Verification and Payment* before a given time, User is able to claim enough deposits of CSP by himself/herself as a penalty in the subsequent *User Claim Phase* after the given time.
- **Payment Commitment:** User commits to CSP that if the search result is valid, CSP is able to earn search fees in *Verification and Payment*. It is realized by temporarily freezing a deposit consists of service fee from User, in which the search requirements are specified.
- **Verification and Payment:** To demonstrate that he/she agrees with the search requirements of User, CSP further freezes a deposit consists of his/her guaranty.[2] Then, CSP earns the service fee of User by providing search results and proving that the search result meets the requirements of User. Finally, User gets search results from the proof of CSP and decrypts them to obtain original documents.

[2]At this time, honest User is capable of ensuring that either a valid search result is obtained later by paying the service fee or enough deposits are claimed in *User Claim Phase* no matter how CSP behaves. Honest CSP can ensure that if the search result is valid he/she will earn the service fee no matter how User behaves.

4) USER CLAIM PHASE

Only if CSP fails to prove that the search result meets the requirements of User before the given time, TKSE comes to *User Claim Phase*. It is used by User to claim enough deposits of CSP no matter how CSP behaves.

V. TRUSTWORTHY SEARCH OVER ENCRYPTED DATA

In this section, we first give an overview of TKSE, and then describe its design in detail.

A. HIGH LEVEL DESCRIPTION

In TKSE, privacy is realized similar to the SSE scheme [5]. The scheme [5] realizes user-side verifiability by designing indexes based on a message authentication code (MAC) with a secret key. Because the MAC secret key is only known by User, the server-side verifiability cannot be achieved in [5]. Furthermore, the idea in [5] cannot be directly combined with Blockchain technologies in that the condition of redeeming search fees should be specified by User and CSP and it requires the MAC secret key. If CSP gets the secret key, he/she can generate "search results" arbitrarily and ensure they can pass the user-side verifiability, which violates payment fairness. In TKSE, we build the index based on a signature scheme and hash functions instead of MAC. In this case, although CSP is able to publicly check the search results, he/she cannot forge valid search results arbitrarily because of the unforgeability of the digital signature and the collision-resistance of hash functions. In order to eliminate TTP and realize compatibility with typical blockchains especially the Bitcoin blockchain, we exploit ECDSA in TKSE.

B. DESIGN DETAILS OF TKSE

1) INITIALIZATION

Let π_0 be a pseudo-random permutation,

$$\pi_0 : \{0,1\}^{\lambda} \times \{0,1\}^{1+\ell_w+\ell} \to \{0,1\}^{1+\ell_w+\ell},$$

where λ is a security parameter, ℓ_w is the bit length of a keyword, and ℓ represents the bit length of a document with $N = 2^{\ell}$. User chooses $K_0 \in \{0,1\}^{\lambda}$ as a key of π_0 and defines $\pi(\cdot) = \pi_0(K_0, \cdot)$. In addition, User chooses a secure cryptographic hash function H and a symmetric encryption scheme with functions

$$(\mathsf{KeyGen}_{\mathrm{SE}}, \mathsf{Encrypt}_{\mathrm{SE}}, \mathsf{Decrypt}_{\mathrm{SE}}),$$

and sets $K_{\mathrm{SE}} = \mathsf{KeyGen}_{\mathrm{SE}}(1^{\lambda})$. Finally, we assume User and CSP choose their own ECDSA public and secret key pairs, denoted by (pk_U, sk_U) and (pk_S, sk_S), respectively. CSP prepares an unredeemed transaction $\mathsf{Tx}^{S}_{\mathrm{sig}}$ of value d_{sig} ₿, which can be redeemed with sk_S. User sets $K = (K_0, K_{\mathrm{SE}}, sk_U)$ as his/her secret key.

2) DATA STORAGE PHASE

The outsourcing of $\mathcal{D}$ is implemented based on the following four procedures.

Algorithm 1 BuildIndex$(K, \mathcal{D}, \mathcal{C}, \mathcal{W}) \to \mathcal{I}$

Input: secret key K, document set $\mathcal{D}$, ciphertext data set $\mathcal{C}$, keyword set $\mathcal{W}$
Output: index $\mathcal{I}$

