Hindawi
Security and Communication Networks
Volume 2017, Article ID 6430850, 11 pages
https://doi.org/10.1155/2017/6430850

Research Article

Comparable Encryption Scheme over Encrypted Cloud Data in Internet of Everything

Qian Meng,[1,2] Jianfeng Ma,[2,3] Kefei Chen,[4] Yinbin Miao,[2,3] and Tengfei Yang[2,3]

[1]*School of Telecommunication Engineering, Xidian University, Xi'an 710071, China*
[2]*Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China*
[3]*School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[4]*School of Science, Hangzhou Normal University, Hangzhou 310036, China*

Correspondence should be addressed to Jianfeng Ma; jfma@mail.xidian.edu.cn

Received 31 August 2017; Revised 14 November 2017; Accepted 7 December 2017; Published 27 December 2017

Academic Editor: Shujun Li

User authentication has been widely deployed to prevent unauthorized access in the new era of Internet of Everything (IOE). When user passes the legal authentication, he/she can do series of operations in database. We mainly concern issues of data security and comparable queries over ciphertexts in IOE. In traditional database, a Short Comparable Encryption (SCE) scheme has been widely used by authorized users to conduct comparable queries over ciphertexts, but existing SCE schemes still incur high storage and computational overhead as well as economic burden. In this paper, we first propose a basic Short Comparable Encryption scheme based on sliding window method (SCESW), which can significantly reduce computational and storage burden as well as enhance work efficiency. Unfortunately, as the cloud service provider is a semitrusted third party, public auditing mechanism needs to be furnished to protect data integrity. To further protect data integrity and reduce management overhead, we present an enhanced SCESW scheme based on position-aware Merkle tree, namely, PT-SCESW. Security analysis proves that PT-SCESW and SCESW schemes can guarantee completeness and weak indistinguishability in standard model. Performance evaluation indicates that PT-SCESW scheme is efficient and feasible in practical applications, especially for smarter and smaller computing devices in IOE.

1. Introduction

With the new era of Internet of Everything (IOE) [1] and cloud computing [2, 3], smaller and smarter computing devices have begun to be integrated into our lives such as e-Health [4, 5], online shopping [6], and image retrieval [7]. Authentication is regarded as a first line of defense and has been widely used to prevent unauthorized access. Series of research efforts [8–17] have been made. User authentication can be password-based authentication [18, 19], biometric-based authentication [20, 21], and others [22–24]. However, security issues of user authentication, especially issues of data security and the availability of ciphertext data, are rather challenging tasks in IOE. When user passes the legal authentication, he/she can do comparable queries over ciphertexts. On the premise of ensuring safety, we concern how to make comparable queries over ciphertexts for authorized users.

As the cloud service provider is not a completely trusted entity, data usually utilize encryption technique by authorized users to guarantee security before being outsourced to the cloud service provider. There exist some scenes such as e-Health and stock exchange, which need to compare numeric data [25] over encrypted data. Unfortunately, what is of prime importance is how to make comparable operations over ciphertexts as well as data integrity without leaking any information.

To ensure comparable query operations over ciphertexts, series of research efforts [26–31] have been made. Among these efforts, one of popular works is a request-based comparable encryption scheme [31] which utilizes the idea of Prefix Preserving Encryption (PPE). Although this scheme can make comparable query operations over ciphertexts, it brings in high computational and storage burden. To this end, an efficient request-based comparable encryption

scheme was discussed by Chen et al. [32] through utilizing sliding window method to reduce computational and storage burden. To further relief ciphertexts storage space, SCE scheme was presented by Furukawa through using PPE idea [33]. Compared with request-based comparable encryption scheme, SCE scheme encrypts each bit into 3-ary, thereby dramatically reducing ciphertexts space and improving work efficiency. As the semitrusted cloud service provider may maliciously conduct a fraction of operations and forge some ciphertexts, we should verify the correctness of outsourced data for the purpose of ensuring data integrity.

To ensure data integrity without maliciously being forged, large amount of work [34–37] aimed to verify the integrity of static and dynamic outsourced data. For example, a remote integrity checking scheme which is based on modular-exponentiation cryptographic techniques was introduced by Deswarte et al. [38] Unfortunately, the new scheme has high computing complexity. To tackle this problem, Gazzoni Filho and Barreto [39] proposed a scheme by utilizing an RSA-based secure hash function in order to achieve safe data transfer transaction through a trusted third party. However, this protocol is still vulnerable to the collusion attack in a P2P environment [39] as most of existing schemes cannot prevent the user data from being leaked to external auditors. After that, Wang et al. [36] proposed a scheme known as privacy-preserving public auditing for data storage security in cloud computing, which was the first privacy-preserving auditing protocol to support scalable and public auditing in the cloud computing. In Wang et al. protocol, computational overhead came from several time-consuming operations. Aiming at reducing high computational and storage overhead, we use position-aware Merkle tree (PMT) [40] to ensure data integrity.

Inspired by the aforementioned sliding window method and PMT, we first propose a basic scheme called SCESW scheme which is based on the sliding window method to reduce computational and storage overhead. Since the cloud service provider is a semitrusted entity which can obtain some sensitive information and then derive plaintexts, we further present an enhanced scheme named PT-SCESW scheme according to PMT to verify the stored data integrity. The main contributions of our work are listed as follows.

(i) *SCESW scheme:* inspired by sliding window method and SCE scheme, we first put forward the basic SCESW scheme to relief computational burden and storage overhead as well as enhance work efficiency.

(ii) *PT-SCESW scheme:* to further protect data integrity for authorized users, we then introduce the enhanced lightweight PT-SCESW scheme based on PMT, which allows the authorized verifier to check the correctness of stored cloud data. Table 1 shows comparisons among various schemes.

(iii) *Security and efficiency:* formal security analysis demonstrates that PT-SCESW and SCESW schemes can guarantee data security and integrity as well as weak indistinguishability in standard model and experimental results using real-world dataset show its efficiency in practice.

Table 1: Comparisons among various schemes.

	SCE	SCESW	PT-SCESW
S_1	Larger	Smaller	Smaller
S_2	Larger	Smaller	Smaller
S_3	Lower	Higher	Higher
S_4	×	√	√
S_5	×	×	√

Note. S_1: storage overhead; S_2: computational overhead; S_3: efficiency; S_4: sliding window method; S_5: public auditing; Yes: √; no: ×.

The reminder of this paper is organized as follows. Section 2 depicts some preliminaries which will be used in our paper. Section 3 gives a detailed description of the proposed basic and enhanced schemes. Section 4 shows security analysis and Section 5 illustrates the performance of proposed schemes.

2. Preliminaries

In this section, we will give some descriptions of sliding window method and position-aware Merkle tree.

2.1. Sliding Window Method. Sliding window method proposed by Koç [41] is one of the widely used methods for exponentiation. For example, computing x^e, we can write e using its binary code, such as $e = (b_{n-1}, \ldots, b_1, b_0)$, $b_i \in \{0, 1\}$, $i = 0, 1, \ldots, n-1$. Based on the value of b_i, $(b_{n-1}, \ldots, b_1, b_0)$ is divided into a tuple of zero windows and nonzero windows. Sliding window technology can bring in the reduction in the amount of computation and management overhead. Algorithm 1 illustrates details of sliding window method [32].

In our schemes, numeric numbers are considered as a sequence of the binary codes. However, we suppose that all the windows have the same window size without distinguishing zero windows or nonzero windows. The fixed window size is chosen by the user's security level requirements. Hence, security and efficiency can be trade-off in practice.

