This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

Checking Only When It Is Necessary: Enabling Integrity Auditing Based on the Keyword with Sensitive Information Privacy for Encrypted Cloud Data

Xiang Gao, Jia Yu, *Member, IEEE*, Yan Chang, Huaqun Wang, Jianxi Fan

***Abstract*—The public cloud data integrity auditing technique is used to check the integrity of cloud data through the Third Party Auditor (TPA). In order to make it more practical, we propose a new paradigm called integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. This paradigm is designed for one of the most common scenario, that is, the user concerns the integrity of a portion of encrypted cloud files that contain his/her interested keywords. In our proposed scheme, the TPA who is only provided with the encrypted keyword, can audit the integrity of all encrypted cloud files that contain the user's interested keyword. Meanwhile, the TPA cannot deduce the sensitive information about which files contain the keyword and how many files contain this keyword. These salient features are realized by leveraging a newly proposed Relation Authentication Label (RAL). The RAL can not only authenticate the relation that files contain the queried keyword, but also be used to generate the auditing proof without sensitive information exposure. We give concrete security analysis showing that the proposed scheme satisfies correctness, auditing soundness and sensitive information privacy. We also conduct the detailed experiments to show the efficiency of our scheme.**

***Index Terms*—Cloud storage; Sensitive information privacy; Keyword search; Data auditing; Privacy**

I. Introduction

The cloud storage service enables people to conveniently outsource their large amounts of data to centralized cloud servers. Taking Electronic Medical Record (EMR) as an example, doctors may upload patients EMRs to cloud servers, which will be accessed afterward by other doctors from different departments. The integrity of EMRs is of great importance, since the tampered EMRs may cause incorrect diagnosis, or even death to a patient. Cloud data integrity auditing techniques can check whether the users' files are intactly stored in the cloud. The integrity auditing task is usually performed by the Third Party Auditor (TPA) which has powerful computational capability that the user does not have.

Generally speaking, the TPA usually adopts the "pay-as-you-go" model to charge users according to the workload of auditing services it provides. The more cloud files are audited, the more money the user needs to pay. It is estimated by the Internet Data Center, the data held by each user will be up to 5200 GB in 2020 [1]. When such large-scale files are moved to the cloud, auditing the integrity of all cloud files periodically would bring a heavy economic burden on the user. Furthermore, it would cause unavoidable waste of resources. In most cases, the user might only concern the integrity of specific files that will be utilized shortly. For example, when a patient comes to the hospital for treatments, the doctor only concerns the integrity of the EMRs about this patient. The doctor may search and extract these EMRs from the cloud according to the identity of this patient. When medical scientists are going to do a diabetes research, they might only concern the integrity of EMRs containing the keyword "diabetes" or "GLU" in the cloud. In these scenarios, it would be more reasonable and cost-effective to only audit the integrity of files that contain the keyword of interest.

Since keywords in files often contain the user's sensitive information, the user needs to encrypt files before uploading them to the cloud. When the user wants to check the integrity of all encrypted cloud files containing the interested keyword, he/she just provides the TPA with the encrypted keyword (search trapdoor). This makes achieving integrity auditing based on the keyword for encrypted cloud data more difficult. Briefly speaking, it faces two critical challenges. The first challenge is how to audit the integrity of all encrypted cloud files containing the queried keyword under the condition that the TPA is only provided with the search trapdoor. When the TPA does not know which files contain this queried keyword, the malicious cloud might provide a valid proof which is computed from files that do not contain this keyword

This research is supported by the National Natural Science Foundation of China(62172245,61572267,61941116,61572086,62076042), the Joint Found of the National Natural Science Foundation of China(U1905211), Major Scientific and Technological Innovation project of Shandong Province(2020CXGC010114), Key Research and Development Project of Qingdao(21-1-2-21-XX), Sichuan Science and Technology Program(21SYSX0082,2017JY0168) and ChengDu Science and Technology Program(2019-YF05-02028-GX)(*Corresponding author: J. Yu*)

X. Gao is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China, also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail:gxcuit@163.com).

J. Yu is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China, also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail: qduyujia@gmail.com).

Y. Chang is with the School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China, also with the Advanced Cryptography and System Security Key Laboratory of Sichuan Province, Chengdu, China (email:cyttkl@cuit.edu.cn).

H. Wang is with Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210023, China. (email: wanghuaqun@aliyun.com)

J. Fan is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (email: jxfan@suda.edu.cn)

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

or a portion of files that contain this keyword to pass the verification. The second challenge is that, from the integrity auditing procedure, the TPA should not know which files contain this queried keyword, or the number of files containing this queried keyword. Because the task of the TPA is only to perform the integrity auditing, such sensitive information should not be exposed to the TPA. The sensitive information may disclose which encrypted keyword is more important, even exposes the inner relation among files.

In order to address above challenges, we explore how to achieve integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. The contributions of this paper can be summarized as follows:

(1) We propose a new paradigm called integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. Different from previous schemes, the TPA could check the integrity of all encrypted cloud files containing one specific keyword only with the search trapdoor in such a scheme. The proof from the cloud can pass the verification from the TPA, if and only if the cloud correctly stores all of the encrypted files that contain this keyword. In addition, the TPA cannot obtain any sensitive information, for example, which files contain the queried keyword and how many files contain the queried keyword. Existing works cannot achieve such security. Therefore, this new paradigm is different from the traditional cloud data integrity auditing. In integrity auditing procedure, our proposal is only with $O(1)$ computation complexity and communication complexity in terms of the total number N of files containing the queried keyword, which is superior to $O(N)$ complexity of the data auditing based on the verifiable searchable encryption.

(2) We propose the first integrity auditing scheme based on the keyword with sensitive information privacy for encrypted cloud data. To construct this scheme, we design a novel form of label named **Relation Authentication Label (RAL)**. This label plays an important role in realizing our design goals. On one hand, the RAL can authenticate the relation that files contain the keyword. On the other hand, it can be used to generate the auditing proof, which does not expose the identity of any file containing this keyword. As a result, the TPA can perform the integrity auditing task only with the search trapdoor. Also, the auditing proof does not expose any sensitive information to the TPA. The cloud cannot deduce the relation between the file and the queried keyword. The plaintext of the file and the queried keyword are also kept secret from the cloud.

(3) We give concrete security analysis showing that the proposed scheme satisfies correctness, auditing soundness and sensitive information privacy. We conduct comprehensive experiments to demonstrate the practicality and the efficiency of our scheme. Experimental results show that it is efficient for the user to generate/update the authenticator and the RAL. Experiment results also show that the challenge/proof generation and the proof verification do not incur heavy computation overhead in the auditing phase.

Organization: In Section II, we show the system model and design goals. In Section III, we explain the notations, preliminaries, definition and security model. In Section IV, we give the concrete scheme. In Section V and Section VI, we give the security analysis and experiment results. In Section VII, we introduce the related work. The conclusion is given in the last section.

II. System model, threat model and design goals

A. System Model

As shown in Fig. 1, the system model in the proposed scheme consists of three entities: the user, the cloud and the TPA.

The user: It is the person who wishes to store a great number of encrypted files on the cloud. He generates the secure index and authenticators, and uploads them along with the encrypted file blocks to the cloud. In order to enable the TPA to perform the auditing task on the files that contain the queried keyword, he sends the search trapdoor to the TPA.

The cloud: It is an entity who has massive storage capacity and computational power. When receiving the auditing challenge for the specific keyword, it first searches through the secure index to find the corresponding encrypted files. Then it computes the auditing proof according to the auditing challenge and sends it back to the TPA.

TPA: It is an entity who performs the auditing task on behalf of the user. It interacts with the cloud in the auditing phase, and checks the integrity of all files that contain the queried keyword.

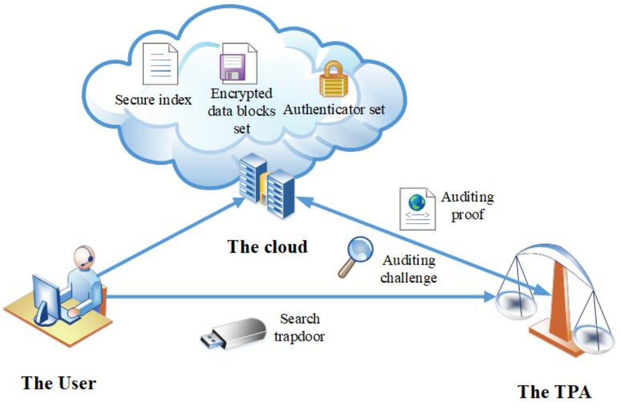


Fig. 1: System model

B. Threat Model

The cloud and the TPA can both pose potential threats.

The cloud: The data stored in the cloud may be altered or removed without the consent from the user. Making things worse, the cloud will hide the data corruption incidence. The cloud tries to fool the TPA into accepting its auditing proof when it does not possess the whole data. In addition, the cloud is curious about the plaintext of the file/queried keyword and the relation between the file and the queried keyword.

The TPA: The TPA is honest in checking the integrity of user files. Besides that, it does not actively perform the leakage-abuse attack. Therefore, we do not consider forward and backward privacy at the TPA in this paper. However, it is curious about the sensitive information of the file. It tries

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

to deduce the plaintext of the keyword and the file. It is also curious about the identities and the amount of files that contain the queried keyword.

C. Design Goals

1) *Correctness*: If the cloud correctly stores all of the files that contain the queried keyword, the auditing proof can pass the verification.
2) *Auditing Soundness*: The malicious cloud cannot forge a valid auditing proof(the aggregated authenticator and the aggregated data block) to pass the verification.
3) *Sensitive information privacy*: (1) **For the cloud**: Except the search pattern and the access pattern, the cloud cannot deduce the relation between the file and the queried keyword. The plaintext of the file and the queried keyword are also kept secret from the cloud. (2) **For the TPA**: The above-mentioned sensitive information which is hidden from the cloud is also unavailable to the TPA. In addition, from the auditing proof, the TPA cannot deduce which files contain the queried keyword and how many files contain the queried keyword. Note that we do not consider forward and backward privacy at TPA.

III. Notations, Preliminaries, Definition and Security Model

A. Notations

Some frequently used notations are shown in Table I.

B. Preliminaries

1) Bilinear map:

A map is called as bilinear map $e : G_1 \times G_1 \rightarrow G_2$ if it satisfies the following three properties:

(i) Computability: It is efficient to compute this map.
(ii) Non-degeneracy : $e(g, g) \neq 1$ for a generator $g \in G_1$.
(iii) Bilinearity: Given $a, b \in Z_q^*$ and $u, v \in G_1$, $e(u^a, v^b) = e(u, v)^{ab}$.

2) Discrete-Logarithm (DL) Assumption:

Given $g, g^a \in G_1$, where $a \in Z_q^*$, computing g is computationally infeasible.

3) Computational Diffie-Hellman (CDH) Assumption:

Given $g, g^a, u \in G_1$, where $a \in Z_q^*$, computing u^a is computationally infeasible.

4) Secure index:

As shown in Fig. 2, the secure index[2] is derived from the inverted index structure[3]. The inverted index enables the cloud to efficiently search files based on the queried keyword. The inverted index includes two parts: the keyword $w_k(1 \leq k \leq m)$ and the index vector $v_{w_k}(1 \leq k \leq m)$. The index vector, which is represented by an $n-bit$ binary string, records the relation between the file and the keyword. If the file F_i contains the keyword, the $i-th$ bit of the index vector is set to $1(\mathrm{v}_{\mathrm{w}_k}[i] \leftarrow 1)$; otherwise, the $i-th$ bit of the index vector is set to $0(\mathrm{v}_{\mathrm{w}_k}[i] \leftarrow 0)$.