for $i \in [\mathsf{num}]$ **do**
  $\mathcal{I}(i) = (\mathtt{dummy}, \mathsf{sig}_U(i \parallel H(\mathtt{dummy})))$
end
for $\omega \in \mathcal{W}$ **do**
  Suppose $\mathcal{D}_\omega = \{D_{s_1}, D_{s_2}, \ldots, D_{s_m}\}$.
  for $j \in [m]$ **do**
    $\mathsf{addr}_{\omega,j} = \pi(0, \omega, j)$
    $\mathsf{tag}_{\omega,j} = \mathsf{sig}_U(\mathsf{addr}_{\omega,j} \parallel H(C_{s_j}))$,
    $\mathcal{I}(\mathsf{addr}_{\omega,j}) = (s_j, \mathsf{tag}_{\omega,j})$
  end
end
for $D_k \in \mathcal{D}$ **do**
  Suppose k has already appeared N_k times in $\mathcal{I}$.
  if $N_k \neq 2^{\ell_w}$ **then**
    for $1 \le r \le 2^{\ell_w} - N_k$ **do**
      $\mathsf{addr}_{r,k} = \pi(1, r, k)$
      $\mathsf{tag}_{r,k} = \mathsf{sig}_U(\mathsf{addr}_{r,k} \parallel H(C_k))$
      $\mathcal{I}(\mathsf{addr}_{r,k}) = (k, \mathsf{tag}_{r,k})$
    end
  end
  Now, the index k appears 2^{ℓ_w} times in $\mathcal{I}$.
end
return $\mathcal{I}$

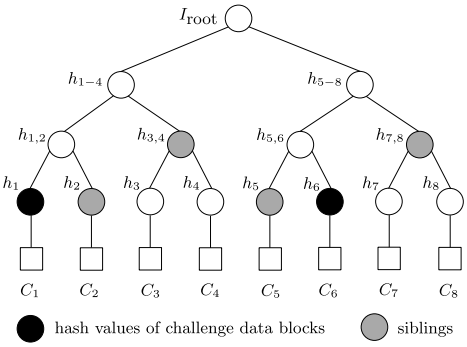


FIGURE 3. The construction of data tree.

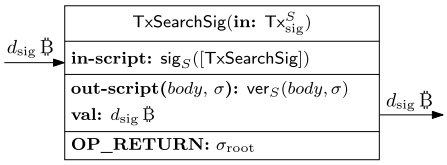


FIGURE 4. The search signature transaction TxSearchSig.

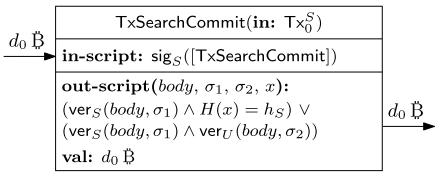


FIGURE 5. The search commitment transaction TxSearchCommit.

- **Data Encryption:** For $i \in [N]$, User computes

$$C_i = \mathsf{Encrypt}_{\text{SE}}(K_{\text{SE}}, D_i).$$

  Then $\mathcal{C} = \{C_i\}_{i \in [N]}$ will be outsourced to CSP.
- **Index Generation:** User generates $\mathcal{I}$ based on BuildIndex as shown in Algorithm 1 and sends $\mathcal{I}$ together with $\mathcal{C}$ to CSP, where $\mathcal{I}$ is an array of size $2^{1+\ell_w+\ell} \triangleq \mathsf{num}$.
- **Storage Enforcement:** CSP stores $\mathcal{I}$ and $\mathcal{C}$ after ensuring the validness based on signature verification. Furthermore, to earn search fees of User in the subsequent phases, a Merkle tree $\mathcal{T}$ is built by CSP, where the leaf nodes consist of $\mathcal{C}$. An example of $N = 8$ is shown in Figure 3, where

$$h_i = H(C_i), h_{i,j} = H(h_i \parallel h_j)$$

  and $I_{\text{root}} = H(h_{1-4} \parallel h_{5-8})$ with $h_{1-4} = H(h_{1,2} \parallel h_{3,4})$ and $h_{5-8} = H(h_{5,6} \parallel h_{7,8})$. Furthermore, CSP computes $\sigma_{\text{root}} = \mathsf{sig}_S(I_{\text{root}})$, and stores σ_{root} on the blockchain by broadcasting a search signature transaction TxSearchSig as illustrated in Figure 4. Finally, CSP sends the transaction ID to User.
- **Storage Confirmation:** Upon receiving the transaction ID from CSP, User first gets σ_{root} from the blockchain. Then, User computes I_{root} based on $\mathcal{C}$. If $\mathsf{ver}_S(I_{\text{root}}, \sigma_{\text{root}}) = \mathsf{true}$, User thinks $\mathcal{C}$ has been stored by CSP. User deletes $\mathcal{D}, \mathcal{C}$ and just stores ℓ and the transaction ID as metadata.

3) DATA SEARCH PHASE

It is introduced based on the following four procedures.