2.2. Position-Aware Merkle Tree. Merkle hash tree [42] is extensively utilized in data integrity [43]. The structure of Merkle tree [44] contains a root on the top of the tree, nonleaf nodes, and leaf nodes, which is shown in Figure 1. Every nonleaf node is labeled as the hash value of its children nodes and every leaf node is defined as the hash value of a file block. $\Lambda = \{A_i \mid A_i = h(x_i), 1 \le i \le 15\}$, where $h(\cdot)$ represents a hash function. The root node of the Λ is regarded as A_{root}. For a node A_i, Auxiliary Authentication Information (AAI) is used to depict the smallest order node set $\Upsilon_i = \{A_1^i \gg A_2^i \gg \cdots\}$. Given a node A_i, AAI contains all the brother nodes related to A_i through root path from A_i to root node A_{root}. For example, the AAI of node A_3 is $\Upsilon_i = \{A_4 \gg A_9 \gg A_{14}\}$, as shown in Figure 1.

In the PMT structure, every node is noted as A_i. Besides, A_i is presented by a 3-tuple $A_i = (A_i \cdot p; A_i \cdot r; A_i \cdot v)$, where $A_i \cdot p$ represents node A_i's relative position to its parents node; $A_i \cdot r$ represents the number of node A_i's leaf nodes; $A_i \cdot v$ represents the value of the node A_i. We label nodes from left

Input: two numbers x and e where x represents base and e represents exponent
Output: $y = x^e$
(1) **for** *all* w, $w = 3, 5, 7, \ldots, 2^d - 1$ **do**
(2) Compute and store x^w;
(3) e is divided into zero windows and non-zero windows F_i of length $T(F_i)$ where $T(F_i)$ represents the length of windows;
(4) **for** $i = n - 1, \ldots, 0$ **do**
(5) Compute the value of $y := x^{F_{n-1}}$;
(6) **for** $0 \le i \le n - 2$ **do**
(7) Compute and store $y := y^{2^{T(F_i)}}$;
(8) **if** $f_i \neq 0$ **then**
(9) Compute and store $y := y \cdot x^{F_i}$;
(10) **else**
(11) F_i is a zero window.
(12) Return y;

Algorithm 1: The sliding window method [32].

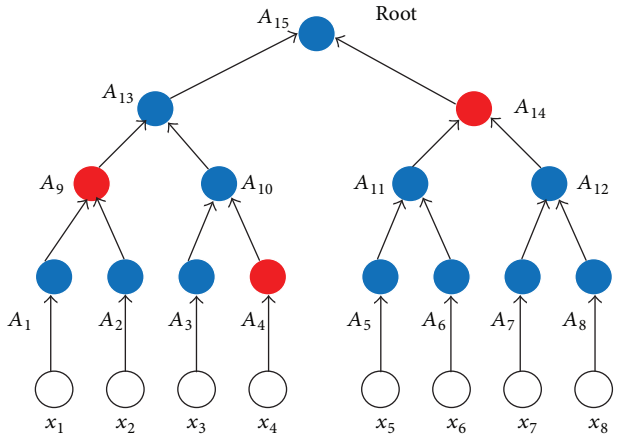

Figure 1: Position-aware Merkle tree.

to right in each layer with $A_i \cdot p$, $A_i \cdot r$, and $A_i \cdot v$ defined as follows, where set Ω_l represents the set of left subtrees, set Ω_r represents the set of right subtrees, set Ω_{root} represents the root of tree, and set Θ represents the set of leaf nodes.

$$A_i \cdot p = \begin{cases} 0 & \text{if } A_i \in \Omega_l; \\ 1 & \text{if } A_i \in \Omega_r; \\ \text{null} & \text{if } A_i \in \Omega_{\text{root}}, \end{cases}$$

$$A_i \cdot r = \begin{cases} A_l^i \cdot r + A_r^i \cdot r & \text{if } A_i \notin \Theta; \\ 1 & \text{if } A_i \in \Theta, \end{cases} \quad (1)$$

$$A_i \cdot v = \begin{cases} h\left(A_i \cdot p \parallel A_l^i \cdot v \parallel A_r^i \cdot v \parallel A_i \cdot r\right) & \text{if } A_i \notin \Theta; \\ h\left(A_i \cdot p \parallel x_i \parallel 1\right) & \text{if } A_i \in \Theta. \end{cases}$$

From Figure 1, we know that node A_3 is a leaf node that relates to the block x_3 and A_3 is located in the left of its parent node A_{10}. According to the formula above, $A_3 \cdot p = 0$, $A_3 \cdot r = 1$, and $A_3 \cdot v = h(0 \parallel x_3 \parallel 1)$. Similarly, we can obtain $A_4 = h(1 \parallel x_4 \parallel 1)$ and $A_{10} = h(1 \parallel A_3 \cdot v \parallel A_4 \cdot v \parallel 2)$. Table 2 illustrates the value of nodes in Figure 1.

Table 2: Nodes of position-aware Merkle tree in Figure 1.

Nodes	$A_i \cdot p$	$A_i \cdot r$	$A_i \cdot v$
A_1	0	1	$h(0 \parallel x_1 \parallel 1)$
A_2	1	1	$h(1 \parallel x_2 \parallel 1)$
A_3	0	1	$h(0 \parallel x_3 \parallel 1)$
A_4	1	1	$h(1 \parallel x_4 \parallel 1)$
A_5	0	1	$h(0 \parallel x_5 \parallel 1)$
A_6	1	1	$h(1 \parallel x_6 \parallel 1)$
A_7	0	1	$h(0 \parallel x_7 \parallel 1)$
A_8	1	1	$h(1 \parallel x_8 \parallel 1)$
A_9	0	2	$h(0 \parallel A_1 \cdot v \parallel A_2 \cdot v \parallel 2)$
A_{10}	1	2	$h(1 \parallel A_3 \cdot v \parallel A_4 \cdot v \parallel 2)$
A_{11}	0	2	$h(0 \parallel A_5 \cdot v \parallel A_6 \cdot v \parallel 2)$
A_{12}	1	2	$h(1 \parallel A_7 \cdot v \parallel A_8 \cdot v \parallel 2)$
A_{13}	0	4	$h(0 \parallel A_9 \cdot v \parallel A_{10} \cdot v \parallel 4)$
A_{14}	1	4	$h(1 \parallel A_{11} \cdot v \parallel A_{12} \cdot v \parallel 4)$
A_{15}	Null	8	$h(\text{null} \parallel A_{13} \cdot v \parallel A_{14} \cdot v \parallel 8)$

3. Proposed Basic and Enhanced Schemes

Before presenting concrete constructions of SCESW and PT-SCESW schemes outlined above, we give some notations which will be utilized in the whole paper, as shown in Notations.