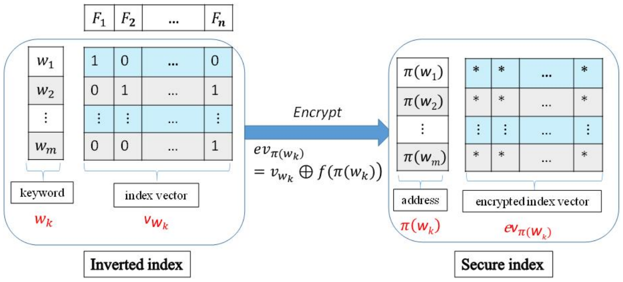


Fig. 2: Secure index

The secure index[2] enables the cloud to efficiently search encrypted files based on the encrypted keyword. The secure index includes two parts: the address $\pi(w_k)(1 \leq k \leq m)$ and the encrypted index vector $ev_{\pi(w_k)}(1 \leq k \leq m)$. The address is a permutation of the keyword. The encrypted index vector is also an $n-bit$ binary string, but hides the relation between the file and the keyword. Given the search trapdoor $\pi(w_k)$, the cloud searches the address in the secure index, and decrypts the corresponding encrypted index vector. Finally, the cloud can find the files containing this queried keyword. The secure index guarantees that : 1) without knowing the search trapdoor, the cloud cannot deduce any relation between the file and the keyword. 2) Given one search trapdoor, the cloud can only know which files contain this queried keyword. The cloud cannot deduce any other relation between other keywords and files.

C. Definition

Definition 1: An integrity auditing scheme based on the keyword with sensitive information privacy for encrypted cloud data includes eight algorithms: ($SysIni(\lambda)$, $Setup(F)$, $IndexGen(x, W, V)$, $AuthGen(x, C)$, $TrapdoorGen(w')$, $ChallGen(T_{w'})$, $ProofGen(Chal, I, C, \Phi)$, $ProofVerify(Chal, Proof)$).

1) $SysIni(\lambda) \rightarrow (pp, x, y)$: This algorithm takes the security parameter λ as input. This algorithm outputs the system parameters, the secret key x and the public key y of the user..
2) $Setup(F) \rightarrow (C, W, V)$: The user executes this algorithm. It takes the file set F as input. This algorithm outputs the encrypted data block set C, the keyword set W, the index vector set V,
3) $IndexGen(x, W, V) \rightarrow I$: The user executes this algorithm. It takes the secret key x, the keyword set W and the index vector set V as input. This algorithm outputs the secure index I.
4) $AuthGen(x, C) \rightarrow \Phi$: The user executes this algorithm. It takes the secret key x and the encrypted data block set C as input. This algorithm outputs the authenticator set Φ.
5) $TrapdoorGen(w') \rightarrow T_{w'}$: The user executes this algorithm. It takes the keyword w' as input. This algorithm outputs the search trapdoor $T_{w'}$.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

TABLE I: Notations

Notation	Meaning
$\pi(\cdot)$	A pseudo random permutation
$f(\cdot)$	A pseudo random function
G_1, G_2	Two $q-order$ multiplicative cyclic groups
e	A bilinear map $e : G_1 \times G_1 \rightarrow G_2$
g, u	Two generators in G_1
n	The total number of files
s	The total number of blocks in each file
m	The total number of keywords
F_i	The $i-th$ file
F	The file set, that is $F = \{F_1,F_2,...,F_i,...,F_n\}$
C_i	The encrypted $i-th$ file
c_{ij}	The $j-th$ block of C_i, that is $C_i = \{c_{i1}, c_{i2}, ..., c_{ij}, ..., c_{is}\}$
C	The encrypted data block set, that is $C = \{C_1, C_2, ..., C_i, ..., C_n\}$
σ_{ij}	The authenticator of encrypted data block c_{ij}
Φ	The authenticator set, that is $\Phi = \{\sigma_{11}, ..., \sigma_{1s}, ..., \sigma_{ij}, ..., \sigma_{ns}\}$
w_k	The $k-th$ keyword
W	The keyword set, that is $W = \{w_1, w_2, ..., w_k, ..., w_m\}$
$\Omega_{\pi(\mathrm{w}_k)}$	The RAL of keyword w_k
$\Omega_{\mathrm{w}_k,j}$	The $j-th$ part of the RAL $\Omega_{\pi(\mathrm{w}_k)}$, that is $\Omega_{\pi(\mathrm{w}_k)} = \{\Omega_{\mathrm{w}_k,1}, \Omega_{\mathrm{w}_k,2}, ...\Omega_{\mathrm{w}_k,j}, ..., \Omega_{\mathrm{w}_k,s}\}$
v_{w_k}	The index vector for keyword w_k
V	The index vector set, that is $V = \{v_{w_1}, v_{w_2}, ..., v_{w_k}, ..., v_{w_m}\}$
$ev_{\pi(\mathrm{w}_k)}[i]$	The $i-th$ bit in the encrypted index vector for keyword w_k
I	The secure index
$T_{w'}$	The search trapdoor of keyword w'

6) $ChallGen(T_{w'}) \rightarrow Chal$: The TPA executes this algorithm. It takes the search trapdoor $T_{w'}$ as input. This algorithm outputs the auditing challenge $Chal$.
7) $ProofGen(Chal, I, C, \Phi) \rightarrow Proof$: The cloud executes this algorithm. It takes the auditing challenge $Chal$, the secure index I, the encrypted data block set C and the authenticator set Φ as input. This algorithm outputs the auditing proof $Proof$.
8) $ProofVerify(Chal, Proof) \rightarrow \{0, 1\}$: The TPA executes this algorithm. It takes the auditing challenge $Chal$ and the auditing proof $Proof$ as input. This algorithm outputs the auditing result.

D. Security Model

The security model involves auditing soundness and sensitive information privacy. Firstly, we introduce the following game between a challenger C and an adversary A to define the auditing soundness. This game includes the following phases:

1) **Setup Phase:** The challenger C executes the $SysIni$ algorithm, and sends the system parameters pp and the public key y of the user to the adversary A.
2) **Query Phase:** The adversary A makes the following queries:
   a) Authenticator Query: The adversary A adaptively selects a series of blocks $\{c_{ij}\}(1 \leq i \leq n, 1 \leq j \leq s)$ and sends them to the challenger C. The challenger C computes the corresponding authenticators and sends them back to the adversary A.
   b) RAL Query: The adversary A adaptively selects a series of encrypted keywords $w_k(1 \leq k \leq m)$ and sends them to the challenger C. The challenger C computes the corresponding RALs and sends them back to the adversary A.
3) **Challenge Phase:** The challenger C sends the auditing challenge $Chal = \{T_w, \{j, v_j\}_{j \in Q}\}$ to the adversary A, where T_w is the search trapdoor of the queried keyword w, j is the challenged index and v_j is the challenged coefficient. The adversary is required to return an auditing proof for the challenge $Chal$.
4) **Forgery Phase:** According to the auditing challenge $Chal$, the adversary A finds files that contain the queried keyword and computes the auditing proof. This auditing proof is composed by the aggregated data block and the aggregated authenticator. If this auditing proof can pass the verification, the adversary A wins in this game.

The above-mentioned security model shows that an adversary who does not keep all challenged blocks correctly, tries to cheat the challenger to accept the auditing proof. Definition 2 shows that there exists a knowledge extractor that can extract the corresponding blocks if the adversary outputs a valid auditing proof. Definition 3 shows that, for the unchallenged blocks, if the auditing proof could pass the verification, the cloud stores them with high probability.

Definition 2(Auditing Soundness): We say that an integrity auditing scheme based on the keyword with sensitive information privacy for encrypted cloud data achieves auditing soundness if the following condition holds: if the adversary A wins in the above-mentioned game with non-negligible probability, there is a knowledge extractor which could extract the corresponding blocks except possibly with negligible probability.

Definition 3(Detectability): We say that an integrity auditing scheme based on the keyword with sensitive information

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

privacy for encrypted cloud data is (ρ, δ) detectable $(0 \leq \rho \leq 1, 0 \leq \delta \leq 1)$ if, given a fraction ρ of tampered blocks, the probability of detection for the tampered blocks is at least δ.

Next, we introduce leakage functions and experiments to define sensitive information privacy. We firstly define two leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$ [4]. These leakage functions capture what is exposed by the ciphertext and the search trapdoors. $\mathcal{L}_1(F, V)$ takes the file set F and the index vector set V as input. It reveals the total number of files $|F|$, each file block size $|F_{ij}|$, the total number of keywords $|W|$, each keyword size $|w|$ and the index i of each file. Because the RAL size is constant, this function also reveals the RAL size $|\Omega_{\pi(w_k)}|$. $\mathcal{L}_2(F, V, w')$ takes the file set F, the index vector set V and the keyword w' as input. It reveals the size of the keyword $|w'|$, the search pattern and the access pattern. Then we define the following experiments in which $\mathcal{A}$ is a stateful adversary and $\mathcal{S}$ is a stateful simulator.

$Real_A(\lambda)$: The challenger gets the secret key of the PRP and PRF. $\mathcal{A}$ sends the file set F and the index vector set V to the challenger and gets the secure index I and the encrypted data block set C. $\mathcal{A}$ makes polynomial times queries and receives the corresponding search trapdoor $T_{w'}$. Finally, $\mathcal{A}$ outputs one bit b.

$Ideal_{A,S}(\lambda)$: $\mathcal{A}$ chooses the file set F and the index vector set V. Given $\mathcal{L}_1(F, V)$, $\mathcal{S}$ sends the secure index I' and the encrypted data block set C' to $\mathcal{A}$. $\mathcal{A}$ makes polynomial times queries. Given $\mathcal{L}_2(F, V, w')$, $\mathcal{S}$ sends the search trapdoor $T'_{w'}$ to A. Finally, $\mathcal{A}$ outputs one bit b.

The above-mentioned experiments formalize a simulator and an adversary who tries to obtain sensitive information from leakage functions. Definition 4 shows that except the search pattern and the access pattern, the cloud cannot deduce any useful sensitive information such as the relation between the file and the queried keyword, the plaintext of files and the queried keyword. All above-mentioned sensitive information which is kept secret from the cloud should also be unavailable to the TPA. In addition, from the auditing proof, the TPA cannot deduce any sensitive information such as which files contain the queried keyword and how many files contain the queried keyword.

Definition 4(Sensitive Information Privacy): We say that an integrity auditing scheme based on the keyword with sensitive information privacy for encrypted cloud data achieves sensitive information privacy if the following conditions hold:

1) There exists a probabilistic polynomial-time (PPT) simulator $\mathcal{S}$ for any PPT adversary $\mathcal{A}$ such that:

$$|Pr[Real_A(\lambda) = 1] - Pr[Ideal_{A,S}(\lambda) = 1]| \leq negl(\lambda).$$

2) The sensitive information which is kept secret from the cloud is also unavailable to the TPA. In addition, the TPA cannot know which files and how many files contain the queried keyword.