- **Search Request:** Suppose User wants to search documents with a keyword $\omega \in \mathcal{W}$, he/she generates a trapdoor T_ω based on Trapdoor as below and sends T_ω to CSP. Note that T_ω is used as a search request.
  Trapdoor$(K, \omega) \to T_\omega$: Taking the secret key K and a keyword ω as inputs, User outputs

$$T_\omega = (\pi(0, \omega, 1), \pi(0, \omega, 2), \ldots, \pi(0, \omega, N)).$$

  Besides, CSP prepares an unredeemed transaction Tx^S_0 of value d_0 Ƀ, which can be redeemed by sk_S and is used as his/her penalty in the case of invalid search. Furthermore, suppose there are unredeemed transactions Tx^U_1 of value d^U_1 Ƀ and Tx^S_1 of value d^S_1 Ƀ, which can be redeemed by User and CSP, respectively. In order to force CSP to compensate User before a specific time once the search is invalid, let $d_0 \ge d^U_1 + d^S_1$. Note that, d^U_1 Ƀ and d^S_1 Ƀ denote the search fee of User and the compensation of CSP in the case of search failure, respectively.
- **Search Commitment:** Upon receiving T_ω, CSP performs CS.Commit(CSP, User, d_0, t, r_S), where $r_S \in_R$

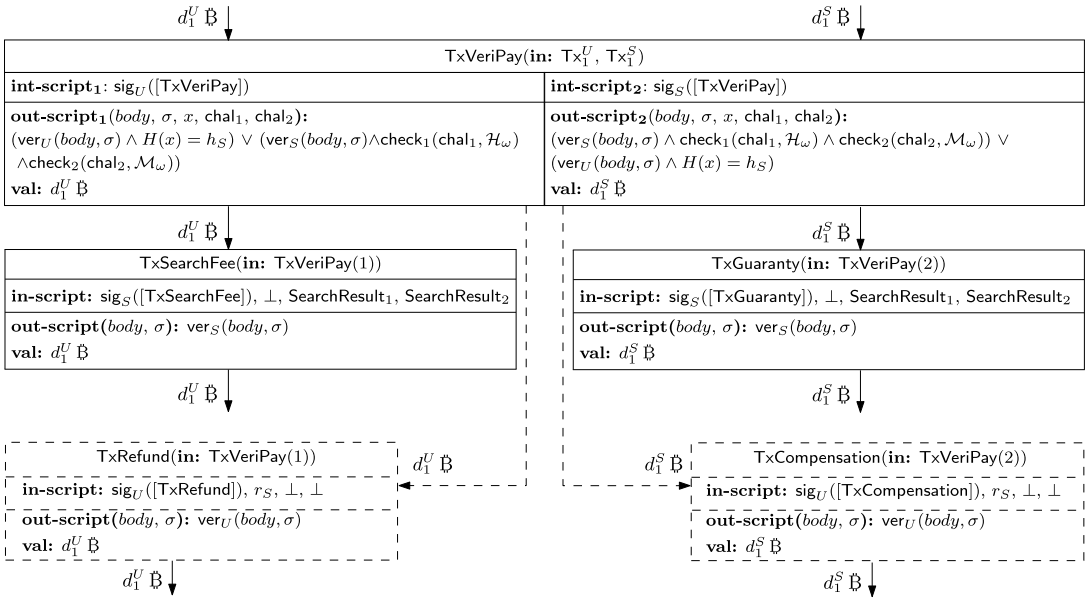


FIGURE 6. **The joint verification and payment transaction** TxVeriPay.

$\{0,1\}^*$. Specifically, CSP posts a deposit transaction TxSearchCommit of value d_0 Ƀ on the blockchain, which commits to that once the search result cannot meet the requirements specified in *Search Request* and malicious CSP refuses to compensate User in *Verification and Payment* before time t, User is able to claim enough deposits of CSP by himself/herself as a penalty in the subsequent *User Claim Phase* after time t. The details of TxSearchCommit are shown in Figure 5, where $h_S = H(r_S)$.

Once TxSearchCommit is included on the blockchain, CSP creates the body of the punishment transaction TxFine, which will be used by User to claim the penalty, signs it and sends the signed body $\mathsf{sig}_S([\mathsf{TxFine}])$ to User. TxOpen and TxFine will be detailed in *Verification and Payment* and *User Claim Phase*, respectively. Certainly, if the search is valid, CSP will eventually get his/her deposit back no matter how User behaves. In particular, CSP performs the search algorithm Search described in Algorithm 2 to obtain $\mathcal{C}_\omega$ and tag_ω. Suppose $\mathcal{C}_\omega = \{C_{s_i}\}_{i\in I_\omega}$. CSP sets $\mathcal{H}_\omega = \{H(C_{s_i})\}_{i\in I_\omega}$ and $\mathbf{path}_\omega = \mathcal{H}_\omega \cup \mathcal{AI}_\omega$, where $\mathcal{AI}_\omega$ represents the set of auxiliary node values of $\mathcal{H}_\omega$ in $\mathcal{T}$. Take the case of $N = 8$ and $\mathcal{C}_\omega = \{C_1, C_6\}$ as an example, as shown in Figure 3. We know $\mathcal{H}_\omega = \{h_1, h_6\}$ and $\mathcal{AI}_\omega = \{h_2, h_5, h_{3,4}, h_{7,8}\}$. It's noted that the root value can be easily recovered based on $\mathbf{path}_\omega$, which is sent to User by CSP.