3.1. System Model. We first describe the system model of PT-SCESW scheme which mainly involves four entities, namely, Data Owner (DO), cloud service provider (CSP), user, and Third-Party Auditor (TPA), as shown in Figure 2. When user passes the legal authentication, he/she can do comparable queries operations over encrypted data. First,

Input: Root node A_{root}, corresponding tag x_i, integrity authentication path Υ_i, k' represents numbers of corresponding nodes in Υ_i
Output: `correct`, $\perp$

(1) Let $p_0 = \neg A_1^i \cdot p$; $r_0 = 1$; $n_0 = 1$; $v_0 = h(p_0 \parallel x_i \parallel r_0)$;
(2) **for** *all* $1 \le j \le k'$ **do**
(3) **if** $j = k'$ **then**
(4) p_j = null;
(5) **else**
(6) $p_j = \neg A_{j+1}^i \cdot p$.
(7) $r_j = \neg A_{j+1}^i \cdot r + r_{j-1}$;
(8) **if** $\neg A_j^i \cdot p = 1$ **then**
(9) $n_j = n_{j-1}$; $v_j = h(p_j \parallel h_{j-1} \parallel A_j^i \cdot v \parallel r_j)$;
(10) **else**
(11) **if** $\neg A_j^i \cdot p = 0$ **then**
(12) $n_j = n_{j-1} + A_j^i \cdot r$; $v_j = h(p_j \parallel A_j^i \cdot v \parallel h_{j-1} \parallel r_j)$;
(13) **if** $r_{k'} = A_{root} \cdot r$; $v_{k'} = A_{root} \cdot v$ and $n_{k'} = i$ **then**
(14) Output `correct`, $n_{k'}$, $A = \{\text{null}, r_{k'}, v_{k'}\}$;
(15) **else**
(16) Output $\perp$.

Algorithm 2: Integrity verification algorithm.

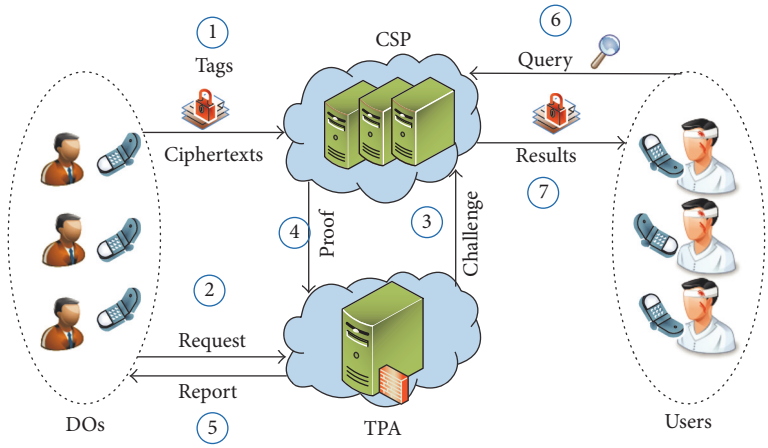

Figure 2: The architecture of PT-SCESW scheme.

the DO encrypts files by using SCESW scheme and finally sends the file $\mathscr{CP}$ and the corresponding φ to the CSP. When user wants to issue the search query over encrypted cloud data, he/she needs to submit a search query to CSP. The CSP returns the result of the query to the user. If the verifier wants to check the outsourced data integrity, she/he sends an auditing request to the TPA and the TPA submits the auditing challenge $\mathscr{CL}$ to the CSP. Upon receiving the auditing challenge $\mathscr{CL}$, CSP computes $\mathscr{P}_1$, $\mathscr{P}_2$ and sends the auditing proof to the TPA. Then TPA conducts the integrity verification algorithm (Algorithm 2) to check the data integrity and returns the auditing report to the verifier. Figure 2 depicts the task of each entity, with an assumption that the DO is the verifier.

(1) DO: it has twofold responsibilities. Firstly, data files are encrypted through SCESW scheme and then outsourced to the CSP, as shown in step ①. Secondly, the DO sends auditing request to the TPA in order to check ciphertexts integrity, as illustrated in step ②.

(2) CSP: it can provide infinite storage and computation resources to the DO and the user. After executing auditing challenge, the CSP sends auditing proof to the TPA, as shown in step ④.

(3) User: it has the following responsibilities. Firstly, the user submits a query to compare a pair of ciphertexts $\mathscr{CP}$ and $\mathscr{CP}^*$, as shown in step ⑥. Secondly, upon doing `Cmp` operation, the CSP returns the relationship of two numeric ciphertexts, as illustrated in step ⑦.

(4) TPA: it has twofold responsibilities. Firstly, the TPA submits auditing challenge to the CSP, as shown in step ③. Secondly, the TPA returns the auditing report to the verifier shown in step ⑤. If the result of auditing is correctness, system continues the `Cmp` step;

SCESW scheme is a tuple of algorithms including `KeyGen, Par, Der, Enc, Cmp`, which are shown as follows:

(i) *KeyGen*(1^k): given the security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$ and master key $\mathcal{MSK}$, the DO runs the algorithm to output the master key $\mathcal{MSK}$ and public parameter $\mathcal{PP}$.

(ii) *Par*$(\mathbb{N})$: given the number $\mathbb{N}$, the DO runs the algorithm to output the number $\mathbb{N}'$ rewritten through its binary code by utilizing sliding window method, where t represents the window size, $m = n/t$ is the number of blocks and $\mathbb{N} = \mathbb{N}'$

$$\mathbb{N} = (b_0, \ldots, b_{n-1}) = \sum_{0 \le i \le n-1} b_i 2^i;$$
$$\mathbb{N}' = (B_0, \ldots, B_{m-1}) = \sum_{0 \le i \le m-1} B_i (2^t)^i.$$

(iii) *Der*$(k, n, \mathcal{MSK}, \mathbb{N})$: given the security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$, master key $\mathcal{MSK}$ and num $0 \le \mathbb{N} \le 2^n$, the DO runs the algorithm to output a token $\mathcal{TK}$.

(iv) *Enc*$(k, n, \mathcal{MSK}, \mathbb{N})$: given the security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$, master key $\mathcal{MSK}$ and num $0 \le \mathbb{N} \le 2^n$, the DO runs the algorithm to generate the ciphertext $\mathcal{CP}$ and submits it to the CSP.

(v) *Cmp*$(\mathcal{PP}, \mathcal{CP}, \mathcal{CP}^*, \mathcal{TK})$: given the public parameter $\mathcal{PP}$, two ciphertexts $\mathcal{CP}$ and $\mathcal{CP}^*$, and token $\mathcal{TK}$, the CSP outputs $-1, 0, 1$, then it returns the relevant search results to the user.

Algorithm 3: Definition of SCESW scheme.

otherwise, the scheme demonstrates that ciphertexts are not with integrity and system stops working.

3.2. The SCESW Scheme. Let t be the window size, which means each block file has t bits. We assume arbitrary number n is a multiple of t. If n is not a multiple of t, we make n a multiple of t by adding zero in the end of the n's binary code. SCESW scheme consists of five algorithms, namely, `KeyGen, Par, Der, Enc, and Cmp`. When user passes the legal authentication, he/she can do comparable queries operations over encrypted data. Thus, we mainly consider data security and comparable queries operations over ciphertexts. A detailed construction of SCESW scheme is depicted as follows.

(1) Definitions of SCESW Scheme. The SCESW scheme is composed of five algorithms involving `KeyGen, Par, Der, Enc, and Cmp`. SCESW system definition can be defined in Algorithm 3.

(2) Details of SCESW Scheme. Concrete construction of SCESW scheme can be defined as follows.

(i) *KeyGen*$(k, n) \rightarrow (\mathcal{PP}, \mathcal{MSK})$: given a security parameter $k \in \mathcal{N}$ and range parameter $n \in \mathcal{N}$, `KeyGen` first randomly chooses hash functions: $H_1(\cdot), H_2(\cdot), H_3(\cdot) : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$ and then returns $\mathcal{PP} = (n, H_1, H_2, H_3)$.

Finally, `KeyGen` algorithm outputs a public parameter $\mathcal{PP}$ and a master key $\mathcal{MSK}$.

