Journal of Systems Architecture 139 (2023) 102876

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Provably secure public key encryption with keyword search for data outsourcing in cloud environments

Sudeep Ghosh [a,b], SK Hafizul Islam [b,*], Abhishek Bisht [c], Ashok Kumar Das [c]

[a] Department of Information Technology, Guru Nanak Institute of Technology, Kolkata, West Bengal 700114, India
[b] Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India
[c] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India

ARTICLE INFO

Keywords:
Cloud server
Data outsourcing
Keyword search
Public key encryption
Probable security

ABSTRACT

In recent days, the application of cloud computing has been gaining significant popularity among people. A considerable amount of data are being stored in the cloud server. However, data owners outsource their encrypted data to the cloud for various security reasons. Unfortunately, encrypted data cannot be searched, like plaintext data. So how to search encrypted data is an interesting problem in this era. Many public key encryption with keyword search (PEKS) schemes have been designed in the literature. However, most of them cannot prevent keyword-guessing attacks. In this paper, we develop a provably secure PEKS scheme in the random oracle model. This scheme may be used for secure email access from an email server containing a list of encrypted keywords. The proposed scheme can resist keyword-guessing attacks, and offer ciphertext and trapdoor indistinguishability properties. We use the data owner's private key during encryption to prevent keyword-guessing attacks. Using the data owner's public key in the verification phase ensures the resilience of keyword-guessing attacks. Finally, the proposed scheme has been tested on a real testbed, and the results show that it can be used in the cloud computing scenario to search for keywords on encrypted data.

1. Introduction

Cloud computing offers on-demand services or access to resources, such as tools, data storage, server, database, software, etc., through the Internet. Cloud storage has been turned into an encouraging paradigm because of the excessive data growth in recent years due to the massive applications of Facebook, Twitter, Youtube, Linkedin, etc. It is more convenient to store data in cloud storage than to keep files on a hard drive or local storage device. Cloud storage provides universal access to on-demand remotely configured storage, as shown in Fig. 1. However, the files outsourced to a cloud server may also comprise sensitive data, like company's financial documents and patients' health records, which may incur protection and privacy issues. One well-known strategy is to encrypt the data before moving it to the cloud server to ensure data confidentiality. However, encrypted data make their utilization more difficult, especially the ability for information retrieval.

With the advancement of the client–server model in cloud computing, more and more confidential and sensitive data are being stored in the cloud server (**CS**) [1,2]. On the other hand, protecting those data stored in **CS** has become a significant concern for the client–server storage model. Therefore, the data owner (**DO**) keeps the data in **CS** in an encrypted format to protect it from outside attacks. However, encrypted data cannot be searched by traditional searching techniques. To search encrypted data, Boneh et al. [3] first introduced the concept of public key encryption with keyword search (PEKS) scheme, which was used for secure email communication. This is secured based on the Bilinear Diffie–Hellman (BDH) assumption in the random oracle model (ROM). The framework of a PEKS is illustrated in Fig. 2.

In a PEKS scheme, three parties are involved. We denote Alice as the data owner (**DO**), Bob as the data consumer (**DC**), and the cloud server (**CS**) that provides a data storage facility. Alice wants to share a sensitive file, F, with Bob. At first, Alice selects some keywords $\{w_j : 1 \le j \le n\}$ from F and encrypts them using the PEKS scheme using Bob's public key PK_{DC}. Let $\{C_{w_j}\}$ be the ciphertext of $\{w_j : 1 \le j \le n\}$, and Alice uploads it in **CS** along with the ciphertext of F, which is encrypted with possibly another encryption scheme. Now, Bob wants to search for a document containing the keyword w_j. For that purpose, Bob computes a trapdoor T_{w_j} using his secret key $\mathbf{SK}_{DC}$ and sends it to **CS**. Now **CS** executes the **Test** algorithm using T_{w_j} and C_{w_j} to check whether the keyword w_j is present in C_{w_j}. After finishing the search, **CS** returns the search result to Bob. During the search, **CS** could not know the content of F and w_j. In Boneh et al.'s PEKS scheme [3], a secure channel is needed to send the encrypted data to **CS**. Nevertheless, using

* Corresponding author.
E-mail address: hafi786@gmail.com (S.H. Islam).