- **Payment Commitment:** Once TxSearchCommit is included on the blockchain, and $\mathsf{sig}_S([\mathsf{TxFine}])$ and $\mathbf{path}_\omega$ are received, User makes a payment commitment that CSP is able to get corresponding search fee if the search result is valid in *Verification and Payment*. In fact, User performs the following procedures:

Algorithm 2 Search$(\mathcal{I}, \mathcal{C}, T_\omega) \to (\mathcal{C}_\omega, \mathsf{tag}_\omega)$

Input: index $\mathcal{I}$, ciphertext data set $\mathcal{C}$, trapdoor T_ω
Output: ciphertext data set $\mathcal{C}_\omega$ and tag set tag_ω with respect to ω

Let $\mathcal{C}_\omega = \emptyset$.
Parse $T_\omega = (\mathsf{addr}_{\omega,1}, \mathsf{addr}_{\omega,2}, \ldots, \mathsf{addr}_{\omega,N})$.
for $j \in [N]$ **do**
  parse $\mathcal{I}(\mathsf{addr}_{\omega,j}) = (s_j, \mathsf{tag}_{\omega,j})$
  if $s_j \neq$ *dummy* **then**
    set $\mathcal{C}_\omega = \mathcal{C}_\omega \cup C_{s_j}$
  end
end
Set $\mathsf{tag}_\omega = (\mathsf{tag}_{\omega,1}, \mathsf{tag}_{\omega,2}, \ldots, \mathsf{tag}_{\omega,N})$.
return $\mathcal{C}_\omega$ and tag_ω

  - Compute I_{root} based on $\mathbf{path}_\omega$.
  - Parse $\mathbf{path}_\omega = \mathcal{H}_\omega \cup \mathcal{AI}_\omega$ and $\mathcal{H}_\omega = \{H(C_{s_i})\}_{i\in I_\omega}$.
  - Check if $\mathsf{ver}_S(I_{\text{root}}, \sigma_{\text{root}}) = \mathsf{true}$, where σ_{root} can be obtained from the blockchain based on metadata. If it is, User proceeds. Otherwise, User quits.
  - Define $h_j^* = H(C_{s_j})$ for $j \in I_\omega$ and $h_j^* = H(\mathtt{dummy})$ for $j \in [N] - I_\omega$.
  - Set $\mathcal{M}_\omega = \{m_{\omega,j}\}_{j\in[N]}$, where $m_{\omega,j} = \mathsf{addr}_{\omega,j} \parallel h_j^*$.
  - Choose single variable x and two variable sets chal_1 and chal_2, where

$$\mathsf{chal}_1 = \{c_{1,j}\}_{j\in I_\omega}, \quad \mathsf{chal}_2 = \{c_{2,j}\}_{j\in[N]}.$$

  - Create the body of the joint deposit transaction TxVeriPay, as illustrated in Figure 7, signs it and sends the signed body $\mathsf{sig}_U([\mathsf{TxVeriPay}])$ to CSP. In Figure 7, check_1 and check_2 are defined as

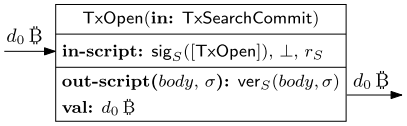


FIGURE 7. The joint verification and payment transaction TxVeriPay.

follows:

$$\mathsf{check}_1(\mathsf{chal}_1, \mathcal{H}_\omega) = \wedge_{j\in I_\omega} H(c_{1,j}) = h_j^*,$$
$$\mathsf{check}_2(\mathsf{chal}_2, \mathcal{M}_\omega) = \wedge_{j\in[N]} \mathsf{ver}_U(m_{\omega,j}, c_{2,j}).$$

Note that VeriPay specifies the search requirements of User and is finally posted on the blockchain by CSP.