(ii) *Par*$(\mathbb{N}, t) \rightarrow (\mathbb{N}')$: given an original number $\mathbb{N}$, the DO rewrites it through its binary code by utilizing sliding window method with (2), where t represents the window size, $m = n/t$ is the number of blocks, and $\mathbb{N} = \mathbb{N}'$:

$$\begin{aligned} \mathbb{N} &= (b_0, \ldots, b_{n-1}) = \sum_{0 \le i \le n-1} b_i 2^i; \\ \mathbb{N}' &= (B_0, \ldots, B_{m-1}) = \sum_{0 \le i \le m-1} B_i \left(2^t\right)^i. \end{aligned} \tag{2}$$

(iii) *Der*$(\mathcal{PP}, \mathcal{MSK}, \mathbb{N}) \rightarrow (\mathcal{TK})$: given $\mathcal{PP} = (n, H_1, H_2, H_3)$, master key $\mathcal{MSK}$, and number $\mathbb{N}$, `Der` algorithm outputs a token $\mathcal{TK} = (d_1, d_2, \ldots, d_m)$ with the following equations:

$$\begin{aligned} \mathbb{N} &= (B_0, \ldots, B_{m-1}) = \sum_{0 \le i \le m-1} B_i \left(2^t\right)^i; \\ B_0 &= (b_0, b_1, \ldots, b_{t-1}), \ldots, \\ B_{m-1} &= (b_{n-t}, b_{n-t+1}, \ldots, b_{n-1}), \\ B_m &= 0; \\ d_i &= H_1 (\mathcal{MSK}, B_m, \ldots, B_i), \quad i = 1, \ldots, m. \end{aligned} \tag{3}$$

(iv) *Enc*$(\mathcal{PP}, \mathcal{MSK}, \mathbb{N}) \rightarrow (\mathcal{CP})$: given $\mathcal{PP} = (n, H_1, H_2, H_3)$, master key $\mathcal{MSK}$, and number $\mathbb{N}$, `Enc` randomly picks $\mathcal{TK} = (d_1, d_2, \ldots, d_m)$ and a random number $I \in \{0,1\}^k$, where

$$\mathbb{N} = (B_0, \ldots, B_{m-1}) = \sum_{0 \le i \le m-1} B_i \left(2^t\right)^i. \tag{4}$$

Next, `Enc` generates f_i, where

$$\begin{aligned} f_i = {} & H_3 (d_{i+1}, I) + H_2 (\mathcal{MSK}, d_{i+1}) \\ & + B_i \bmod \left(2^{(t+1)} - 1\right) \quad (i = m-1, \ldots, 0). \end{aligned} \tag{5}$$

PT-SCESW scheme is a series of algorithms namely *Setup, Encryption, Auditing, Comparison* phases, which are shown as follows:

Setup Phase. The DO chooses a security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$ and master key $\mathcal{MSK}$ to generate a public parameter $\mathcal{PP}$. The DO runs `KeyGen` to produce the secret key $\mathcal{SK}$ and public key $\mathcal{PK}$. `Setup` phase outputs the secret key $\mathcal{SK}$, public key $\mathcal{PK}$, public parameter $\mathcal{PP}$ and master key $\mathcal{MSK}$. `Setup` phase contains `KeyGen` algorithm in SCESW scheme. The DO shares $\mathcal{PK}$ with others and preserves $\mathcal{SK}$ as a secret.

Encryption Phase

(i) $Par(\mathbb{N})$, $Der(\mathcal{PP}, \mathcal{MSK}, \mathbb{N})$: system definitions are similar to SCESW scheme, as shown in Algorithm 3.

(ii) $Enc(\mathcal{PP}, \mathcal{MSK}, \mathbb{N})$: given a security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$, master key $\mathcal{MSK}$, public key $\mathcal{PK}$, private key $\mathcal{SK}$ and num $0 \le \mathbb{N} \le 2^n$, the DO runs the algorithm to output a ciphertext $\mathcal{CP}$, set φ and the metadata. Then file $\mathcal{CP}$ and set φ will be sent by the DO to the CSP. The metadata might be signed and kept by the DO.

Auditing Phase

(i) $ChalGen(s) \rightarrow (\mathcal{CL})$: given the secret parameter s, the verifier outputs auditing challenge $\mathcal{CL}$ for the query.

(ii) $ProofGen(g_s, F, \varphi, \mathcal{CL}) \rightarrow (\mathcal{P})$: given the DO's public parameter g_s, file F, set φ and auditing challenge $\mathcal{CL}$, the TPA outputs the auditing proof $\mathcal{P}$ to verify that the CSP owns the outsourced file correctly.

(iii) $ProofCheck(\mathcal{PK}, \mathcal{CL}, metadata, \mathcal{P}) \rightarrow (correct, \perp)$: given the DO's public key $\mathcal{PK}$, evidence $\mathcal{P}$, metadata and auditing challenge $\mathcal{CL}$, the TPA outputs `correct` or $\perp$. If the proof $\mathcal{P}$ passes the verification, the function outputs `correct`; otherwise, the function outputs $\perp$ and the system stops to work. At last, TPA sends the auditing report to the verifier.

Comparison Phase

(i) $Cmp(\mathcal{CP}, \mathcal{CP}^*, \mathcal{TK})$: system definition is similar to SCESW scheme, as illustrated in Algorithm 3.

Algorithm 4: Definition of PT-SCESW scheme.

`Enc` finally outputs ciphertexts $\mathcal{CP} = (I, F) = (I, (f_0, f_1, \ldots, f_{m-1}))$. The DO submits $\mathcal{CP}$ to the CSP.

Here, $(f_0, f_1, \ldots, f_{m-1})$ can be encoded into F_t to reduce storage space, where

$$F_t = \sum_{0 \le i \le m-1} f_i \cdot \left(2^{(t+1)} - 1\right)^i. \tag{6}$$

(v) $Cmp(\mathcal{CP}, \mathcal{CP}^*, \mathcal{TK}) \rightarrow (\varpi)$: given two ciphertexts $\mathcal{CP} = (I, (f_0, f_1, \ldots, f_{m-1}))$, $\mathcal{CP}^* = (I', F^*) = (I', (f'_0, f'_1, \ldots, f'_{m-1}))$, and a token $\mathcal{TK} = (d_1, d_2, \ldots, d_m)$, `Cmp` algorithm sets $j = m - 1$ and keeps producing c_j by decreasing j by 1 at each step, where

$$\begin{aligned} c_j = {} & f_j - f'_j - H_3\left(d_{j+1}, I\right) \\ & + H_3\left(d_{j+1}, I'\right) \bmod \left(2^{(t+1)} - 1\right). \end{aligned} \tag{7}$$

This algorithm stops when `Cmp` produces c_j such that $c_j \neq 0$ or when $c_j = 0$ for all $i = m-1, m-2, \ldots, 0$. If $\mathbb{N} > \mathbb{N}^*$, then $1 \le c_j \le 2^t - 1$ holds. If $\mathbb{N} < \mathbb{N}^*$, then $2^t \le c_j \le 2^{(t+1)} - 2$ holds. If $\mathbb{N} = \mathbb{N}^*$, then $c_j \equiv 0$ holds. Then we have

$$\varpi = \begin{cases} -1 & \text{if } 1 \le c_j \le 2^t - 1 \\ 0 & \text{if } c_j \equiv 0 \\ 1 & \text{if } 2^t \le c_j \le 2^{(t+1)} - 2. \end{cases} \tag{8}$$

3.3. The PT-SCESW Scheme

(1) Definitions of PT-SCESW Scheme. To efficiently support public auditing, we propose an enhanced SCESW scheme called PT-SCESW scheme. PT-SCESW scheme consists of four phases *Setup, Encryption, Auditing and Comparison*, defined in Algorithm 4. When user passes the legal authentication, he/she can do comparable queries operations over encrypted data. Thus, we mainly consider data security and comparable queries operations over ciphertexts.