https://doi.org/10.1016/j.sysarc.2023.102876
Received 2 January 2023; Received in revised form 7 March 2023; Accepted 6 April 2023
Available online 18 April 2023

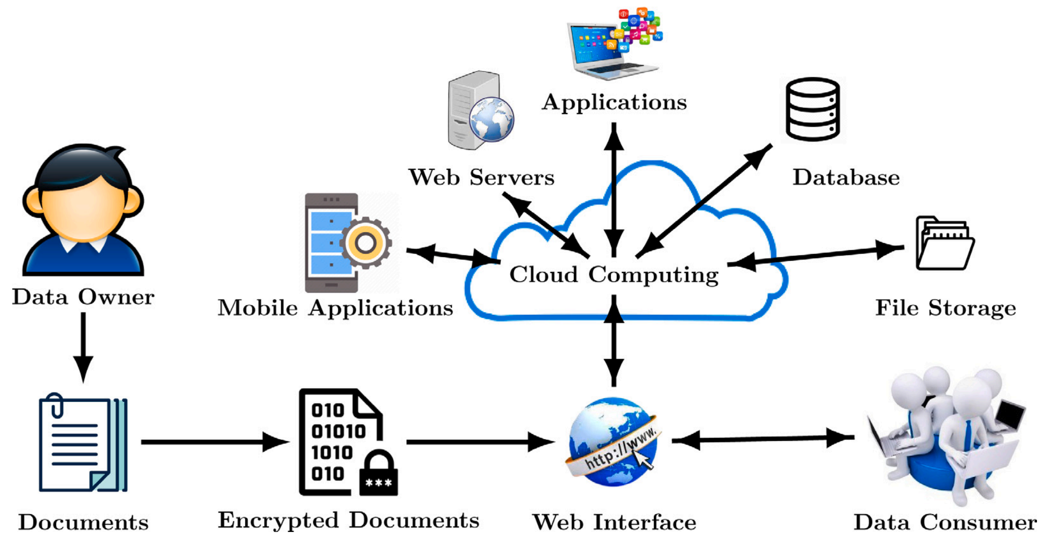

Fig. 1. General framework of cloud storage and secure data access.

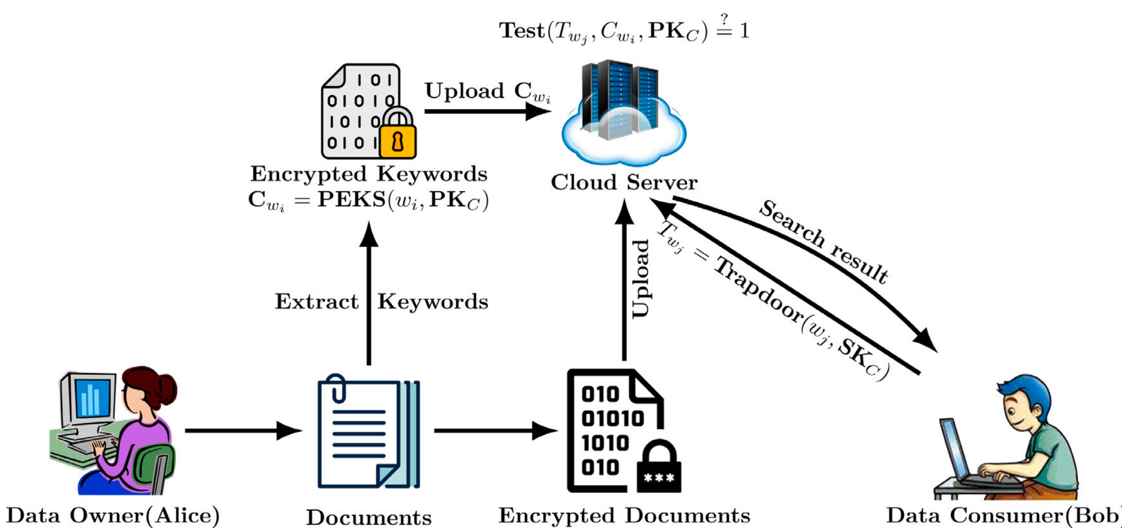

Fig. 2. Schematic flow diagram of PEKS scheme.

a secure public channel in real-life is not a good solution. Later, other researchers [4,5] introduced several schemes without using the secure public channel.