- **Verification and Payment:** Upon $\mathsf{sig}_U([\mathsf{TxVeriPay}])$ is received, CSP further signs the body of TxVeriPay and puts it on the blockchain. As shown in Figure 7, the joint deposit in TxVeriPay consists of the search fee d_1^U Ƀ from User and the guaranty d_1^S Ƀ from CSP, where the guaranty is used as the compensation in *User Claim Phase*.
  In order to earn the search fee d_1^U Ƀ of User and get his/her guaranty d_1^S Ƀ back, CSP redeems TxVeriPay based on transactions TxSearchFee and TxGuaranty as described in Figure 7, respectively. CSP has to provide a valid search result consists of $\mathsf{SearchResult}_1$ and $\mathsf{SearchResult}_2$ before time $t - 2\max_B$ to prove that they meet the requirements of User. In fact, CSP sets $\mathsf{SearchResult}_1 = \mathcal{C}_\omega$ and $\mathsf{SearchResult}_2 = \mathsf{tag}_\omega$ based on Algorithm 2.
  Besides, CSP performs CS.Open(CSP, User, d_0, t, r_S), in which CSP opens the search commitment made in *Search Commitment* by posting the opening transaction TxOpen on the blockchain before time t. The detail of TxOpen is shown in Figure 7, which redeems TxSearchCommit and hence CSP can get his/her commitment deposit back. User gets search results from $\mathsf{SearchResult}_1 = \mathcal{C}_\omega$. Note that, User can ensure the validness of $\mathsf{SearchResult}_1$ based on Verify as described in Algorithm 3. Finally, CSP obtains document set $\mathcal{D}_\omega = \{D_{s_i}\}_{i\in I_\omega}$ and quits, where $D_{s_i} = \mathsf{Decrypt}_{\mathrm{SE}}(K_{\mathrm{SE}}, C_{s_i})$.

4) USER CLAIM PHASE

Suppose CSP fails to prove that the search result meets the requirements of User before time $t - 2\max_B$, TKSE comes to *User Claim Phase*. It is used by User to claim enough deposits of CSP no matter how CSP behaves. We consider the following two cases.

- **Case 1.** CSP refuses to pay the compensation of d_1^S Ƀ to User. User performs CS.Fine(CSP, User, d_0, t, r_S), in which User gets the penalty d_0 Ƀ by posting the punishment transaction TxFine on the blockchain, and then quits. The detail of TxFine is given in Figure 8.
- **Case 2.** CSP refuses to pay the penalty of d_0 Ƀ to User. User gets both the refund d_1^U Ƀ and the compensation d_1^S Ƀ by immediately putting the refund transaction TxRefund and the compensation transaction TxCompensation on the blockchain, respectively. Then User quits. TxRefund and TxCompensation are explained in Figure 7.

Algorithm 3 $\mathsf{Verify}(T_\omega, \mathcal{C}_\omega, \mathsf{tag}_\omega) \to$ `1` or `0`

Input: trapdoor T_ω, ciphertext data set $\mathcal{C}_\omega$ and tag set tag_ω related to ω
Output: `accept(1)` or `reject(0)`

```
Parse T_ω = (addr_{ω,1}, addr_{ω,2}, ..., addr_{ω,N}).
Parse C_ω = {C_{s_i}}_{i∈I_ω}.
Parse tag_ω = (tag_{ω,1}, tag_{ω,2}, ..., tag_{ω,N}).
for j ∈ [N] do
    if j ∈ I_ω then
        set h_j^* = H(C_{s_j})
    else
        set h_j^* = H(dummy)
    end
    if ver_U(addr_{ω,j} ‖ h_j^*, tag_{ω,j}) = false then
        return 0
    end
end
return 1
```

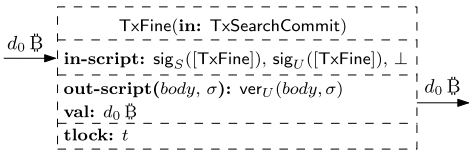


FIGURE 8. The punishment transaction TxFine.

VI. SECURITY, COMPATIBILITY AND PERFORMANCE ANALYSIS

A. SECURITY ANALYSIS AND COMPATIBILITY

TKSE is secure based on Theorem 1, Theorem 3 and Theorem 2 and it achieves compatibility based on Theorem 4.

Theorem 1 (Privacy and User-Side Verifiability): TKSE satisfies privacy and user-side verifiability if the adopted hash function H is collision-resistant, the symmetric key encryption scheme has left-or-right indistinguishability as defined by [46] and the ECDSA signature is unforgeable under chosen message attacks.

Proof: We know from Section V that an underlying V-SSE scheme is used in TKSE. In particular, V-SSE is designed based on[5] and [21]. Specifically, based on the SSE scheme [21], the scheme [5] further realizes verification of search results by designing indexes based on MAC with a secret key. In V-SSE, we generate indexes based on H and ECDSA. Therefore, if H is collision-resistant, it directly follows from the unforgeability of ECDSA signature and MAC that V-SSE has the same security as [5]. Based on the security of [5], we know that TKSE satisfies privacy and user-side verifiability as described in Theorem 1. □

TABLE 2. Feature comparison of SSE schemes.