IV. Our Proposed Scheme

In this section, we firstly give two straightforward approaches to achieve integrity auditing based on the keyword for encrypted cloud data. The first is the naive approach that requires the cloud to return all files containing the queried keyword to the TPA in the auditing phase. It will incur a heavy communication burden. Besides, this approach cannot realize sensitive information privacy. The second is a slightly better approach, which has better communication efficiency, but still may disclose sensitive information to the TPA. Then we give our core scheme to achieve integrity auditing based on the keyword with sensitive information privacy over encrypted cloud data.

A. A naive approach

This approach is designed partially based on the Verifiable Searchable Encryption (VSE) technique[5]. We name it as the data auditing based on the VSE. The user and the TPA share one secret key for one MAC algorithm. The cloud stores the encrypted files and the secure index along with a MAC set. In order to generate the MAC value in this MAC set, the user runs the MAC algorithm with inputting each encrypted keyword and all encrypted files that contain this keyword. When the TPA wants to check the integrity of files containing the specific keyword, it sends the encrypted keyword as the auditing challenge to the cloud. According to the secure index, the cloud finds all of the encrypted files that contain this keyword. Then he returns them along with the corresponding MAC value to the TPA. Because the TPA holds the secret key for the MAC algorithm, it can verify whether this MAC value is valid based on these received encrypted files. If it is valid, it means all files containing this keyword are intact. However, in this approach, the cloud needs to return all MACs and files containing the queried keyword to the TPA. Assume there is N files that contain the queried keyword in total. It will incur $O(N)$ communication overhead in integrity auditing procedure. In addition, the TPA has to check the validity of these MACs based on the $O(N)$ received files independently. It will incur $O(N)$ computation overhead in integrity auditing procedure. Obviously, the data auditing based on the VSE is not efficient, especially when the number or the size of files containing this keyword is large. Besides, it inevitable exposes the sensitive information, like which files contain this queried keyword, to the TPA. Therefore, this approach is unpractical.

B. A slightly better approach

Similarly to the first approach, the user and the TPA also share one secret key for one MAC algorithm in this approach. In addition, the cloud stores the encrypted files and the secure index along with a MAC set. Different from the first approach, the user sets the encrypted keyword and the corresponding file identities as the input of the MAC algorithm. When the cloud receives the encrypted keyword, it first finds the corresponding files according to the secure index. Then the cloud just returns the corresponding MAC value and file identities back to the TPA. The TPA verifies whether the MAC value is valid based on the encrypted keyword and the returned file identities. If it is valid, the TPA will know which files contain the queried keyword. By executing the regular PDP scheme[6, 7], the TPA can verify whether the cloud intactly stores all of files containing the queried keyword. Obviously, this approach does not need the cloud to send all related files to the TPA. So it

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

is much more efficient than the first approach. However, the TPA also knows which files contain this queried keyword. This sensitive information will be fully exposed to the TPA. So this approach is still practically unacceptable.

High Level Explanation: The reason why the above-mentioned approaches expose sensitive information is that, in regular PDP schemes[6, 7], for each data block c_j, the user uses his secret key x to compute an authenticator $\sigma_j = (H(ID||j) \cdot g^{c_j})^x$. For each file, the user uploads data blocks along with authenticators to the cloud. In the auditing phase, the cloud computes the aggregated authenticator and the aggregated data block as the auditing proof. Actually, the hash of the file identity is one key factor of the aggregated authenticator in the auditing proof. The TPA verifies the auditing proof with the file identity and the public key of the user. Thus, if we utilize the PDP technique directly in our setting, the TPA must know the corresponding file identities to perform the auditing task. In other words, the TPA needs to know which files contain the queried keyword. To achieve sensitive information privacy, all identities of the files containing the queried keyword need to be hidden in the auditing proof. Meanwhile, the integrity of all files that contain the queried keyword should be able to be verified. In order to realize this goal, we design a novel label called **Relation Authentication Label (RAL)**, and implant it into the secure index as shown in Fig. 3. We consider to multiple the regular aggregated authenticator[6] with the RAL to generate the final aggregated authenticator in our scheme. In order to hide the identities of files that contain the queried keyword, we set the aggregated hash inversion of these identities as one multiplication factor of the RAL. When the regular aggregated authenticator is multiplied by the RAL, all file identities could be eliminated from the auditing proof. The sensitive information privacy can be achieved by this way. However, if the RAL is only composed by this factor, the proposed scheme will not be secure against the replace attack[8, 9]. So we set the trapdoor and the corresponding block number as other two multiplication factors of the RAL. These two factors along with the aggregated hash inversion of file identities constitute the core factors in the final RAL. When this designed RAL multiples the regular aggregated authenticator, file identities could be eliminated from the auditing proof. Meanwhile, the TPA is able to verify the integrity of all files containing the queried keyword based on the trapdoor. More details will be given in the later section. Generally speaking, the designed RAL is the core component of the proposed scheme. The RAL can not only be used to authenticate the relation that files contain the queried keyword, but also is used to eliminate the sensitive information such as which files and how many files contain the queried keyword.

C. Our Core Scheme

Scheme Details: Fig. 4 shows the process of the proposed scheme. 1) The user firstly executes $Setup$, $IndexGen$ and $AuthGen$ algorithms to generate the encrypted data block set, the authenticator set, the secure index. He sends them to the cloud. 2) The user executes $TrapdoorGen$ algorithm to generate the search trapdoor, and sends it to the TPA. This search trapdoor contains the encrypted keyword, which will be used as one part of the auditing challenge. 3) The TPA executes $ChalGen$ algorithm to generate the auditing challenge, and sends it to the cloud. This auditing challenge includes the search trapdoor which enables the cloud to find the files. This auditing challenge also designates which blocks the TPA challenges. 4) The cloud executes $ProofGen$ to search through the database and finds the corresponding data blocks and authenticators. Then the cloud computes the auditing proof and sends it back to the TPA. Finally, the TPA verifies whether the auditing proof is valid.

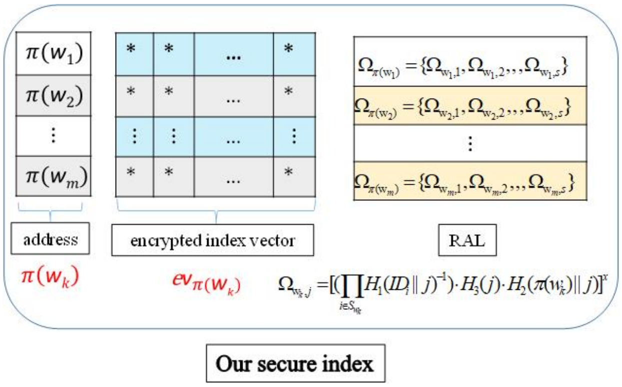

$\pi(w_1)$
$\pi(w_2)$
⋮
$\pi(w_m)$
* * ... *
⋮ ⋮ ... ⋮
$\Omega_{\pi(w_1)} = \{\Omega_{w_1,1}, \Omega_{w_1,2},,, \Omega_{w_1,s}\}$
$\Omega_{\pi(w_2)} = \{\Omega_{w_2,1}, \Omega_{w_2,2},,, \Omega_{w_2,s}\}$
⋮
$\Omega_{\pi(w_m)} = \{\Omega_{w_m,1}, \Omega_{w_m,2},,, \Omega_{w_m,s}\}$
address
encrypted index vector
RAL
$\pi(w_k)$
$ev_{\pi(w_k)}$
$\Omega_{w_k,j} = [(\prod_{i \in S_{w_k}} H_1(ID_i || j)^{-1}) \cdot H_3(j) \cdot H_2(\pi(w_k) || j)]^x$
Our secure index

Fig. 3: Our secure index with RAL

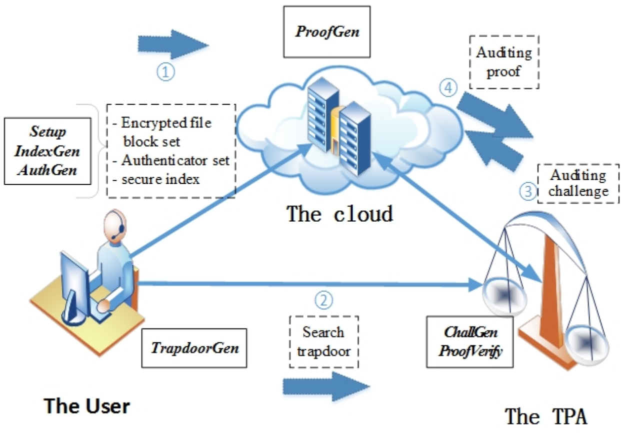


Fig. 4: the process of the proposed scheme

Now let us describe the scheme details.

1) $SysIni(\lambda) \rightarrow (pp, x, y)$
   a) Choose the system parameters pp as follows: two $q-order$ multiplicative cyclic groups G_1, G_2, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, two generators $u, g \in G_1$, three secure hash functions $H_1 : \{0, 1\}^* \rightarrow G_1, H_2 : \{0, 1\}^* \rightarrow G_1, H_3 : \{0, 1\}^* \rightarrow G_1$, a symmetric encryption algorithm $Enc(\cdot, k_0)$ with key k_0, a pseudo random permutation(PRP) $\pi_{k_1}(\cdot)$ with key k_1, and a pseudo random function(PRF) $f_{k_2}(\cdot)$ with key k_2. For simplification, we will use $\pi(\cdot)$ to denote $\pi_{k_1}(\cdot)$ and $f(\cdot)$ to denote $f_{k_2}(\cdot)$ in the detailed scheme.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

Algorithm 1 Setup

```
Input: The file set $F$ .
Output: The encrypted data block set $C$, the keyword set $W$, the index vector set $V$, the secret key $x$ and the public key $y$.
for each $F_i \in F(1 \leq i \leq n)$ do
    Split it into $s$ blocks $F_{i1}, F_{i2}, ..., F_{is}$;
    for each $1 \leq j \leq s$ do
        Compute $c_{ij} = Enc(F_{ij}, k_0)$;
    end for
end for
Set $C = \{c_{ij}\}(1 \leq i \leq n, 1 \leq j \leq s)$;
Extract all keywords, and build $W$;
for each $w_k \in W(1 \leq k \leq m)$ do;
    Create an $n - bit$ binary string $v_{w_k}$;
    Initiate all elements in $v_{w_k}$ to 0;
    for each $F_i \in F(1 \leq i \leq n)$ do
        if $F_i$ contains $w_k$ then
            Set $v_{w_k}[i] = 1$;
        end if
    end for
end for
Set $V = \{v_{w_1}, v_{w_2}, ..., v_{w_m}\}$;
Randomly choose $x \in Z_q^*$, and compute $y = g^x$;
return $(C, W, V, x, y)$;
```

Algorithm 2 IndexGen

```
Input: The secret key $x$, the keyword set $W$, the index vector set $V$.
Output: The secure index $I$.
for each $w_k \in W(1 \leq k \leq m)$ do
    Extract $v_{w_k}$ from $V$;
    Compute $\pi(w_k)$;
    Compute $ev_{\pi(w_k)} = v_{w_k} \oplus f(\pi(w_k))$;
    Create an empty set $S_{w_k} = \emptyset$;
    for each $i \in [1, n]$ do
        if $v_{w_k}[i] == 1$ then
            Add $i$ to set $S_{w_k}$;
        end if
    end for
    for each $j \in [1, s]$ do
        Compute:
            $\Omega_{\mathrm{w}_k,j} = [(\prod_{i \in S_{w_k}} H_1(ID_i||j)^{-1}) \cdot H_3(j) \cdot H_2(\pi(w_k)||j)]^x$
    end for
    Set $\Omega_{\pi(\mathrm{w}_k)} = \{\Omega_{\mathrm{w}_k,1}, \Omega_{\mathrm{w}_k,2}, ..., \Omega_{\mathrm{w}_k,s}\}$;
end for
return $I = \{\pi(w_k), ev_{\pi(\mathrm{w}_k)}, \Omega_{\pi(\mathrm{w}_k)}\}_{k=1,2,...,m}$;
```

b) Randomly choose the secret key for the user $x \in Z_q^*$, and the corresponding public key $y = g^x$.