1.1. Literature survey

Boneh et al. [3] first introduced the concept of a bilinear pairing-based PEKS scheme. We denote it by the BPEKS scheme. The BPEKS scheme is secure in the random oracle model based on the hardness assumption of the Bilinear Diffie–Hellman (BDH) problem. Moreover, they have shown that BPEKS is secure against chosen keyword attacks in the random oracle model. However, we found that the BPEKS scheme is vulnerable to keyword-guessing attacks.

1. **Keyword-guessing attack:** We assume that **CS** is ***honest-but-curious*** to recover the keyword present in a trapdoor T_w. From the keyword space, a curious **CS** can choose a keyword w_i and compare whether it is equal to the keyword present in T_w. This attack works as follows:

   (a) Assume that a keyword w to be searched by **DC** over some encrypted data. Next, **DC** computes a trapdoor T_w, and sends it to **CS** for searching.

   (b) Now, **CS** chooses a keyword w_1 from the keyword space, and runs the **PEKS** algorithm to encrypt w_1. Let the ciphertext generated by **PEKS** be C_{w_1}. **CS** now runs the **Test** algorithm on C_{w_1}, T_w using the public key of **DC**. If the test fails, **CS** chooses another keyword from keyword space, and executes the **Test** algorithm using the same input as above. **CS** will continue the execution of **Test** algorithm until to have the file owned by **DC** file that contains the keyword w.

   It is a fact that the keyword space in a real-life application is usually not so big, therefore **CS** could perform a keyword-guessing attack in a reasonably short time.

2. **Trapdoor security:** Boneh et al. [3] used a secure channel in their BPEKS scheme so that **DC** can securely send the trapdoor to **CS**. Nevertheless, in real-life use of secure channels is not a good solution. To solve this problem, Baek et al. [4] first proposed a secure channel-free PEKS (SCF-PEKS) scheme. However, Rhee et al. [5] described that the security model proposed in [4] had limited capabilities to the adversary, and the proposed scheme was not secure.

In recent days, various PEKS schemes and their variants have been proposed. We can classify these works into the following types: (i) PEKS for multi-user access control [6–8], (ii) PEKS for trapdoor privacy [5,9,10], (iii) PEKS for flexible keyword search [11–13], and (iv) PEKS for fuzzy keyword search [14,15]. The concept of searchable encryption (SE) was first introduced in 2000 by Song et al. [16] using symmetric key encryption. Later, Golle et al. [11] discussed the first SE scheme using conjunctive keyword and also pointed out the limitation of the single-user model associated with Song et al.'s scheme [16]. However, the search time of Golle et al.'s scheme is linear and limited to a small keyword size. Cash et al. [17] proposed the first SE scheme, which is nonlinear, supports boolean queries and can be applied to large data sizes. However, because the schemes mentioned above are based on symmetric key cryptography, they all have key exchange issues.

The PEKS was introduced by Boneh et al. [3]. In 2008, Baek et al. [4] designed a PEKS scheme to eliminate the use of a secure channel in the PEKS scheme. In Bake et al.'s PEKS scheme, only a particular server can verify whether the keyword of the trapdoor is similar to any stored keyword. Nevertheless, later, Rahee et al. [5] showed that the security model in [4] is inefficient, and the capability of the adversary is limited. Therefore, Rahee et al. [5] redesigned the security model in [4], and proposed the concept of trapdoor indistinguishability. Also, they have shown that the scheme in [5] is secure from offline keyword-guessing attacks. Zaho et al. [18] proposed a secure method against trapdoor indistinguishability in the chosen-plaintext attack (CPA) model. Hu et al. [19,20] improved the scheme proposed in [4]. Hu et al. showed that the scheme in [4] is not secure against the offline keyword-guessing attack if the server is malicious. Unfortunately, Ni et al. [21] showed that schemes in [19,20] are not secure against offline keyword-guessing attacks if the server is malicious. In 2017, Huang et al. [22] proposed a secure PEKS scheme against keyword-guessing attacks. Nevertheless, the trapdoor is fixed in this scheme, i.e., the scheme always creates the same trapdoor for the same keyword. As the trapdoor is fixed, an attacker may get some information regarding the encryption pattern. Also, the scheme does not offer ciphertext indistinguishability in the CPA model. In [23], the authors show that the schemes [3,4] may reveal the keywords due to the leakage of encryption patterns. Other than the keyword search, some new techniques have been discussed in the public key setting, like verifiable keyword search [24–26], decryptable searchable encryption [27] and proxy re-encryption with keyword search [28,29]. In [30], the authors proposed a PEKS scheme with cryptographic reverse firewalls (SPKE-CRF). They used the JPBC library to implement the scheme and showed that the scheme could resist the chosen keyword attack. In [31], the authors proposed a PEKS technique with parallelism and forward privacy, namely the parallel and forward private searchable public key encryption (PFP-SPE). They have shown that their proposed scheme achieves parallelism and forward privacy, but the storage cost is slightly higher than other schemes.