Schemes	Privacy	User-side Verifiability	Server-side Verifiability	Payment Fairness	No TTP	Compatibility	
						Bitcoin	Ethereum
[21]	√	×	×	–	×[a]	–	–
[5]	√	√	×	–	×[a]	–	–
[38]	√	√	×	×	×[b]	×	×[b]
[37]	√	√	×[c]	×	×[d]	×	×[e]
[35]	√	√	×	×[f]	√	×[g]	√
Our TKSE	√	√	√	√	√	√	√

[a] Trusted banks are needed for fair payment.
[b] The miners are supposed to be honest to collect votes in the network, and some network nodes are supposed to be trustworthy initially.
[c] The cloud server verifies a signature which is not related to the outsourced encrypted data.
[d] Trusted network nodes are needed for a fair judgment.
[e] Some network nodes are required to send file indexes and tokens to others.
[f] The data owner may loss data without any compensation.
[g] The string concatenation is involved in the transaction output script.

Theorem 2 (Server-Side Verifiability): TKSE satisfies server-side verifiability if the adopted hash function H is collision-resistant and the ECDSA signature is unforgeable under chosen message attacks.

Proof: In the data storage phase of TKSE, after receiving ciphertext data set $\mathcal{C}$ and index $\mathcal{I}$ from User, CSP firstly performs ECDSA verification to ensure the validness of $\mathcal{C}$ and index $\mathcal{I}$. Then, the storage enforcement procedure is performed by CSP. Therefore, if H is collision-resistant, it directly follows from the public verifiability and unforgeability of the ECDSA signature that TKSE satisfies server-side verifiability. □

Theorem 3 (Fair Payment and No TTP): TKSE realizes fair payment without needing any TTP if the adopted hash function H is collision-resistant and the ECDSA signature is unforgeable under chosen message attacks.

Proof: In TKSE, pubic blockchains are introduced. Therefore, eavesdropping attacks may be made by a malicious party to undermine the fairness of the honest party. In the following, we consider two cases.

Suppose User is honest and CSP is malicious. In this case, CSP wants to obtain search fees from User without returning a search result satisfying the requirements of User before time $t - 2\max_B$. At the same time, CSP cannot pay compensation or penalty to User. If User does not get a valid search result, the joint deposit transaction TxVeriPay cannot be redeemed by CSP based on TxSearchFee and TxGuaranty before time $t - 2\max_B$ in the verification and payment phase. According to the definitions of check_1, check_2 in the payment commitment phase and $\mathsf{SearchResult}_1$, $\mathsf{SearchResult}_2$ in the verification and payment phase, we know CSP cannot obtain the search fee d_1^C ฿ from User unless CSP is able to forge an ECDSA signature or find a collision of the hash function H. To be specific, if CSP refuses to pay the compensation d_1^S ฿ to User, which means CSP does not open the search commitment by broadcasting TxOpen based on CS.Open before time t, User performs CS.Fine, in which User obtains the penalty d_0 ฿ with $d_0 \geq d_1^C + d_1^S$ by posting the punishment transaction TxFine on the blockchain. On the other hand, if CSP refuses to pay the penalty to User, which means CSP opens the search commitment by performing CS.Open to post TxOpen on the blockchain before time t, User can claim both the refund d_1^C ฿ and the compensation d_1^S ฿ by posting TxRefund and TxCompensation on the blockchain, respectively. Obviously, in any case, if CSP cannot return a valid search result, User is able to claim enough compensation besides the search fee refund or penalty from CSP no matter how CSP behaves. Similarly, payment fairness can also be realized if User is malicious. Therefore, fair payment is enabled in TKSE without needing any TTP if H is collision-resistant and ECDSA is unforgeable under chosen message attacks. □

Theorem 4 (Compatibility): TKSE is compatible with the Bitcoin blockchain and the Ethereum blockchain.

Proof: It easily follows from the design details of TKSE in Section V that the operations, which have to be performed on blockchains such as hashing and ECDSA verification, are allowed by Bitcoin and Ethereum [44], [45]. Accordingly, compatibility is realized in TKSE. □

B. PERFORMANCE ANALYSIS

In this section, we first make a feature comparison between TKSE and previous SSE schemes. Then, we evaluate the performance of TKSE in terms of the number of transactions.

1) FEATURE COMPARISON

In this section, we compare TKSE with [5], [21], [35], [37], [38] in terms of *Privacy*, *User-side Verifiability*, *Server-side Verifiability*, *Payment Fairness*, *No TTP* and *Compatibility*. We use the symbol "–" to represent that the corresponding property is not considered. As shown in Table 2, the scheme [21] cannot realize user-side verifiability, and server-side verifiability is only obtained in TKSE. In other words, TKSE is the first SSE scheme supporting two-side verifiability. Only schemes [35], [37], [38] and TKSE introduce blockchain technologies into SSE. However, only the scheme [35] and TKSE do not need any TTP, and only TKSE enables fair payment. Additionally, only TKSE is compatible with Bitcoin and Ethereum.