(2) Details of PT-SCESW Scheme. Concrete construction of PT-SCESW scheme is defined as follows.

Setup Phase. This phase contains the `KeyGen` algorithm, which is utilized by the DO to initialize system.

The DO chooses a security parameter $k \in \mathcal{N}$, range parameter $n \in \mathcal{N}$, master key $\mathcal{MSK} \in \{0,1\}^*$, and hash functions H_1, H_2, H_3. Then he/she calculates the secret key $\mathcal{SK} = (p, q)$ and public key $\mathcal{PK} = (N = p \cdot q, g)$, where p, q are two large primes and g is the generator of a high-order cyclic group. Besides, he/she defines $\mathcal{PP} = (n, H_1, H_2, H_3)$. The DO runs `KeyGen` algorithm to generate the public parameter $\mathcal{PP}$ and secret key $\mathcal{SK}$. `Setup` phase contains `KeyGen` algorithm in SCESW scheme.

`Setup` phase outputs the secret key $\mathcal{SK}$, public key $\mathcal{PK}$, public parameter $\mathcal{PP}$, and master key $\mathcal{MSK}$.

Encryption Phase. `Par` algorithm is run by the DO to generate the num $\mathbb{N}$ which adopts the sliding window method. `Der` algorithm is used by the DO to produce the token of the num $\mathbb{N}$. `Enc` algorithm is run by the DO to generate ciphertexts of the num $\mathbb{N}$.

(i) $Par(\mathbb{N}) \rightarrow (\mathbb{N}')$, $Der(\mathcal{PP}, \mathcal{MSK}, \mathbb{N}) \rightarrow (\mathcal{TK})$: algorithms are similar to SCESW scheme.

(ii) $Enc(\mathcal{PP}, \mathcal{MSK}, \mathbb{N}) \rightarrow (\mathcal{CP})$: suppose that a public parameter $\mathcal{PP} = (n, H_1, H_2, H_3)$, master key $\mathcal{MSK}$, and number $\mathbb{N}$ are given, where

$$\mathbb{N} = (B_0, \ldots, B_{m-1}) = \sum_{0 \le i \le m-1} B_i \left(2^t\right)^i, \tag{9}$$

`Enc` algorithm randomly chooses a token $\mathcal{TK} = (d_1, d_2, \ldots, d_m)$ and a random number $I \in \{0,1\}^k$. Next `Enc` generates

$$f_i = H_3(d_{i+1}, I) + H_2(\mathcal{MSK}, d_{i+1}) + B_i \bmod \left(2^{(t+1)} - 1\right) \quad (i = m-1, \ldots, 0). \tag{10}$$

`Enc` finally outputs ciphertexts $\mathcal{CP} = (I, (f_0, f_1, \ldots, f_{m-1}))$. Here, $(f_0, f_1, \ldots, f_{m-1})$ can be encoded into an integer F_t to make ciphertexts shorter, where

$$F_t = \sum_{0 \le i \le m-1} f_i \cdot \left(2^{(t+1)} - 1\right)^i. \tag{11}$$

The DO regards $F = (f_0, f_1, \ldots, f_{m-1})$ as m parts. For each file block f_i, $(i = 0, \ldots, m-1)$, the DO computes tag $x_i = g^{m_i} \bmod N$. Then the user constructs the PMT according to the data block tags $X = \{x_1, \ldots, x_m\}$ and calculates root value A_{root} as the metadata, where $A_{\text{root}} = (A_{\text{root}} \cdot p, A_{\text{root}} \cdot r, A_{\text{root}} \cdot v)$. A and $\mathcal{SK}$ can be kept by the DO, while file F and set $\varphi = \{X, \text{PMT}\}$ can be sent to the CSP.

Auditing Phase. `ChalGen` algorithm is run by the verifier to produce the auditing challenge $\mathcal{CL}$. `ProofGen` algorithm is used by the TPA to generate the auditing proof $\mathcal{P}$. `ChalGen` algorithm is conducted by the TPA to produce auditing results.

(i) $ChalGen(s) \rightarrow (\mathcal{CL})$: the verifier randomly chooses the secret parameter $s \in Z_N^*$ and calculates the public parameter $g_s = g^s \bmod N$. The verifier randomly chooses c weight coefficient pairs $Q = \{i_j, a_j\}_{j=1,\ldots,c}$, where $i_j \in [1, m]$ is different from each other, $a_j \in \{0,1\}^l$. Then the verifier sends the auditing request to the TPA, and TPA sends auditing challenge $\mathcal{CL} = \{g_s, Q\}$ to the CSP.

(ii) $ProofGen(g_s, F, \varphi, \mathcal{CL}) \rightarrow (\mathcal{P})$: upon receiving the $\mathcal{CL} = \{g_s, Q\}$ sent by the verifier, the CSP computes $\mathcal{P}_1 = g_s^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N$, $\mathcal{P}_2 = \{x_{i_j}, \Upsilon_{i_j}\}_{j=1,\ldots,c}$. Then the CSP returns auditing proof $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ to the TPA.

(iii) $ProofCheck(PK, \mathcal{CL}, metadata, \mathcal{P}) \rightarrow (correct, \perp)$: upon receiving the auditing proof $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$, the TPA conducts Algorithm 2 to verify $(A, x_{i_j}, \Upsilon_{i_j}) \rightarrow \{\texttt{correct}, \perp\}$, in which $j = 1, \ldots, c$. If Algorithm 2 outputs `correct`, it means tags corresponding to the auditing request are correct. Then, the TPA computes $\mathcal{P}_3 = \prod_{j=1}^{c} (x_{i_j})^{a_j} \bmod N$. If $\mathcal{P}_3^s = \mathcal{P}_1$ holds, it outputs `correct`, which means the auditing challenge $\mathcal{CL}$ passes the verification and the system continues the `Cmp` algorithm; otherwise, it outputs $\perp$, which means the outsourced file was forged at the CSP side and the system stops the `Cmp` algorithm.

Comparison Phase. `Cmp` algorithm is employed by the user to compare the relationship of the numbers $\mathbb{N}$ and $\mathbb{N}^*$ from $\mathcal{CP}$ and $\mathcal{CP}^*$.

(i) $Cmp(\mathcal{CP}, \mathcal{CP}^*, \mathcal{TK}) \rightarrow (\omega)$: `Cmp` algorithm is similar to SCESW scheme. Given two ciphertexts $\mathcal{CP} = (I, (f_0, f_1, \ldots, f_{m-1}))$, $\mathcal{CP}^* = (I', F^*) = (I', (f_0', f_1', \ldots, f_{m-1}'))$ and a token $\mathcal{TK} = (d_1, d_2, \ldots, d_m)$, `Cmp` algorithm outputs $-1, 0, 1$.

4. Security Analysis

In this section, we will give properties of completeness and weak indistinguishability in PT-SCESW scheme by theoretical analysis, which are similar to SCESW scheme.