2) $Setup(F) \to (C, W, V)$

a) For each file, the user splits it into s blocks, and encrypts them by the symmetric encryption algorithm (e.g. AES or 3DES).

b) The user extracts all keywords, then builds the keyword set W.

c) For each keyword w_k, the user creates an $n - bit$ binary string as the index vector v_{w_k}. He initiates every element in this index vector to 0. For each file F_i, if it contains keyword w_k, the user sets the $i - th$ bit of the index vector to 1: $v_{w_k}[i] = 1$.

d) All of these index vectors v_{w_k} compose the index vector set $V = \{v_{w_1}, v_{w_2}, ..., v_{w_m}\}$.

3) $IndexGen(x, W, V) \to I$

a) For each keyword w_k, the user computes $\pi(w_k)$ as the address of each row in the secure index.

b) For each keyword w_k, the user encrypts the index vector $ev_{\pi(w_k)} = v_{w_k} \oplus f(\pi(w_k))$.

c) For each keyword w_k, the user creates an empty set $S_{w_k} = \emptyset$. For each $i \in [1, n]$, if $v_{w_k}[i] = 1$, the user adds this file index i to the set S_{w_k}.

d) For each keyword w_k, the user computes the RAL $\Omega_{\pi(\mathrm{w}_k)} = \{\Omega_{\mathrm{w}_k,1}, \Omega_{\mathrm{w}_k,2}, ..., \Omega_{\mathrm{w}_k,s}\}$, where $\Omega_{\mathrm{w}_k,j} = [(\prod_{i \in S_{w_k}} H_1(ID_i||j)^{-1}) \cdot H_3(j) \cdot H_2(\pi(w_k)||j)]^x$.

e) The user sets $I = \{\pi(w_k), ev_{\pi(\mathrm{w}_k)}, \Omega_{\pi(\mathrm{w}_k)}\}_{k=1,2,...,m}$.

4) $AuthGen(x, C) \to \Phi$

a) For each encrypted data block c_{ij}, the user computes the authenticator $\sigma_{ij} = [H_1(ID_i||j) \cdot u^{c_{ij}}]^x$.

b) The user sets $\Phi = \{\sigma_{ij}\}(1 \leq i \leq n, 1 \leq j \leq s)$.

5) $TrapdoorGen(w') \to T_{w'}$

The user computes the search trapdoor as $T_{w'} = \{\pi(w'), f(\pi(w'))\}$.

6) $ChalGen(T_{w'}) \to Chal$

a) The TPA randomly chooses a c-elements subset $Q \subseteq [1, s]$.

b) For each $j \in Q$, the TPA randomly chooses $v_j \in Z_q^*$.

c) The TPA sets the auditing challenge as $Chal = \{T_{w'}, \{j, v_j\}_{j \in Q}\}$.

7) $ProofGen(Chal, I, C, \Phi) \to Proof$

a) The cloud parses the challenge $Chal = \{T_{w'}, \{j, v_j\}_{j \in Q}\}$, where $T_{w'} = \{\pi(w'), f(\pi(w'))\}$.

b) According to the address $\pi(w_k) = \pi(w')$, the cloud finds the corresponding encrypted row $ev_{\pi(w_k)}$ and the RAL $\Omega_{\pi(\mathrm{w}_k)}$ in the secure index. Then the cloud decrypts the corresponding encrypted index vector $v_{w_k} = ev_{\pi(w_k)} \oplus f(\pi(w_k))$.

c) The cloud initiates an empty set $S_{w_k} = \emptyset$. For each $i \in [1, n]$, if $v_{w_k}[i] = 1$, the cloud adds i to S_{w_k}.

d) The cloud computes $T = \prod_{i \in S_{w_k}} \prod_{j \in Q} {\sigma_{ij}}^{v_j} \cdot \prod_{j \in Q} {\Omega_{w_k,j}}^{v_j}$, $\mu = \sum_{i \in S_{w_k}} \sum_{j \in Q} c_{ij} \cdot v_j$. The cloud sets the auditing proof $Proof = \{T, \mu\}$.

8) $ProofVerify(Chal, Proof) \to \{0, 1\}$

The TPA checks the validity of the following equation:

$$e(T, g) \overset{?}{=} e((\prod_{j \in Q} (H_3(j) \cdot H_2(\pi(w')||j))^{v_j}) \cdot u^{\mu}, y) \quad (1)$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

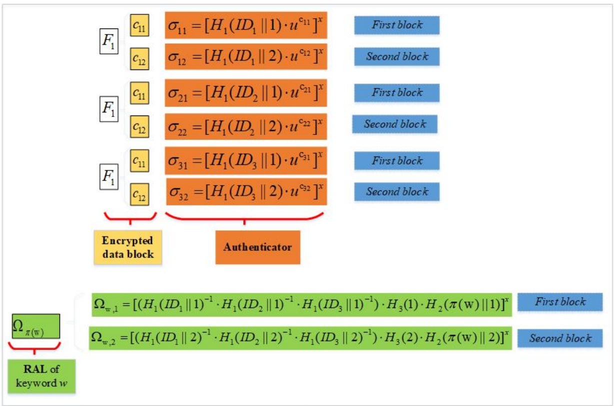
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

Algorithm 3 ProofGen

```
Input: The auditing challenge $Chal$, the secure index $I$, the
    encrypted data block set $C$, the authenticator set $\Phi$.
Output: The auditing proof $Proof$.
Extract the auditing challenge $Chal = \{T_{w'}, \{j, v_j\}_{j \in Q}\}$,
    where $T_{w'} = \{\pi(w'), f(\pi(w'))\}$;
Search the corresponding row in the secure index, where
    $\pi(w_k) = \pi(w')$;
Compute $v_{w_k} = ev_{\pi(w_k)} \oplus f(\pi(w_k))$;
Initiate an empty set: $S_{w_k} = \emptyset$;
for each $i \in [1, n]$ do
    if $v_{w_k}[i] == 1$ then
        Add $i$ to $S_{w_k}$;
    end if
end for
Compute $T = \prod_{i \in S_{w_k}} \prod_{j \in Q} {\sigma_{ij}}^{v_j} \cdot \prod_{j \in Q} {\Omega_{w_k,j}}^{v_j}$ and $\mu =$
    $\sum_{i \in S_{w_k}} \sum_{j \in Q} c_{ij} \cdot v_j$;
return $Proof = \{T, \mu\}$;
```

If this equation holds, the TPA outputs 1, which means that the files containing the queried keyword are correctly stored in the cloud; otherwise, he outputs 0 , which means that some files have been tampered.

We show an example of the RAL construction in Fig. 5. Suppose there are three files F_1, F_2, F_3 containing the keyword w. Each file is encrypted and divided into two blocks. We use $\mathrm{c}_{\mathrm{i}1}, \mathrm{c}_{\mathrm{i}2} (i = 1, 2, 3)$ to represent the first and the second encrypted data blocks of file F_i. Each encrypted data block $\mathrm{c}_{\mathrm{ij}} (i = 1, 2, 3; j = 1, 2)$ is related to an authenticator σ_{ij}. We use $\Omega_{\pi(\mathrm{w})} = \{\Omega_{\mathrm{w},1}, \Omega_{\mathrm{w},2}\}$ to represent the RAL of keyword w, where $\Omega_{\mathrm{w},1}$ and $\Omega_{\mathrm{w},2}$ correspond to the first block and the second block, respectively.



F_1
c_{11}
$\sigma_{11} = [H_1(ID_1 \| 1) \cdot u^{c_{11}}]^x$
First block
c_{12}
$\sigma_{12} = [H_1(ID_1 \| 2) \cdot u^{c_{12}}]^x$
Second block
F_1
c_{11}
$\sigma_{21} = [H_1(ID_2 \| 1) \cdot u^{c_{21}}]^x$
First block
c_{12}
$\sigma_{22} = [H_1(ID_2 \| 2) \cdot u^{c_{22}}]^x$
Second block
F_1
c_{11}
$\sigma_{31} = [H_1(ID_3 \| 1) \cdot u^{c_{31}}]^x$
First block
c_{12}
$\sigma_{32} = [H_1(ID_3 \| 2) \cdot u^{c_{32}}]^x$
Second block
Encrypted data block
Authenticator
$\Omega_{w,1} = [(H_1(ID_1 \| 1)^{-1} \cdot H_1(ID_2 \| 1)^{-1} \cdot H_1(ID_3 \| 1)^{-1}) \cdot H_3(1) \cdot H_2(\pi(\mathrm{w}) \| 1)]^x$
First block
$\Omega_{\pi(\mathrm{w})}$
$\Omega_{w,2} = [(H_1(ID_1 \| 2)^{-1} \cdot H_1(ID_2 \| 2)^{-1} \cdot H_1(ID_3 \| 2)^{-1}) \cdot H_3(2) \cdot H_2(\pi(\mathrm{w}) \| 2)]^x$
Second block
RAL of keyword w

Fig. 5: An example of the RAL construction

Note that, if the RAL is multiplied by all of authenticators of files that contain the keyword w, the file identity embedded in the authenticator could be eliminated. In this case, $\Omega_{\mathrm{w},1} \cdot \sigma_{11} \cdot \sigma_{21} \cdot \sigma_{31} = [H_2(\pi(w)||1) \cdot H_3(1) \cdot u^{\mathrm{c}_{11} + \mathrm{c}_{21} + \mathrm{c}_{31}}]^x$ (for the first block) and $\Omega_{\mathrm{w},2} \cdot \sigma_{12} \cdot \sigma_{22} \cdot \sigma_{32} = [H_2(\pi(w)||2) \cdot H_3(2) \cdot u^{\mathrm{c}_{12} + \mathrm{c}_{22} + \mathrm{c}_{32}}]^x$ (for the second block).

D. Discussion

The proposed scheme can be extended to support efficient dynamic operation. When the user modifies his file, he needs to update the secure index. The secure index includes the encrypted index vector and the RAL. It is easy to update the encrypted index vector, which is similar to other dynamic searchable encryption(SE) schemes[10–12]. The RAL generation takes most of the computation overhead of the secure index generation. However, the user does not need to re-execute the entire RAL generation when he adds/deletes/updates his files. Instead, the user only needs to update the first part $(\prod_{i \in S_{w_k}} H_1(ID_i||j)^{-1})^x$ of the corresponding RAL. Here, we give an example. Suppose three files F_1, F_2, F_3 with identities ID_1, ID_2, ID_3 contain the keyword w. The RAL of the keyword is designed as $\Omega_{\mathrm{w},j} = [(\prod_{i \in \{1,2,3\}} H_1(ID_i||j)^{-1}) \cdot H_3(j) \cdot H_2(\pi(w)||j)]^x$. If the user adds a file F_4 which also contains the keyword w, the new RAL $\Omega'_{\mathrm{w},j}$ can be easily computed based on the old RAL : $\Omega'_{\mathrm{w},j} = \Omega_{\mathrm{w},j} \cdot H_1(ID_4||j)^{-x}$. The deletion/update operation is similar to the addition operation. The computation overhead is relatively low compared with re-executing the entire RAL generation. In **Section VI**, we compare the computation overhead of the RAL update with re-executing the entire RAL generation.