2) NUMBER OF TRANSACTIONS

The number of transactions in blockchain-based solutions has an important effect on performance. We further analyze and

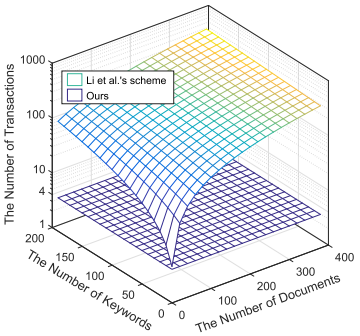


FIGURE 9. The number of transactions in a data storage and search phase.

compare the number of transactions in the blockchain-based SSE scheme [35] (Li et al.'s scheme) and TKSE, in which TTP is not needed. We consider a data storage and search phase based on the Bitcoin blockchain in the normal case. Suppose the size of each document is 40 bytes, which is equal to the output size of the opcode OP_RETURN in Bitcoin. In [35], suppose SHA-1 is used. In the data storage phase, $n_d + \frac{n_k}{2}$ transactions are created by User, where n_d is the number of documents and n_k is the number of keywords. In the data search phase, User and CSP create two and three transactions, respectively. As for TKSE, only one transaction TxSearchSig in involved in the *Data Storage Phase*. In the *Data Search Phase*, the transaction TxSearchCommit is created in the search commitment procedure. In the verification and payment procedure, transactions TxVeriPay, TxSearchFee, TxGuaranty and TxOpen are created. Note that TxOpen is created by CSP and it is not related to data search in the normal case. In the *User Claim Phase*, either the transaction TxFine or transactions TxRefund and TxCompensation are needed. Note that TxRefund and TxCompensation can be replaced with one transaction because they only need signatures of User. Similarly, TxSearchFee and TxGuaranty can be combined into one transaction. As shown in Figure 9, the number of involved transactions in TKSE is small and constant, which is not affected by the number of documents and keywords.

VII. CONCLUSION

Aiming to realize secure keyword search over encrypted data against malicious users and malicious cloud service providers, we proposed TKSE, a trustworthy keyword search scheme over encrypted cloud storage without needing any TTP. In TKSE, besides privacy and user-side verifiability, server-side verifiability is enabled for the first time in the literature, which protects honest cloud servers from being framed by malicious data owners in the data storage phase. In addition, fair payment is fulfilled based on blockchain technologies and hash functions without introducing any TTP. Our security analysis and performance evaluation indicated that TKSE is secure and efficient.