1.2. Motivations and contributions

In the literature, we found that many PEKS schemes and their variants are available. However, only some of them can prevent keyword-guessing attacks. Also, ensuring the trapdoor and ciphertext security are challenging for the PEKS scheme. For example, some schemes required a secure channel [3] to send data to the cloud server. We list the main achievements of this paper as follows:

1. We propose a secure ***channel-free*** PKES scheme to prevent keyword-guessing attacks and to ensure trapdoor and ciphertext security in the CPA model.
2. To prevent the keyword-guessing attack by an ***honest-but-curious*** server, we use the data owner's private key so that server cannot encrypt a chosen keyword and thus, the server cannot relaunch a keyword-guessing attack successfully.
3. To ensure trapdoor security, only the server can test the trapdoor because we assume that an adversary can get a trapdoor but cannot execute the **Test** algorithm successfully.
4. The security proof is based on the hardness assumption of Bilinear Diffie–Hellman (BDH), and Hash Diffie–Hellman (HDH) problems in the random oracle model.
5. We consider various cryptographic operations and compute their running cost in pairing-based cryptography (PBC) with five different security levels recommended by the NIST.
6. We compare the execution and communication costs of the proposed PEKS scheme with related PEKS schemes and found that the proposed scheme is efficient. The simulation results show that our scheme satisfies all the required security and security attributes.
7. We implement the proposed PEKS scheme in a real test-bed environment using the PBC library and C socket programming. We compute the execution time taken to generate encrypted keywords while varying the number of keywords. We found that the keyword generation time increases linearly with the number of keywords.

1.3. Paper organization

The rest of the paper is organized as follows. Section 2 briefly describes various mathematical preliminaries. We demonstrate the security model of a PEKS scheme in Section 3. Next, we describe the proposed PEKS scheme in Section 4. In Section 5, we prove its provable security in the CPA model. We give the experimental and comparison results with other existing PEKS schemes in Sections 6 and 7. Furthermore, finally, we conclude the paper in Section 8.

2. Preliminaries

2.1. Bilinear pairing

In the construction of many cryptography schemes, bilinear pairing [32] plays a vital role. Assume that $\mathbf{G}_1$, and $\mathbf{G}_2$ be two multiplicative cyclic groups of prime order p. The bilinear mapping $\mathbf{e}$ is defined as $\mathbf{e} : \mathbf{G}_1 \times \mathbf{G}_1 \rightarrow \mathbf{G}_2$. The bilinear mapping $\mathbf{e}$ has the following properties:

- **Bilinearity:** For any $g,\ h \in \mathbf{G}_1$ and $a, b \in \mathbb{Z}_p^*$, $e(g^a, h^b) = e(g, h)^{ab}$.
- **Non-degeneracy:** Let g be a generator of $\mathbf{G}_1$, then $e(g, g)$ is a generator of $\mathbf{G}_2$.
- **Computability:** For any $g, h \in \mathbf{G}_1$, there must exit an algorithm $\mathcal{A}$ that can compute $e(g, h)$ within polynomial time.

Definition 1 (*Negligible Function*)**.** Given an integer λ as security parameter, a negligible function, denoted by ϵ is defined as $\epsilon \leq \frac{1}{\lambda^l}$ if $\forall\ l > 0$, $\exists\ \lambda_0$ such that $\forall\ \lambda \geq \lambda_0$.