Theorem 1. *The PT-SCESW scheme is complete as long as H_1, H_2, and H_3 are pseudorandom functions and the CSP honestly performs operations according to the auditing challenge.*

Proof. We denote that $\mathcal{CP}$ and $\mathcal{CP}^*$ are generated from $\mathbb{N}$ and $\mathbb{N}^*$, respectively.

$$\mathbb{N} = \sum_{0 \le i \le n-1} b_i 2^i = \sum_{0 \le i \le m-1} B_i \left(2^t\right)^i;$$
$$\mathbb{N}^* = \sum_{0 \le i \le n-1} \beta_i 2^i = \sum_{0 \le i \le m-1} B_i' \left(2^t\right)^i, \tag{12}$$

where t is the window size; $m = n/t$ is the number of blocks via utilizing sliding window technology.

$$TK = (d_0, \ldots, d_m);$$
$$TK^* = \left(d_0', \ldots, d_m'\right);$$

$$\mathscr{CP} = (I, (f_0, \ldots, f_{m-1}));$$
$$\mathscr{CP}^* = \left(I', \left(f_0', \ldots, f_{m-1}'\right)\right). \tag{13}$$

From (3) we know that d_i and d_i' depend on B_i, B_{i+1}, B_i', B_{i+1}', and $\mathscr{MSK}$. Suppose that l is the first different block of $\mathbb{N}$ and $\mathbb{N}^*$, for $i = l+1, \ldots, m-1$; if the equation $B_{i+1} = B_{i+1}'$ holds, then $d_{i+1} = d_{i+1}'$ holds. Hence, if $\mathbb{N} = \mathbb{N}^*$, $c_j = 0$ holds for $i = 0, \ldots, m-1$, and then ϖ outputs 0. If $\mathbb{N} \neq \mathbb{N}^*$, for this arbitrary j,

$$\begin{aligned} c_j &= f_j - f_j' - H_3\left(d_{j+1}, I\right) \\ &\quad + H_3\left(d_{j+1}, I'\right) \bmod \left(2^{(t+1)} - 1\right) \\ &= \left(f_j - H_3\left(d_{j+1}, I\right)\right) - \left(f_j'\right. \\ &\quad \left. - H_3\left(d_{j+1}, I'\right)\right) \bmod \left(2^{(t+1)} - 1\right) \\ &= \left(H_3\left(d_{j+1}, I\right) + H_2\left(\mathscr{MSK}, d_{j+1}\right) + B_j\right. \\ &\quad \left. - H_3\left(d_{j+1}, I\right)\right) - \left(H_3\left(d_{j'+1}, I\right)\right. \\ &\quad \left. + H_2\left(\mathscr{MSK}, d_{j'+1}\right) + B_j'\right. \\ &\quad \left. - H_3\left(d_{j+1}, I'\right)\right) \bmod \left(2^{(t+1)} - 1\right) \\ &= B_j - B_j' \bmod \left(2^{(t+1)} - 1\right). \end{aligned} \tag{14}$$

For $j = 0, \ldots, m-1$, equation $B_j - B_j' = 0$ means $\mathbb{N} = \mathbb{N}^*$; equation $B_j - B_j' \neq 0$ means j is the first different bit. Specifically, if $1 \leq c_j \leq (2^t - 1)$, then $\mathbb{N} > \mathbb{N}^*$; if $2^t \leq c_j \leq (2^{(t+1)} - 2)$, then $\mathbb{N} < \mathbb{N}^*$.

Upon receiving $\mathscr{CL} = \{g_s, Q\}$ sent by the verifier, the CSP computes $\mathscr{P}_1 = g_s^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N$, $\mathscr{P}_2 = \{x_{i_j}, A_{i_j}, \Upsilon_{i_j}\}_{j=1,\ldots,c}$. According to the PMT formula, we can prove that x_{i_j} is the corresponding tag to the leaf node A_{i_j} and the result outputs correctly.

The verifier computes

$$\begin{aligned} \mathscr{P}_3 &= \prod_{j=1}^{c} \left(x_{i_j}\right)^{a_j} \bmod N \\ &= \prod_{j=1}^{c} \left(g_{f_{i_j}}\right)^{a_j} \bmod N \\ &= g^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N, \\ \mathscr{P}_3^s \bmod N &= \left(g^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N\right)^s \bmod N \\ &= g^{s \cdot \sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N = g_s^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N \\ &= \mathscr{P}_1 \bmod N. \end{aligned} \tag{15}$$

Hence, the PA-SCESW scheme is complete. □

Table 3: Comparison of computational cost in various schemes.

	PT-SCESW scheme	SCESW scheme	SCE scheme [33]
Encryption phase	$3m \cdot a + m \cdot E$	$3m \cdot a$	$4n \cdot a$
Comparison phase	$2(m - L + 1) \cdot a$	$2(m - L + 1) \cdot a$	$2(n - L + 2) \cdot a$
Auditing phase	$(2+c)E+(k'+1)a$	0	0

Note. m, n, L, a, c, k', E represent sliding window numbers of $\mathbb{N}$, original window numbers of $\mathbb{N}$, hash operations, number of windows for verification, numbers of corresponding nodes in Υ_i, and exponentiation operation, respectively.

Table 4: Comparison of storage overhead in various schemes.

	Ciphertext generation phase	Token generation phase
SCE scheme [33]	$(n+1) \cdot k$	$k + (\ln 3 / \ln 2) \cdot n$
SCESW scheme	$m \cdot k$	$k + (\ln(2^{t+1} - 1) / \ln(t+1)) \cdot m$
PA-SCESW scheme	$m \cdot k$	$k + (\ln(2^{t+1} - 1) / \ln(t+1)) \cdot m$

Note. m, n, k represent sliding window numbers of $\mathbb{N}$, original window numbers of $\mathbb{N}$, and output bits of hash operations, respectively.

Theorem 2. *The PT-SCESW scheme is weakly indistinguishable if H_1, H_2, and H_3 are pseudorandom functions.*

Proof. Let C, C_A, and C_B represent challengers. Suppose that there exists an adversary A such that $\mathrm{Adv}_{C,A} := |\Pr(\mathrm{Exp}_{C,A}^k = 0) - \Pr(\mathrm{Exp}_{C,A}^k = 1)| \geq \epsilon$ in the weak distinguishing game. Then, we know that hash function is distinguishable from the random function, which is against the assumption that they are pseudorandom functions. In particular, we consider a sequence of games by challengers C, C_A, and C_B and then prove the theorem by the hybrid argument. From literature [33], we know that $|\mathrm{Adv}_{C,A} - \mathrm{Adv}_{C_B,A}| < \epsilon$ as long as hash is a pseudorandom function as well as $\mathrm{Adv}_{C_B,A} = 0$. Hence, $\mathrm{Adv}_{C,A} < \epsilon$ and Theorem 2 is proved. □

5. Performance

In this section, we first compare our schemes with SCE scheme in *Encryption Phase*, *Comparison Phase*, and *Auditing Phase* in experiments, as shown in Tables 3 and 4, respectively. In *Auditing Phase*, auditing costs of [40] are almost of PT-SCESW scheme, so we just evaluate the actual performance of PT-SCESW scheme in experiments. These experiments are conducted using C on a Ubuntu Server 15.04 with Intel Core i5 Processor 2.3 GHz and Paring Based Cryptography (PBC). In Table 3, L is the L bit of numbers $\mathbb{N}_1 = (\beta_0, \ldots, \beta_{m-1})$, $\mathbb{N}_2 = (\gamma_0, \ldots, \gamma_{m-1})$ such that $(\beta_L, \ldots, \beta_{m-1}) = (\gamma_L, \ldots, \gamma_{m-1})$, $\beta_{L-1} < \gamma_{L-1}$ for two numbers. We randomly choose k and n, where $k = 160$ bits, $n = 1024$ bits in experimental simulations. Experimental tests are conducted for 100 times.