When the user updates files, he also needs to compute the authenticators of the corresponding blocks and the authenticated tag. This computation is compatible with many Dynamic Provable Data Possession (DPDP) schemes, such as DPDP based on Merkle tree[13, 14], DPDP based on Oblivious RAM[15], DPDP based on authenticated dictionaries[16], DPDP based on linked table and location array[17]. Since the data dynamic is not the focus of this paper, we omit the detailed description and refer readers to the above-mentioned references for more details.

As we have analyzed previously, the computation complexity and the communication complexity of the data auditing based on the VSE are both $O(N)$ in integrity auditing procedure, where N is the total number of files containing the queried keyword. In the proposed scheme, the proof sent from the cloud to the TPA is $Proof = \{T, \mu\}$. It only incurs $O(1)$ communication overhead in terms of N. Therefore, the communication complexity of the proposed scheme is lower compared with that of the data auditing based on the VSE in integrity auditing procedure. In addition, the computation complexity for TPA to check the integrity is also only $O(1)$. In conclusion, the proposed scheme outperforms the data auditing based on the VSE in both communication complexity and computation complexity.

V. Correctness and Secure analysis

Theorem 1 (Correctness)**.** *If the cloud correctly stores all of the files that contain the queried keyword, the auditing proof can pass the verification.*

Proof. It is because:

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

$$\begin{aligned}
&e(T,g)\\
&= e(\prod_{i\in S_{w_k}}\prod_{j\in Q}{\sigma_{ij}}^{v_j}\cdot\prod_{j\in Q}{\Omega_{w_k,j}}^{v_j},g)\\
&= e(\prod_{i\in S_{w_k}}\prod_{j\in Q}[H_1(ID_i||j)\cdot u^{\mathrm{c}_{ij}}]^{x\cdot v_j}\cdot\\
&\prod_{j\in Q}[(\prod_{i\in S_{w_k}}H_1(ID_i||j)^{-1})\cdot H_3(j)\cdot H_2(\pi(w')||j)]^{x\cdot v_j},g)\\
&= e((\prod_{i\in S_{w_k}}\prod_{j\in Q}H_1(ID_i||j)^{v_j})\cdot(u^{\sum\limits_{i\in S_{w_k}}\sum\limits_{j\in Q}c_{ij}\cdot v_j})\cdot\\
&\prod_{j\in Q}\prod_{i\in S_{w_k}}H_1(ID_i||j)^{-v_j}\cdot\prod_{j\in Q}H_3(j)^{v_j}\cdot\\
&\prod_{j\in Q}H_2(\pi(w')||j)^{v_j},g^x)\\
&= e((\prod_{j\in Q}(H_3(j)\cdot H_2(\pi(w')||j))^{v_j})\cdot u^{\mu},y)
\end{aligned}$$

□

Theorem 2 (Detectability)**.** *Assume that the cloud stores total n data blocks for one file with m tampered blocks. The TPA randomly selects c blocks to check. The cloud can detect the cloud's malicious behavior with the probability of at least $1-(\frac{n-m}{n})^c$.*

Proof. Let X be the random variable denoting the number of tampered blocks that are sampled. The probability P_{Detect} is computed as follows:

$$\begin{aligned}
&P_{Detect} = P\{X\geq 1\} = 1-P\{X=0\}\\
&= 1-\tfrac{n-m}{n}\cdot\tfrac{n-m-1}{n-1}\cdot\ldots\cdot\tfrac{n-m-c+1}{n-c+1}\\
&\geq 1-(\tfrac{n-m}{n})^c
\end{aligned}$$

Remark: When only 1% files has been tampered, the probability is higher than 95% by setting $c=300$; or higher than 99% by setting $c=460$. The relation between the detectability and the number of sampling blocks can be found in Ref. [18]. □

Theorem 3 (Auditing soundness)**.** *If the CDH problem in G_1 is hard, the proposed scheme achieves auditing soundness.*

Proof. Now, we utilize a series of games to prove that the adversary cannot forge the aggregated data block and authenticator.

Game0: This game is similar to the game in the subsection *III.D*.

Game1: Game1 is as same as **Game0**, except that the challenger keeps the record of the auditing proof $P=\{T,\mu\}$. If the adversary submits a forged proof and passes the verification, but the aggregated authenticator T^* is not in the record list, this game aborts.

Analysis. Assume (T,μ) is the valid proof. From the equation(1), we know
$e(T,g)=e((\prod_{j\in Q}(H_3(j)\cdot H_2(\pi(w')||j))^{v_j})\cdot u^{\mu},y)$.

Assume the adversary outputs a forged proof (T^*,μ^*) and this proof passes the verification. From the equation (1), we know

$$e(T^*,g)=e((\prod_{j\in Q}(H_3(j)\cdot H_2(\pi(w')||j))^{v_j})\cdot u^{\mu^*},y) \quad (2)$$

Because **Game1** aborts, we have $\mu\neq\mu^*$ (Otherwise $T=T^*$). We define $\Delta\mu=\mu^*-\mu$. Next, we will show, if **Game1** aborts, the simulator can solve the CDH problem.

Given $g,g^{\partial},w\in G_1$, the simulator tries to output w^{∂}. The simulator behaves like in **Game0**, except for the following:

The simulator randomly selects $\alpha,\beta\in Z_q^*$ and sets $u=g^{\alpha}w^{\beta}$. The simulator is given g^{∂}, and does not know the secret key ∂.

For each hash query $H_1(ID_i||j)$, the simulator randomly selects $r_{ij}\in Z_q^*$ and performs random oracle as $H_1(ID_i||j)=\frac{g^{r_{ij}}}{(g^{\alpha}w^{\beta})^{c_{ij}}}$. For each hash query $H_2(\pi(w_k)||j)$, the simulator can get the corresponding file index set S_{w_k}. Then it randomly selects $r_{kj}\in Z_q^*$ and performs random oracle as $H_2(\pi(w_k)||j)=\frac{g^{r_{kj}}}{(g^{\alpha}w^{\beta})^{\sum_{i\in S_{w_k}}c_{ij}}}$. For each hash query $H_3(j)$, the simulator randomly selects $r_j\in Z_q^*$ and performs random oracles as $H_3(j)=g^{r_j}$.

Thus, it can compute the authenticator without knowing the secret key ∂:

$$\begin{aligned}
&\sigma_{ij}\\
&= [H_1(ID_i||j)\cdot u^{\mathrm{c}_{ij}}]^x\\
&= [\tfrac{g^{r_{ij}}}{(g^{\alpha}w^{\beta})^{c_{ij}}}\cdot(g^{\alpha}w^{\beta})^{c_{ij}}]^{\partial}\\
&= [g^{r_{ij}}]^{\partial}=[g^{\partial}]^{r_{ij}}
\end{aligned}$$

Similarly, the simulator can compute the RAL without knowing the secret key ∂ :

$$\begin{aligned}
&\Omega_{\mathrm{w}_k,j}\\
&= [(\prod_{i\in S_{w_k}}H_1(ID_i||j)^{-1})\cdot H_3(j)\cdot H_2(\pi(w_k)||j)]^{\partial}\\
&= [\frac{(g^{\alpha}w^{\beta})^{\sum\limits_{i\in S_{w_k}}c_{ij}}}{g^{\sum\limits_{i\in Si_{w_k}}r_{ij}}}\cdot g^{r_j}\cdot\frac{g^{r_{kj}}}{(g^{\alpha}w^{\beta})^{\sum\limits_{i\in S_{w_k}}c_{ij}}}]^{\partial}\\
&= [g^{r_{kj}+r_j-\sum\limits_{i\in S_{w_k}}r_{ij}}]^{\partial}\\
&= [g^{\partial}]^{r_{kj}+r_j-\sum\limits_{i\in S_{w_k}}r_{ij}}
\end{aligned}$$

Now, by dividing equation (2) by (1), we can get $e(\frac{T^*}{T},g)=e(u^{\Delta\mu},g^{\partial})$.

It is clear $\frac{T^*}{T}=(g^{\alpha}w^{\beta})^{\partial\cdot\Delta\mu}$. Thus, we know $w^{\partial}=[\frac{T^*}{T}\cdot(g^{\partial})^{-\alpha\cdot\Delta\mu}]^{\frac{1}{\beta\cdot\Delta\mu}}$.

Because β is randomly chosen from Z_q^* and $\Delta\mu\neq 0$, the probability of $\beta\cdot\Delta\mu\neq 0$ is $1-\frac{1}{q}$. This means that the simulator could solve the CDH problem if the difference between the adversary's probabilities of success in **Game0** and **Game1** is non-negligible.

Game2: **Game2** is as same as **Game1**, with one difference. If the adversary successfully forges a proof and passes the verification, but the aggregated data block μ^* is not in the record list, **Game2** aborts.

Analysis. Suppose the valid auditing proof is (T,μ) and the forged auditing proof is (T^*,μ^*). According to equation (1) and (2), we have:

$$e(T,g)=e((\prod_{j\in Q}(H_3(j)\cdot H_2(\pi(w')||j))^{v_j})\cdot u^{\mu},y)$$

$$e(T^*,g)=e((\prod_{j\in Q}(H_3(j)\cdot H_2(\pi(w')||j))^{v_j})\cdot u^{\mu^*},y)$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

Next, we will show if **Game2** aborts, the simulator can solve the DL problem.

Given $g, w = g^{\partial}$, the simulator tries to output ∂. The simulator behaves like in **Game1**, except for the following:

The simulator randomly selects $\alpha, \beta \in Z_q^*$ and sets $u = g^{\alpha} w^{\beta}$.

Because **Game2** aborts, we have $\mu \neq \mu^*$. We define $\Delta\mu = \mu^* - \mu$. In **Game1**, we have showed $T = T^*$. Therefore, $u^{\mu} = u^{\mu^*}$.

$$\begin{aligned} & (g^{\alpha} w^{\beta})^{\mu} = (g^{\alpha} w^{\beta})^{\mu^*} \\ \Rightarrow & (g^{\alpha} w^{\beta})^{\Delta\mu} = 1 \\ \Rightarrow & \left(g^{\alpha} \cdot g^{\partial \cdot \beta}\right)^{\Delta\mu} = 1 \\ \Rightarrow & g^{-\alpha \cdot \Delta\mu} = g^{\partial \cdot \beta \cdot \Delta\mu} \\ \Rightarrow & \partial = -\frac{\alpha}{\beta} \end{aligned}$$

As long as $\beta \neq 0$ (the probability is $1 - \frac{1}{q}$), the DL problem can be solved. This means that if the difference between the adversary's probabilities of success in **Game1** and **Game2** is non-negligible, the simulator could solve the DL problem.

The hardness of the CDH problem implies the hardness of the DL problem. Therefore, the difference between these games is negligible if the CDH problem in G_1 is hard.