REFERENCES

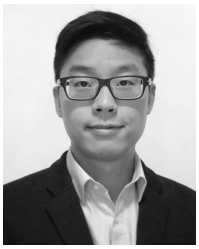
[1] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015.
[2] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Eurocrypt*, vol. 3027. Berlin, Germany: Springer, 2004, pp. 506–522.
[5] K. Kurosawa and Y. Ohtaki, "UC-secure searchable symmetric encryption," in *Financial Cryptography*, vol. 7397. Berlin, Germany: Springer, 2012, pp. 285–298.
[6] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 917–922.
[7] X. Chen, J. Li, and W. Susilo, "Efficient fair conditional payments for outsourcing computations," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 6, pp. 1687–1694, Dec. 2012.
[8] Z. Cai, H. Yan, P. Li, Z.-A. Huang, and C. Gao, "Towards secure and flexible EHR sharing in mobile health cloud under static assumptions," *Cluster Comput.*, vol. 20, no. 3, pp. 2415–2422, 2017.
[9] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.
[10] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Inf. Sci.*, vol. 379, pp. 42–61, Feb. 2017.
[11] Y. Zhang, A. Wu, and D. Zheng, "Efficient and privacy-aware attribute-based data sharing in mobile cloud computing," *J. Ambient Intell. Humanized Comput.*, pp. 1–10, May 2017. [Online]. Available: https://link.springer.com/article/10.1007%2Fs12652-017-0509-1
[12] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018.
[13] K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, and X. Shen, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 563–571, Apr. 2017.
[14] J. Shen, T. Zhou, X. Chen, J. Li, and W. Susilo, "Anonymous and traceable group data sharing in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 912–925, Apr. 2018.
[15] T. Li, J. Li, Z. Liu, P. Li, and C. Jia, "Differentially private Naive Bayes learning over multiple data sources," *Inf. Sci.*, vol. 444, pp. 89–104, May 2018.
[16] X. Zhang, Y.-A. Tan, C. Liang, Y. Li, and J. Li, "A covert channel over volte via adjusting silence periods," *IEEE Access*, vol. 6, pp. 9292–9302, Feb. 2018.
[17] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018, doi: 10.1109/JIOT.2018.2825289.
[18] Z. Liu, Y. Huang, J. Li, X. Cheng, and C. Shen, "DivORAM: Towards a practical oblivious RAM with variable block size," *Inf. Sci.*, vol. 447, pp. 1–11, Jun. 2018.
[19] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang, "Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks," *J. Netw. Comput. Appl.*, vol. 106, pp. 117–123, Mar. 2018.
[20] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.
[21] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[22] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, and D. S. Wong, "L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing," *Knowl.-Based Syst.*, vol. 79, pp. 18–26, May 2015.
[23] J. Wang *et al.*, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Comput. Sci. Inf. Syst.*, vol. 10, no. 2, pp. 667–684, 2013.
[24] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM*, May 2014, pp. 522–530.
[25] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[26] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 546–556, Sep. 2015.
[27] J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206–1216, May 2015.
[28] J. Shen, C. Wang, T. Li, X. Chen, X. Huang, and Z.-H. Zhan, "Secure data uploading scheme for a smart home system," *Inf. Sci.*, vol. 453, pp. 186–197, Jul. 2018.
[29] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014.
[30] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[31] V. Buterin, "A next-generation smart contract and decentralized application platform," White Paper, 2014. [Online]. Available: https://www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
[32] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2014, pp. 443–458.
[33] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Fair two-party computations via bitcoin deposits," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2014, pp. 105–121.
[34] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Proc. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2014, pp. 421–439.
[35] H. Li, F. Zhang, J. He, and H. Tian. (2017). "A searchable symmetric encryption scheme using blockchain." [Online]. Available: https://arxiv.org/abs/1711.01030
[36] H. G. Do and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Services (SERVICES)*, Jun. 2017, pp. 90–93.
[37] C. Cai, X. Yuan, and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.
[38] C. Cai, X. Yuan, and C. Wang, "Hardening distributed and encrypted keyword search via blockchain," in *Proc. IEEE Symp. Privacy-Aware Comput. (PAC)*, Aug. 2017, pp. 119–128.
[39] P. Jiang, F. Guo, K. Liang, J. Lai, and Q. Wen, "Searchain: Blockchain-based private keyword search in decentralized storage," *Future Gener. Comput. Syst.*, to be published, doi: 10.1016/j.future.2017.08.036.
[40] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, "Bitcoin-based fair payments for outsourcing computations of fog devices," *Future Gener. Comput. Syst.*, vol. 78, pp. 850–858, Jan. 2018.
[41] G. Ateniese, M. T. Goodrich, V. Lekakis, C. Papamanthou, E. Paraskevas, and R. Tamassia, "Accountable storage," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2017, pp. 623–644.
[42] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, Feb. 2018.
[43] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, and Y. Tang, "An ID-based linearly homomorphic signature scheme and its application in blockchain," *IEEE Access*, vol. 6, pp. 20632–20640, Feb. 2018.
[44] Bitcoin Wiki. *Script*. Accessed: Jul. 10, 2015. [Online]. Available: https://en.bitcoin.it/wiki/Script
[45] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[46] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *Proc. 38th Annu. Symp. Found. Comput. Sci.*, Oct. 1997, pp. 394–403.

YINGHUI ZHANG (M'18) received the Ph.D. degree in cryptography from Xidian University, China, in 2013. He is currently an Associate Professor with the National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications. He is also a Research Fellow with Singapore Management University. He has published over 50 research articles including ASIACCS, ACISP, Computer Networks, the IEEE Internet of Things Journal, and *Computers & Security*. His research interests include cloud security, public key cryptography, and wireless network security.

ROBERT H. DENG (F'16) has been an AXA Chair Professor of cybersecurity and a Professor of information systems with the School of Information Systems, Singapore Management University, since 2004. His research interests include data security and privacy, multimedia security, and network and system security. He served/is serving on the editorial boards of many international journals in security, including the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Dependable and Secure Computing, and the IEEE *Security and Privacy Magazine*. He has been the Chair of the Steering Committee of the ACM Asia Conference on Computer and Communications Security since 2012.

JIANGANG SHU received the B.E. and M.S. degrees from the Nanjing University of Information Science and Technology, Nanjing, China, in 2012 and in 2015, respectively. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science, City University of Hong Kong, Hong Kong. He is currently a Visiting Postgraduate Research Student with the School of Information System, Singapore Management University. His research interests include security and privacy in crowdsourcing, cloud computing security, and steganography.



KAN YANG (M'13) received the B.Eng. degree in information security from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2013. He is currently a Tenure-Track Assistant Professor with the Department of Computer Science, University of Memphis, Memphis, TN, USA. His research interests include security and privacy issues in cloud computing, big data, crowdsourcing, and Internet of Things, applied cryptography, wireless communication and networks, and distributed systems.

DONG ZHENG received the Ph.D. degree in communication engineering from Xidian University, China, in 1999. He was a Professor with the School of Information Security Engineering, Shanghai Jiao Tong University. He is currently a Professor with the National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications. He has published over 100 research articles including CT-RSA, the IEEE Transactions on Industrial Electronics. His research interests include cloud computing and public key cryptography.