We will mainly focus on the computational and storage overhead. Due to the fact that SCESW scheme utilizes sliding window method, a comparison in computational and storage overhead between SCESW scheme and SCE scheme is made, which shows that SCESW scheme is cost-effective. Analysis can demonstrate that PT-SCESW scheme by using sliding window technology can relief the high computational and storage overhead. To largely reduce storage overhead, $(f_0, f_1, \ldots, f_{m-1})$ can be encoded into an integer F_t to make ciphertexts shorter in SCESW scheme and PT-SCESW scheme, shown in Table 4, where

$$F_t = \sum_{0 \le i \le m-1} f_i \cdot \left(2^{(t+1)} - 1\right)^i. \tag{16}$$

Considering computational costs, we just only consider several time-consuming operations, such as exponentiation operation "E" and Hash_i $(i = 1, 2, 3, 4, 5)$ operations. Table 3 shows the theoretical analysis of these schemes. Now we give detailed theoretical analysis of PT-SCESW scheme as an example.

(1) In *Encryption Phase,* computing $\mathscr{CP}$ and tags x_i for each block f_i can bring the exponentiation operation "E" and Hash_i $(i = 1, 2, 3)$ operation "h." Overall, this phase costs $3m \cdot a + m \cdot E$ operations.

(2) In *Comparison Phase,* costs mainly depends on computing c_j, with computing c_j only bringing Hash_i $(i = 1, 2, 3)$ operation "h." Overall, this phase costs $2(m - L + 1) \cdot a$ operations.

(3) In *Auditing Phase,* costs mainly depend on computing $g_s = g^s \bmod N$, $\mathscr{P}_1 = g_s^{\sum_c^{i=1} a_j \cdot f_{i_j}} \bmod N$, $\mathscr{P}_3 = \prod_{j=1}^{c} (x_{i_j})^{a_j} \bmod N$, and hash operations in Algorithm 2. Overall, this phase costs $(2 + c)E + (k + 1)a$ operations.

In Figure 3, we set $n = 1024$ bits and vary numbers of sliding windows m from 4 to 512, and then we notice that the encryption time in PT-SCESW scheme approximately increases with m. For example, when we set $m = 32$, encryption costs of SCESW scheme and PT-SCESW scheme are 1.214 ms and 2.034 ms, respectively, which is much more smaller than SCE scheme. Due to using sliding window method, PT-SCESW scheme and SCESW scheme can significantly reduce encryption costs.

In Figure 4, we set $n = 1024$ bits and $m = 256$, and then we notice that the comparable time in PT-SCESW scheme approximately decreases with L. For example, when setting $L = 63$, our scheme needs 4.674 ms to compare ciphertexts. In *Comparison Phase*, the PT-SCESW scheme and SCESW scheme have similar computational burden. Based on sliding window method, our PT-SCESW scheme and SCESW scheme can significantly reduce the computational overhead when these schemes are compared with SCE scheme.

In Figure 5, we set $n = 1024$ bits and vary number of windows for verification presented by c from 2 to 256, and then we notice that the auditing time in PT-SCESW scheme approximately increases with c. For example, when setting $c = 16$, our scheme needs 2.472 ms to make

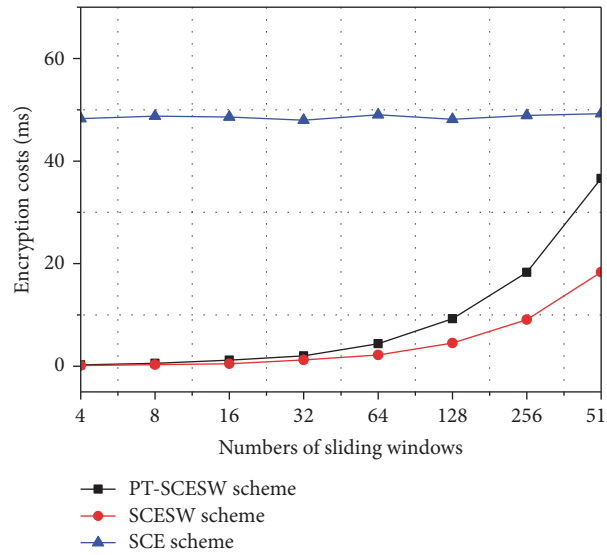

Figure 3: Encryption costs in PT-SCESW scheme.

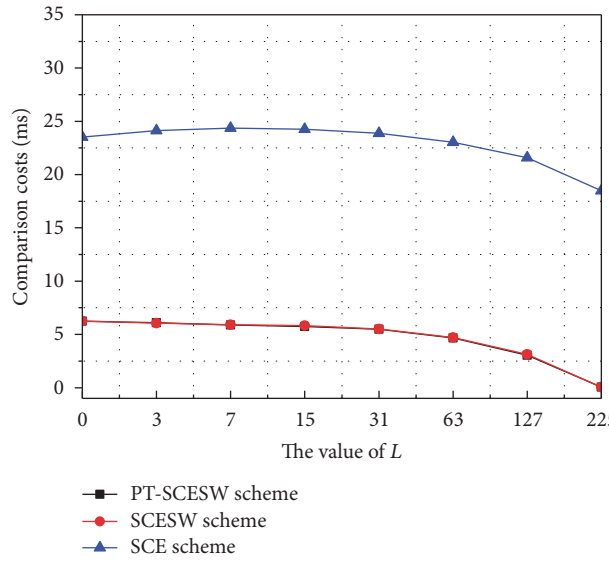

Figure 4: Comparison costs in PT-SCESW scheme.

auditing. Therefore, our PT-SCESW scheme is still acceptable in practice, especially for users with constrained computing resources and capacities.

In summary, actual performance results are completely in accord with the theoretical analysis shown in Tables 3 and 4. Exploring PT-SCESW scheme mainly focuses on achieving one property that is auditing. PT-SCESW scheme is feasible and efficient in practice applications, especially for users with constrained computing resources and capacities.

6. Conclusion

In this paper, a basic scheme named SCESW scheme is proposed for relief of the computational and storage overhead by using sliding window method. Furthermore, PT-SCESW scheme is presented for authorized users to support public

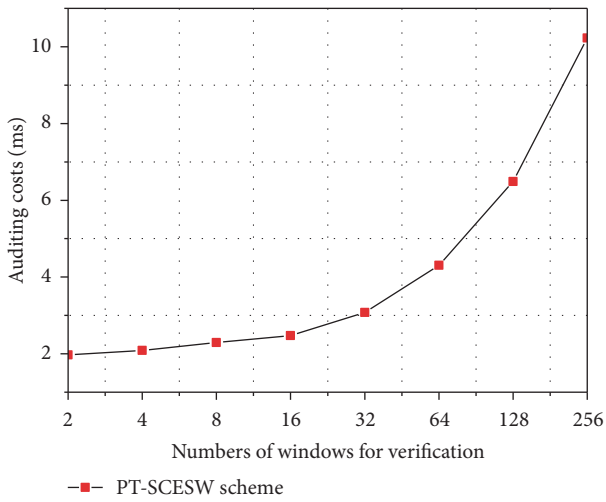

Figure 5: Auditing costs in PT-SCESW scheme.

auditing and reduce computational and storage overhead. Formal security analysis proves that PT-SCESW and SCESW schemes can guarantee data security and integrity as well as weak indistinguishability in standard model. Actual performance evaluation shows that, compared with SCE scheme, SCESW scheme and PT-SCESW scheme can relieve the computational and storage burden to some extent. In our future work, we will enhance PT-SCESW scheme by deducing its computational and storage overhead. Nevertheless, there exists another important problem to be solved. How to apply the PT-SCESW scheme to image retrieval field is rather a challenging task to be solved in cloud computing and artificial intelligence fields.