Finally, we construct a knowledge extractor to complete this proof. The goal of the knowledge extractor is to extract $c_{ij} (i \in S_w, j \in Q)$. Without loss of generality, we assume that the challenger challenges $|Q|$ blocks and $|S_w|$ files contain the queried keyword. The knowledge extractor selects $|Q| \cdot |S_w|$ different coefficients and executes $|Q| \cdot |S_w|$ times challenge on the same auditing challenge. The knowledge extractor could therefore get $|Q| \cdot |S_w|$ independently linear equations. By solving these equations, the knowledge extractor could extract $c_{ij} (i \in S_w, j \in Q)$. Therefore, we have proven that the if the auditing proof passes the verification, the proposed scheme achieves auditing soundness.

□

Theorem 4 (Sensitive information privacy). *If the PRP and PRF are secure, the proposed scheme achieves sensitive information privacy.*

Proof. We firstly prove that the cloud cannot obtain the sensitive information. Then prove the TPA cannot obtain the sensitive information.

Simulating C': Given the leakage function $\mathcal{L}_1$, $\mathcal{S}$ randomly selects k_0 for the CPA-security encryption algorithm. Then he computes $c_{ij}{}' = Enc(0^{|F_{ij}|}, k_0)$, and sets $C' = \{c_{ij}{}'\} (1 \leq i \leq n, 1 \leq j \leq s)$. Due to the CPA-security encryption algorithm, C' and C are indistinguishable from each other.

Simulating I': $\mathcal{S}$ randomly selects $|w|$ bits for each $\pi'(w_k)$, n bits for each $ev'_{\pi(w_k)}$, $|\Omega_{\pi(w_k)}|$ bits for each $\Omega'_{\pi(w_k)}$ and sets $I' = \{\pi'(w_k), ev'_{\pi(\mathrm{w}_k)}, \Omega'_{\pi(\mathrm{w}_k)}\}_{k=1,2,\ldots,m}$. Due to the security of PRP and PRF, I and I' are indistinguishable from each other.

Simulating $T'_{w'}$: If this query has not been queried before, $\mathcal{S}$ randomly selects $|w'|$ bit as $\pi'(w')$, and creates an $n - bit$ binary string v. Leakage function $\mathcal{L}_2$ reveals the file index set S_{w_k}. For each $1 \leq i \leq n$, if $i \in S_{w_k}$, $\mathcal{S}$ sets $v[i] = 1$; otherwise, sets $v[i] = 0$. $\mathcal{S}$ computes $f'(\pi'(w)) = v \oplus ev'_{\pi(w)}$. Finally, $\mathcal{S}$ returns $T'_{w'} = \{\pi'(w), f'(\pi'(w))\}$ to $\mathcal{A}$. If this query has been queried before, $\mathcal{S}$ returns the corresponding $T'_{w'}$. Due to the security of PRP and PRF, $T'_{w'}$ and $T_{w'}$ are indistinguishable from each other.

Therefore, $\mathcal{A}$ cannot distinguish C' from C, I' from I, and $T'_{w'}$ from $T_{w'}$. That is, $|Pr[Real_A(\lambda) = 1] - Pr[Ideal_{A,S}(\lambda) = 1]| \leq negl(\lambda)$. We have proved that the cloud cannot deduce any sensitive information.

Since the cloud stores the secure index, the encrypted files and search trapdoor, whereas the TPA only possesses the search trapdoor, the cloud knows more information than the TPA. It is clear that the TPA also cannot deduce the relation between the file and the queried keyword, and the plaintext of files and the queried keyword.

Now, we show that, from the auditing proof, the TPA cannot know which files and how many files contain the queried keyword. From the equation (1): $e(T, g) = e((\prod_{j \in Q} (H_3(j) \cdot H_2(\pi(w')||j))^{v_j}) \cdot u^{\mu}, y)$, the TPA does not need to use file identities to check this equation. And the aggregated data block and aggregated authenticator do not expose the file identity and the number of files related to the queried keyword. Therefore, the TPA cannot know which files and how many files contain the queried keyword. □

Discussion. Leakage-abuse attack[19, 20], as one kind of powerful attacks, threatens the security of searchable encryption. Most searchable encryption schemes leak search pattern and access pattern. With above leakage information and some knowledge of the database, an adversary could build the co-occurrence matrix to recover the queried keyword. How to defend the leakage-abuse attack is not the focus of the proposed scheme. Even if this is considered, it is hard for the TPA to perform the leakage-abuse attack. In the proposed scheme, the TPA only knows the encrypted keyword and the auditing proof, i.e. the aggregated data block and the aggregated authenticator. So it is unable to get all of the required leakage information. In this case, the TPA cannot deduce the plaintext of the queried keyword. However, for the cloud, it indeed may perform the leakage-abuse attack. We could apply some selective countermeasures, such as padding technique[20] to deal with this attack.

VI. Performance evaluation

A. Numerical Analysis

We use $Hash_{G_1}$, Mul_{G_1} and Pow_{G_1} to denote one hash operation, one multiplication operation and one exponentiation operation in G_1, respectively. We use $Pair$ to denote one pairing operation. Assume there are n files and m keywords in total. Each file is divided into s blocks. For simplification, assume each keyword is related to $|S_w|$ files. We neglect the simple operations like PRP, PRF and XOR. Table II shows the computation overhead in different phases of our proposed scheme. The computation overhead for RAL generation is $m \cdot s \cdot [(|S_w| + 2) \cdot Hash_{G_1} + (|S_w| + 1) \cdot Mul_{G_1} + Pow_{G_1}]$. The computation overhead for authenticator generation is $n \cdot s \cdot (Hash_{G_1} + Mul_{G_1} + Pow_{G_1})$. The computation overhead for proof generation is $(2 \cdot c \cdot |S_w| + c) \cdot Pow_{G_1}$. And the

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

TABLE II: Computation overhead

Phase	Computation overhead
RAL generation	$m \cdot s \cdot [(\lvert S_w \rvert + 2) \cdot Hash_{G_1} + (\lvert S_w \rvert + 1) \cdot Mul_{G_1} + Pow_{G_1}]$
authenticator generation	$n \cdot s \cdot (Hash_{G_1} + Mul_{G_1} + Pow_{G_1})$
proof generation	$(2 \cdot c \cdot \lvert S_w \rvert + c) \cdot Pow_{G_1}$
proof verification	$2 \cdot Pair + 2 \cdot c \cdot Hash_{G_1} + (c+1) \cdot Mul_{G_1} + (c+1) \cdot Pow_{G_1}$

computation overhead for proof verification is $2 \cdot Pair + 2 \cdot c \cdot Hash_{G_1} + (c+1) \cdot Mul_{G_1} + (c+1) \cdot Pow_{G_1}$.

B. Experiment results

We utilize C-programming language, GMP library[21] and Pairing-Based-Cryptography (PBC) library[22] to simulate the proposed scheme. We test our experiment on Ubuntu 16.04 LTS with 1 CPU, 25GB Storage, and 1GB RAM. Since the computation overhead mainly comes from keyword searching and cloud data auditing, we test the efficiency of them, respectively. In our experiments, we utilize type-A pairing with 160 bits group order and 512 bits base field order. The length of each element in Z_q^* is 20 bytes, and the length of each element in G_1 is 128 bytes.

1) **Evaluation of authenticator generation** In the proposed scheme, each file is composed by multiple data blocks. The user needs to compute one authenticator for one data block. We test the authenticator generation time for different numbers of files, ranging from 100 to 1000, and different numbers of blocks in each file, ranging from 100 to 500. As shown in Fig. 6, when 100 files are about to be uploaded and each file contains 100 blocks, the user needs 58.9s to generate these 100*100 authenticators. When 1000 files are about to be uploaded and each file contains 500 blocks, the user needs 2865s to generate these 1000*500 authenticators. The average time to generate one authenticator is 0.0059s. We can observe that the authenticator generation time is related to the number of files and the number of blocks in each file. In reality, the user only needs to generate authenticators once, and this work can be done offline.

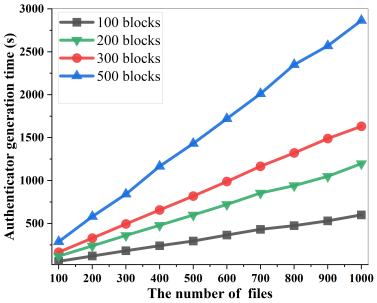


Fig. 6: The authenticator generation time

2) **Evaluation of RAL generation** For each keyword, the user needs to compute s RALs, where s is the total number of blocks in each file. We test the RAL generation time for different numbers of blocks in each file, ranging from 100 to 500, and different numbers of files, ranging from 100 to 1000. As shown in Fig. 7, when 100 files contain the keyword and each file contains 100 blocks, the user needs 27.1s to generate the RAL. When 1000 files contain the keyword and each file contains 500 blocks, the user needs 1308s to generate the RAL. When 100 files contain the keyword, the average time of generating one RAL for one block is 0.268s. When 1000 files contain the keyword, the average time of generating one RAL for one block is 0.281s. We can observe that the RAL generation time is related to the number of files containing the keyword and the number of blocks in each file. In reality, the user only needs to generate the RAL once when he computes the secure index, and this work can also be done offline.

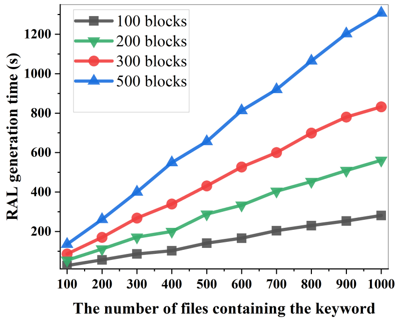


Fig. 7: The RAL generation time

3) **Evaluation of RAL update** When the user adds/deletes/updates files, our scheme achieves nice efficiency for the RAL update. The main reason is that the user does not need to re-execute the entire RAL generation. We test the RAL update time for different numbers of the affected files, ranging from 2 to 20. In our test, we assume there are total 1000 files. As shown in Fig. 8, when there are two affected files, the user needs 1301.33s if he re-executes the entire RAL generation. In contrast, the user only needs 2.51s to update the RAL in our scheme.
4) **Evaluation of proof generation** For one queried keyword, the cloud needs to use all files containing this keyword to compute the auditing proof. In our test, there are total 10000 files in the cloud. We test the proof generation time for different numbers of challenged blocks, ranging from 100 to 1000, and different numbers of files that contain the queried keyword, ranging from 100 to 1000. As shown in Fig.9, when 100 files contain

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

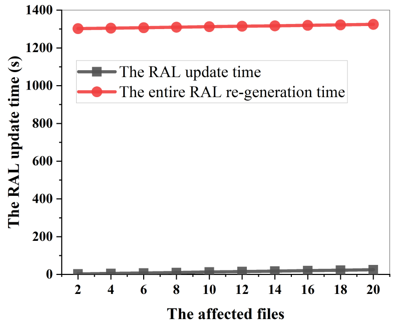


Fig. 8: The RAL update time

the queried keyword and the TPA challenges 100 blocks, the cloud needs 2.688s to generate the auditing proof. When 1000 files contain the queried keyword and the TPA challenges 1000 blocks, the cloud needs 241.33s to generate the auditing proof.

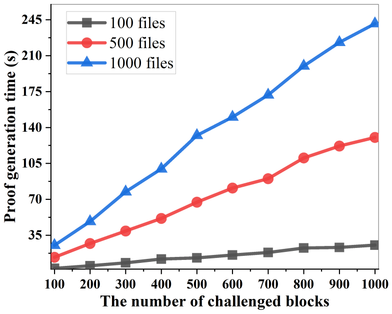


Fig. 9: The proof generation time

We also compare the search time with the auditing proof generation time. In this experiment, the TPA challenges 300 blocks. On average, the cloud needs 0.131s to finds all of files that contain the queried keyword. As shown in Fig. 10, when 0.05% of the total files contain the queried keyword, the cloud needs 76.73% of the total time to compute the auditing proof (the aggregated authenticator and the aggregated data block). When 10% of the total files contain the queried keyword, the cloud needs 99.7% of the total time to generate the auditing proof.

5) **Evaluation of challenge generation and proof verification** The TPA needs to generate the auditing challenge and verify the auditing proof. We test the challenge generation time and the proof verification time for different numbers of challenged blocks, ranging from 100 to 1000, and different numbers of queried keywords. As shown in Fig.11 and Fig. 12, when challenging 100 blocks and one keyword, the TPA only needs 0.000054s to generate the auditing challenge and 0.0085362s to verify the auditing proof. When challenging 1000 blocks and 5 keywords, the TPA needs 0.0026s to generate the auditing challenge and 0.04233s to verify the auditing proof. We can observe that the challenge generation time and the proof verification time are related to the number of challenged blocks and the number of queried keywords.

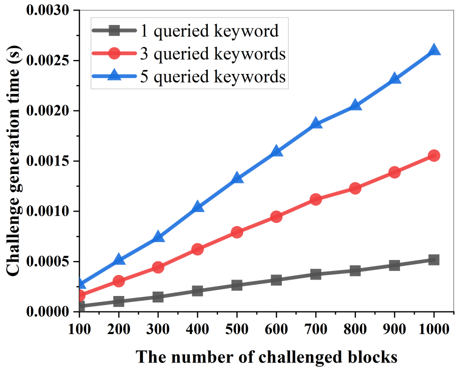


Fig. 11: The challenge generation time

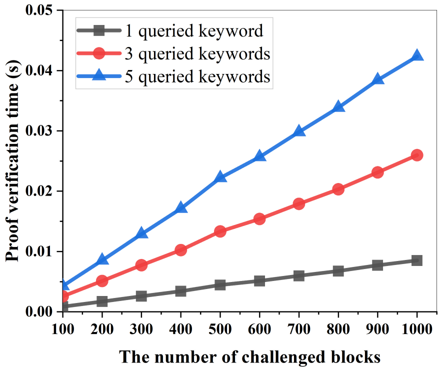


Fig. 12: The proof verification time

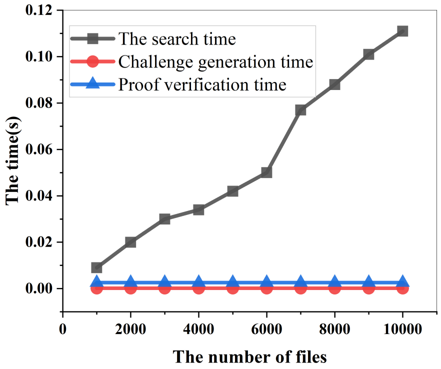
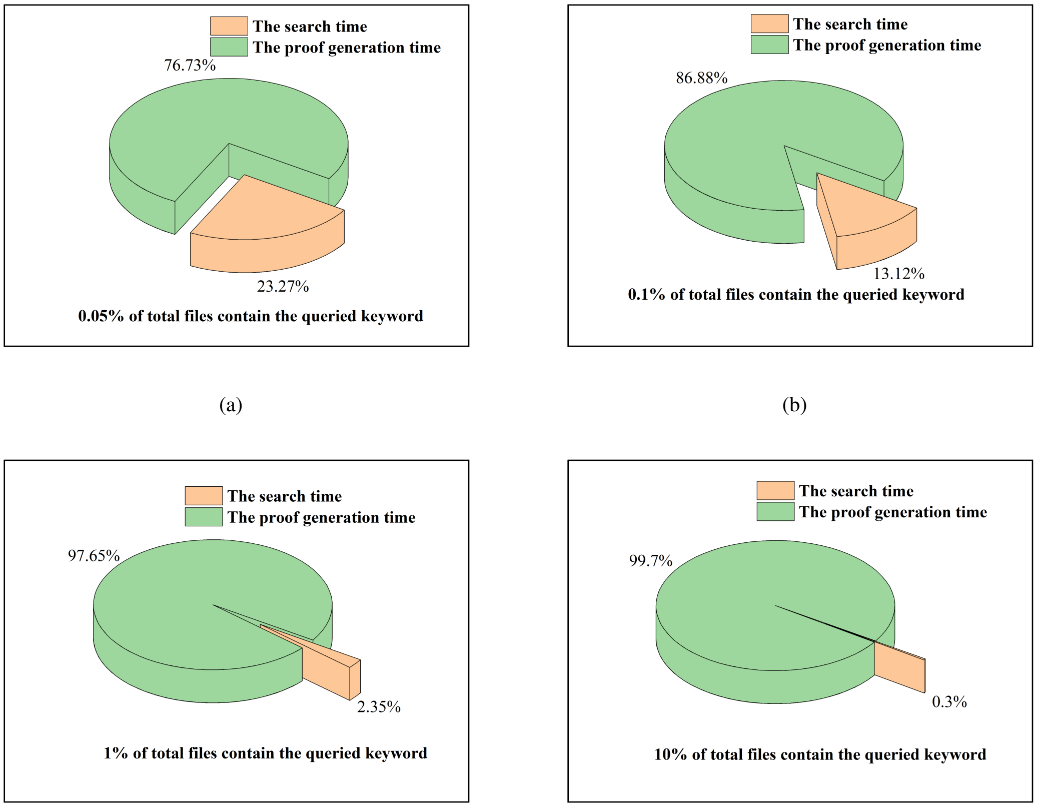


Fig. 13: The computation overhead comparison

6) **The comparison among search, challenge generation and proof verification** We evaluate the scalability and the practicality of the proposed scheme in a commonly used real-world dataset (Enron email dataset[23], https://www.cs.cmu.edu/enron/) by comparing search/challenge generation and proof verification. This dataset contains

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing



(a)

(b)

(c)

(d)

Fig. 10: The comparison between the search time and proof generation time

12,000 email files in total. To evaluate the scalability of the proposed scheme, we sample different number of files, ranging from 1000 to 10000. As shown in Fig.13, when there are 1000 files, the cloud needs 0.009s to extract files that contain the queried keyword. The TPA needs 0.000147s to generate the auditing challenge and 0.00258s to verify the auditing proof. When there are 10000 files, the cloud needs 0.111s to extract files that contain the queried keyword. The search time is linear to the number of files. The computational overhead of the TPA is independent of the number of files. Therefore, the proposed scheme is practical and can be applied to large-size data set.

7) **Evaluation of the computation and the storage overhead of the secure index** The user needs to generate the secure index for all files. As shown in Fig. 14 and Fig.15, when there are 1000 files, the user needs 1.278s to generate the secure index, and the size of the secure index is 16.64MB. When there are 10000 files, the user needs 30.918s to generate the secure index, and the size of the secure index is 466.51MB. We can observe that the secure index generation time is related to the number of files.

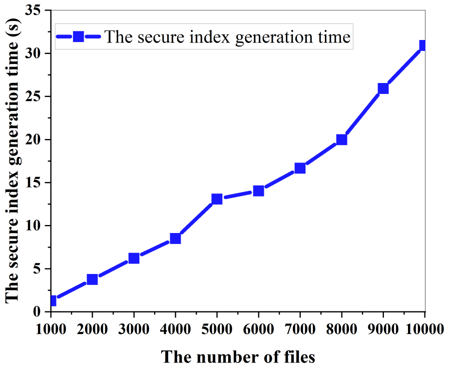


Fig. 14: The secure index generation time

VII. Related work

Cloud data auditing Provable Data Possession (PDP)[6] and Proof of Retrievability(POR)[24] are used to check the integrity of users data without retrieving all of them. The data stored in the cloud are often updated, supporting data dynamic is an essential requirement in PDP/POR. Ateniese *et al.*[25] proposed the first dynamic PDP scheme which supports data deletion and modification. Wang *et al.*[14] proposed a fully dynamic PDP scheme based on the Merkle Hash Tree.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

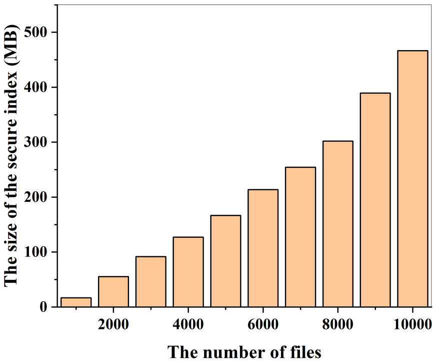


Fig. 15: The size of the secure index

This scheme supports all data dynamic operations including deletion, modification and insertion. Protecting the privacy is essential for PDP in some scenarios. The TPA should not obtain the file content or the user identity because its duty is only to check the integrity of the cloud file. To prevent the TPA from deducing the file content, Wang *et al.*[18] proposed the first privacy-preserving PDP scheme by utilizing the random-masking technique. To prevent the TPA from deducing the user identity, Wang *et al.* designed two PDP schemes[26, 27] based on ring signature and group signature. In their schemes, TPA can perform the auditing task without knowing the user identity. In the cloud data sharing scenario, users are often enrolled or revoked from a group. Wang *et al.*[28] utilized proxy re-signature technique to achieve user revocation in PDP. Once the user is revoked from a group, the authenticators generated by him should not be valid any more. Zhang *et al.*[29] realized more efficient user revocation by updating the non-revoked users signing key instead of their authenticators. Wang *et al.*[30] addressed the data transfer problem in PDP. In their scheme, the data transfer task can be securely outsourced to the public cloud. Yu *et al.*[31, 32] addressed the key-exposure problem in PDP. In their schemes, the adversary cannot forge any authenticator before or after the key-exposure period. All above-mentioned schemes are designed for checking the integrity of the specified file under the condition that the TPA provides the file name/identity. Different from the existing schemes, the TPA could check the integrity of all encrypted cloud files containing the specific keyword only with the search trapdoor in our proposed scheme. In other words, this paper initiates the first study on how to achieve integrity auditing based on the keyword for encrypted cloud files. In addition, our proposed scheme achieves sensitive information privacy. The TPA cannot deduce which files and how many files containing the queried keyword from the auditing proof in our proposed scheme.