Notations

n: Number's length
a: Hash operations
$\mathscr{P}$: Auditing proof
$\mathscr{CL}$: Auditing challenge
I: $k$bits random number
m: Number of window blocks
H_i: Hash$_i$ $(i = 1, 2, 3)$ function
Υ: Smallest order node set.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National High Technology Research and Development Program (863 Program) (no. 2015AA016007.2015AA017203), the National Natural Science Foundation of China (no. 61702404), China Postdoctoral Science Foundation Funded Project (no. 2017M613080), the Fundamental Research Funds for the Central Universities (no. JB171504), the Key Program of NSFC (no. U1405255), the 111 Project (no. B16037), the Shaanxi Science & Technology Coordination & Innovation Project (no. 2016TZC-G-6-3), and the Fundamental Research Funds for the Central Universities (no. BDZ011402).

References

[1] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling smart cloud services through remote sensing: An internet of everything enabler," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 276–288, 2014.

[2] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, "Dynamic remote data auditing for securing big data storage in cloud computing," *Information Sciences*, 2015.

[3] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-Based Keyword Search over Hierarchical Data in Cloud Computing," *IEEE Transactions on Services Computing*, pp. 1-1.

[4] L. Guo, C. Zhang, J. Sun, and Y. Fang, "PAAS: a privacy-preserving attribute-based authentication system for eHealth networks," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS '12)*, pp. 224–233, IEEE, Macau, June 2012.

[5] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "VKSE-MO: verifiable keyword search over encrypted data in multi-owner settings," *Science China Information Sciences*, vol. 60, no. 12, 122105, 15 pages, 2017.

[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[7] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards Privacy-preserving Content-based Image Retrieval in Cloud Computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[8] D. Wang and P. Wang, "Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound," *IEEE Transactions on Dependable and Secure Computing*, pp. 1-1.

[9] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipfs law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[10] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "VKSE-MO: verifiable keyword search over encrypted data in multi-owner settings," *Science China Information Sciences*, vol. 60, no. 12, 2017.

[11] Y. Miao, J. Liu, and J. Ma, "Efficient keyword search over encrypted data in multi-cloud setting," *Security and Communication Networks*, vol. 9, no. 16, pp. 3808–3820, 2016.

[12] Q. Jiang, J. Ma, Z. Ma, and G. Li, "A privacy enhanced authentication scheme for telecare medical information systems," *Journal of Medical Systems*, vol. 37, no. 1, article no. 9897, 2013.

[13] Q. Jiang, J. Ma, and Y. Tian, "Cryptanalysis of smart-card-based password authenticated key agreement protocol for session initiation protocol of Zhang et al.," *International Journal of Communication Systems*, vol. 28, no. 7, pp. 1340–1351, 2015.

[14] Q. Jiang, Z. Chen, B. Li, J. Shen, L. Yang, and J. Ma, "Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1–3, 2017.

[15] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[16] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[17] M.-S. Hwang and L.-H. Li, "A new remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28–30, 2000.

[18] B.-L. Chen, W.-C. Kuo, and L.-C. Wuu, "Robust smart-card-based remote user password authentication scheme," *International Journal of Communication Systems*, vol. 27, no. 2, pp. 377–389, 2014.

[19] X. Li, J. Niu, M. Khurram Khan, and J. Liao, "An enhanced smart card based remote user password authentication scheme," *Journal of Network and Computer Applications*, vol. 36, no. 5, pp. 1365–1371, 2013.

[20] S. Ghosh, A. Majumder, J. Goswami, A. Kumar, S. P. Mohanty, and B. K. Bhattacharyya, "Swing-Pay: One Card Meets All User Payment and Identity Needs: A Digital Card Module using NFC and Biometric Authentication for Peer-To-Peer Payment," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 82–93, 2017.

[21] Y. Lu, L. Li, H. Peng, and Y. Yang, "An Enhanced Biometric-Based Authentication Scheme for Telecare Medicine Information Systems Using Elliptic Curve Cryptosystem," *Journal of Medical Systems*, vol. 39, no. 3, 2015.

[22] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social Attribute Aware Incentive Mechanism for Device-to-Device Video Distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.

[23] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially Aware Energy-Efficient Mobile Edge Collaboration for Video Distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2197–2209, 2017.

[24] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic Trust Relationships Aware Data Privacy Protection in Mobile Crowd-Sensing," *IEEE Internet of Things Journal*, pp. 1-1.

[25] P. Karras, A. Nikitin, M. Saad, R. Bhatt, D. Antyukhov, and S. Idreos, "Adaptive indexing over encrypted numeric data," in *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data, SIGMOD 2016*, pp. 171–183, USA, July 2016.

[26] D. Agrawal, A. El Abbadi, F. Emekci, and A. Metwally, "Database management as a service: Challenges and opportunities," in *Proceedings of the 25th IEEE International Conference on Data Engineering, ICDE 2009*, pp. 1709–1716, China, April 2009.

[27] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proceedings of the 34th IEEE Symposium on Security and Privacy, SP 2013*, pp. 463–477, USA, May 2013.

[28] H. Kadhem, T. Amagasa, and H. Kitagawa, "A secure and efficient Order Preserving Encryption Scheme for relational databases," in *Proceedings of the International Conference on Knowledge Management and Information Sharing, KMIS 2010*, pp. 25–35, esp, October 2010.

[29] D. Liu and S. Wang, "Programmable order-preserving secure index for encrypted database query," in *Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, pp. 502–509, USA, June 2012.

[30] R. Agrawal, J. Kiernan, R. Srikant, and Y. R. Xu, "Order preserving encryption for numeric data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*, pp. 563–574, ACM, Paris, France, June 2004.

[31] J. Furukawa, "Request-based comparable encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 8134, pp. 129–146, 2013.

[32] P. Chen, J. Ye, and X. Chen, "Efficient request-based comparable encryption scheme based on sliding window method," *Soft Computing*, vol. 20, no. 11, pp. 4589–4596, 2016.

[33] J. Furukawa, "Short comparable encryption," in *Cryptology and network security*, vol. 8813 of *Lecture Notes in Comput. Sci.*, pp. 337–352, Springer, Cham, 2014.

[34] Y. Yu, M. H. Au, G. Ateniese et al., "Identity-Based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, 2017.

[35] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[36] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5789, pp. 355–370, 2009.

[37] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[38] Y. Deswarte, J. J. Quisquater, and A. Sadane, "Remote integrity checking," in *Integrity and internal control in information systems VI*, vol. 12, pp. 1–12, 2004.

[39] D. L. Gazzoni Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," *IACR Cryptology ePrint Archive*, p. 150, 2006.

[40] 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.

[41] C. K. Koç, "Analysis of sliding window techniques for exponentiation," *Computers & Mathematics with Applications*, vol. 30, no. 10, pp. 17–24, 1995.

[42] Y. Wang, Y. Shen, H. Wang, J. Cao, and X. Jiang, "MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verifications," *IEEE Transactions on Big Data*, pp. 1-1.

[43] P. Cong, Z. Ning, F. Xue, K. Xu, B. Fan, and H. Li, "Hierarchy merkle tree wireless sensor network architecture," *International Journal of Wireless and Mobile Computing*, vol. 12, pp. 341–348, 2017.

[44] M. Szydlo, "Merkle tree traversal in log space and time," in *Advances in cryptology-EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Comput. Sci.*, pp. 541–554, Springer, Berlin, Germany, 2004.