Searchable Encryption(SE) Song *et al.*[33] proposed the first symmetric searchable encryption with two-layered encryption structure. However, the search time is linearly related to the number of files in the cloud. To improve efficiency and achieve data update in SE, Kamara *et al.*[34] proposed a dynamic SE scheme based on the inverted index. This approach can achieve sub-linear search time. Due to the spelling errors, the keyword queried by the user may not exactly match the one stored in the cloud. Li *et al.*[35] proposed the first fuzzy SE scheme over encrypted data. Their scheme utilizes edit distance to evaluate the similarity between two keywords. In order to retrieve files in a ranked order, Wang *et al.*[36] proposed the first ranked SE scheme. However, the above-mentioned schemes cannot guarantee the correctness and the completeness of search results. In other words, the malicious cloud could just return partial search results which do not contain all of the required files. Besides, the file contents can also be tampered. Verifiable Searchable Encryption(VSE)[37] has been proposed to address these issues. Zhu *et al.*[38] proposed the first generic VSE in multi-user mode. They also applied Merkle Patricia Tree to support data dynamic. In order to realize the search results verifiability and defend the keyword guessing attack[39], Miao *et al.*[40] proposed a verifiable searchable encryption scheme. They also extended it to support data dynamic, multi-keyword query and multi-key encryption. Miao *et al.*[41] proposed a VSE scheme based on enhanced vector commitment. They also extended it to support database dynamic and conjunctive keyword search. The conjunctive keyword search scheme proposed by Wang *et al.*[42] can achieve the verification of the search result even if this result is an empty set. Ge *et al.*[12] designed the Accumulative Authentication Tag(AAT) to achieve dynamic VSE. Since the AAT is based on symmetric-key cryptography, it is more efficient than RSA accumulator[11] and bilinear map accumulator[5]. Data dynamic operations for SE may lead to some secure issues, such as leakage-abuse attacks[19, 20]. In this attack scenario, the cloud could deduce whether the newly added file matches the previously searched keyword. Forward secure SSE[10] was proposed to address this security issue. Zhang *et al.*[43] firstly considered the verification for forward secure SSE, and proposed an efficient forward secure VSE scheme based on multiset hash functions. Compared with other related works, their scheme achieves superior efficiency of search and data update. Li *et al.*[44] proposed a forward and backward secure keyword search scheme with flexible keyword shielding and un-shielding. They provided the formal security proof for the proposed scheme. Wang *et al.*[45] proposed a spatial dynamic searchable encryption scheme. Besides, they gave an improved scheme to ensure the forward privacy. Li *et al.*[46] extended the notation of forward privacy, and proposed a new notation called forward search privacy. It ensures that the query on newly added files does not leak any information about past queries.

VIII. Discussion and Future Work

Our scheme enables the TPA to audit the integrity of all encrypted cloud files containing one specific keyword without exposing sensitive information. In the future, the following problems can be further explored.

Enriching the functions. The proposed scheme in this paper only supports integrity auditing based on one keyword. In order to enrich the functions, we will further explore how to enable the TPA to perform the integrity auditing based

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

on complex query conditions, such as multiple keywords query[13][41][47] and the SQL-like query[48][49][50]. Existing approaches of verifiable multiple keywords search schemes and verifiable rich query schemes might be good candidates. However, these approaches cannot be directly applied to the cloud data auditing as they expose sensitive information to the TPA. Therefore, it is interesting to design new forms of RAL for supporting complex functions in these scenarios.

Improving computation and storage efficiency. In the proposed scheme, the user needs to compute one RAL for each keyword and each data block. The cloud also needs to store all of these RALs along with the secure index. The computation overhead and storage overhead of the RAL is $O(WN)$, where W is the number of keywords and N is the number of data blocks. It is a challenge to further reduce the computation overhead and the storage overhead to make the complexity independent of W or N.

Protecting forward and backward privacy. The proposed scheme supports data dynamic updates. It is efficient for the user to update the data, the authenticator and the RAL in this scheme. Note that data dynamics may lead to some secure issues, such as the leakage-abuse attacks. We do not consider forward and backward privacy for them in this paper. It is very valuable to design new schemes to protect forward and backward privacy.

IX. Conclusion

In this paper, we address a new problem of how to achieve cloud data integrity auditing based on the keyword with sensitive information privacy. We design a new label called RAL, which is used to not only authenticate the relation that files contain the queried keyword but also generate the auditing proof without exposing any identity of file containing the queried keyword. We prove the security of the proposed scheme and evaluate the practical effectiveness by comprehensive experiments.

References

[1] "By 2020, there will be 5,200 GB of data for every person on earth," Website, https://www.computerworld.com/article/2493701/.

[2] X. Ge, J. Yu, C. Hu, H. Zhang, and R. Hao, "Enabling efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *IEEE Access*, vol. 6, pp. 45 725–45 739, 2018.

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

[4] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 258–274.

[5] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *2015 IEEE Conference on Computer Communications*. IEEE, 2015, pp. 2110–2118.

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

[7] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems and Software*, vol. 113, pp. 130–139, 2016.

[8] Y. Yu, J. Ni, M. H. Au, Y. Mu, B. Wang, and H. Li, "Comments on a public auditing mechanism for shared cloud data service," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 998–999, 2014.

[9] Y. Yu, J. Ni, M. H. Au, Y. Mu, B. Wang, and H. Li, "On the security of a public auditing mechanism for shared cloud data service," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 998–999, 2014.

[10] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable dynamic symmetric searchable encryption: Optimality and forward security." *IACR Cryptol. ePrint Arch.*, 2016.

[11] X. Zhu, Q. Liu, and G. Wang, "A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 845–851.

[12] X. Ge, J. Yu, H. Zhang, C. Hu, Z. Li, Z. Qin, and R. Hao, "Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[13] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, and J. Liu, "A position-aware merkle tree for dynamic cloud data integrity verification," *Soft Computing*, vol. 21, no. 8, pp. 2151–2164, 2017.

[14] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 355–370.

[15] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious ram," *Journal of Cryptology*, vol. 30, no. 1, pp. 22–57, 2017.

[16] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security*, vol. 17, no. 4, pp. 1–29, 2015.

[17] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402–2415, 2017.

[18] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 IEEE Conference on Computer Communications*. IEEE, 2010, pp. 1–9.

[19] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification,

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

attack and mitigation." in *Network and Distributed System Security Symposium*, vol. 20. Citeseer, 2012.

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

[21] "The gnu multiple precision arithmetic library (gmp)," Website, http://gmplib.org/.

[22] B. Lynn *et al.*, "Pbc library," *Online: http://crypto. stanford. edu/pbc*, vol. 59, pp. 76–99, 2006.

[23] "Enron email dataset," Website, https://www.cs.cmu.edu/~enron/.

[24] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2008, pp. 90–107.

[25] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*. ACM, 2008, pp. 1–10.

[26] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[27] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Applied Cryptography and Network Security*. Springer, 2012, pp. 507–525.

[28] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, 2013.

[29] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 608–619, 2020.

[30] H. Wang, D. He, A. Fu, Q. Li, and Q. Wang, "Provable data possession with outsourced data transfer," *IEEE Transactions on Services Computing*, 2019.

[31] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1931–1940, 2017.

[32] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167–1179, 2015.

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

[34] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, 2012, pp. 965–976.

[35] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *2010 IEEE Conference on Computer Communications*. IEEE, 2010, pp. 1–5.

[36] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *2010 30th International Conference on Distributed Computing Systems*. IEEE, 2010, pp. 253–262.

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

[38] J. Zhu, Q. Li, C. Wang, X. Yuan, Q. Wang, and K. Ren, "Enabling generic, verifiable, and secure data search in cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 8, pp. 1721–1735, 2018.

[39] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Workshop on secure data management*. Springer, 2006, pp. 75–83.

[40] Y. Miao, Q. Tong, R. Deng, K.-K. R. Choo, X. Liu, and H. Li, "Verifiable searchable encryption framework a-gainst insider keyword-guessing attack in cloud storage," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[41] M. Miao, J. Wang, S. Wen, and J. Ma, "Publicly verifiable database scheme with efficient keyword search," *Information Sciences*, vol. 475, pp. 18–28, 2019.

[42] J. Wang, X. Chen, S.-F. Sun, J. K. Liu, M. H. Au, and Z.-H. Zhan, "Towards efficient verifiable conjunctive keyword search for large encrypted database," in *European Symposium on Research in Computer Security*. Springer, 2018, pp. 83–100.

[43] Z. Zhang, J. Wang, Y. Wang, Y. Su, and X. Chen, "Towards efficient verifiable forward secure searchable symmetric encryption," in *European Symposium on Research in Computer Security*. Springer, 2019, pp. 304–321.

[44] Z. Li, J. Ma, Y. Miao, X. Liu, and K. K. R. Choo, "Forward and backward secure keyword search with flexible keyword shielding," *Information Sciences*, 2021.

[45] X. Wang, J. Ma, X. Liu, Y. Miao, and D. Zhu, "Spatial dynamic searchable encryption with forward security," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 746–762.

[46] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou, "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 460–474, 2021.

[47] J. Li, J. Ma, Y. Miao, Y. Ruikang, X. Liu, and K.-K. R. Choo, "Practical multi-keyword ranked search with access control over encrypted cloud data," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

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

[49] T. Xiang, X. Li, F. Chen, S. Guo, and Y. Yang, "Processing secure, verifiable and efficient SQL over outsourced database," *Information Sciences*, vol. 348, pp. 163–178,

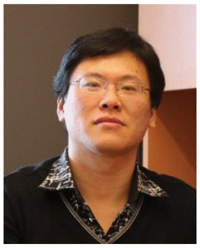
This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3106780, IEEE Transactions on Dependable and Secure Computing

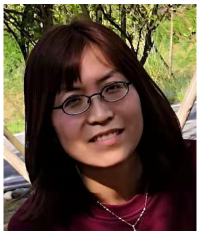
2016.

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

Xiang Gao is currently pursuing the master's degree in Qingdao University. He received his B.Eng degree in School of CyberSecurity from Chengdu University of Information and Technology in 2018. He is current interested in the security and application of the blockchain. While he also works on other security fields, including cloud security and quantum cryptography. Email: gxcuit@163.com



Jia Yu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. Since 2012, he has been a full professor and the department director of information security, at Qingdao University, China. He was a visiting professor with the Department of Computer Science and Engineering, the State University of New York at Buffalo from 2013 to 2014. His research interests include applied cryptography, cloud computing security, big data security, and network security. He has published over 150 academic papers in many international journals and conferences, such as the IEEE Transactions on Dependable and Secure Computing, the IEEE Transactions on Information Security and Forensics, and the IEEE Transactions on Service Computing. Email: qduyujia@gmail.com



Yan Chang received her Ph.D. degree in information security from University of Electronic Science and Technology of China in 2016. She is currently a professor in School of CyberSecurity, Chengdu University of Information Technology. Her research interests are quantum information, information security, big data and Cloud computing. Email:cyttkl@cuit.edu.cn

Huaqun Wang received the BS degree in mathematics education from the Shandong Normal University and the MS degree in applied mathematics from the East China Normal University, both in China, in 1997 and 2000, respectively. He received the Ph.D. degree in information security from Nanjing University of Posts and Telecommunications in 2006. He is currently a professor of Nanjing University of Posts and Telecommunications, China. His research interests include applied cryptography, network security, and cloud computing security. E-mail: wanghuaqun@aliyun.com

Jianxi Fan received his B.S., M.S.,and Ph.D. degrees in computer science from Shandong Normal University, Jinan, Shandong University, Jinan, and the City University of Hong Kong, Hong Kong, in 1988, 1991, and 2006, respectively. He is currently a professor in the School of Computer Science and Technology at Soochow University, Suzhou. He was a visiting scholar in the Department of Computer Science at Montclair State University (May 2017August 2017) and a senior research fellow in the Department of Computer Science at the City University of Hong Kong (May 2012August 2012), Hong Kong. His research interests include parallel and distributed systems, interconnection architectures, data center networks, and graph theory. Email: jxfan@suda.edu.cn

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/