Definition 2 (*Bilinear Diffie–Hellman (BDH) Assumption*)**.** Let g is a generator of $\mathbf{G}_1$. Given $g, g^a, g^b, g^c \in \mathbf{G}_1$, $a, b, c \in \mathbb{Z}_p^*$, it is computationally hard for any probabilistic polynomial time-bounded (PPT) algorithm $\mathcal{B}$ to compute $e(g, g)^{abc} \in \mathbf{G}_2$. The probability of success in solving the BDH problem in $\mathbf{G}_1$ is defined as

$$\mathbf{Adv}_{\mathcal{B}}^{BDH}(t) = |\mathbf{Pr}[\mathcal{B}(g, g^a, g^b, g^c) = e(g, g)^{abc} \ :\ a, b, c \xleftarrow{R} \mathbb{Z}_p^*]|$$

For any PPT algorithm $\mathcal{B}$, $\mathbf{Adv}_{\mathcal{B}}^{BDH}(t) \leqslant \epsilon$.

Definition 3 (*Decisional BDH (DBDH) Assumption*)**.** Let g is a generator of $\mathbf{G}_1$. Given $g, g^a, g^b, g^c \in \mathbf{G}_1$, $a, b, c \in \mathbb{Z}_p^*$, it is computationally hard for any PPT algorithm $\mathcal{B}$ to distinguish $e(g, g)^{abc} \in \mathbf{G}_2$ from a random element $e(g, g)^z \in \mathbf{G}_2$, where $z \in \mathbb{Z}_p^*$. The probability of success in solving DBDH problem in $\mathbf{G}_1$ is defined as

$$\mathbf{Adv}_{\mathcal{B}}^{DBDH}(t) = |\mathbf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1 : a, b, c \xleftarrow{R} \mathbb{Z}_p^*] - \mathbf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^z) = 1 : z \xleftarrow{R} \mathbb{Z}_p^*]|$$

For any PPT algorithm $\mathcal{B}$, $\mathbf{Adv}_{\mathcal{B}}^{DBDH}(t) \leqslant \epsilon$. The probability has been taken over the random choice of $g \xleftarrow{R} \mathbf{G}_1$, $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$ and random coin tossed by $\mathcal{B}$ [33,34].

Definition 4 (*Hash Diffie–Hellman (HDH) Assumption*)**.** Let ℓ is a number and $H : \{0,1\}^* \to \{1,0\}^\ell$. Given $(g, g^a, g^b, H(g^c)) \in \mathbf{G}_1 \times \{0,1\}^\ell$ and $H : \{0,1\}^* \to \{0,1\}^\ell$. A PPT algorithm $\mathcal{B}$ outputs 1 if $ab = c$, otherwise 0. The probability of success in solving HDH problem in $\mathbf{G}_1$ is defined as

$$\mathbf{Adv}_{\mathcal{B}}^{HDH}(t) = |\mathbf{Pr}[\mathcal{B}(g, g^a, g^b, H(g^{ab})) = 1 : g \xleftarrow{R} \mathbf{G}_1, a, b \xleftarrow{R} \mathbb{Z}_p^*] - \mathbf{Pr}[\mathcal{B}((g, g^a, g^b, \eta)) = 1 : g \xleftarrow{R} \mathbf{G}_1, \eta \xleftarrow{R} \{0,1\}^\ell, a, b \xleftarrow{R} \mathbb{Z}_p^*]|$$

For any PPT algorithm $\mathcal{B}$, $\mathbf{Adv}_{\mathcal{B}}^{HDH}(t) \leqslant \epsilon$. Here the probability is taken over the random value of $g \xleftarrow{R} \mathbf{G}_1$, the random value of $\eta \in \{0,1\}^\ell$ and the random elements of $a, b \xleftarrow{R} \mathbb{Z}_p^*$ [35].

3. Security model

We follow the security model defined in [3], which requires that there is no PPT adversary $\mathcal{A}$ who can distinguish with probability more than $\frac{1}{2}$ a computed trapdoor (and ciphertext), and a trapdoor (and ciphertext) selected randomly. However, there is a significant difference between the security model defined in [3] and our security model proposed in this paper. In our security model, the adversary is also given access to the trapdoor generation oracle and ciphertext generation oracle. Because in the proposed scheme, any third party other than the data owner cannot encrypt a keyword as we use the data owner's private key to encrypt a keyword. We can classify the type of adversary as follows:

1. $\mathcal{A}_1$: A malicious server, and
2. $\mathcal{A}_2$: An outside attacker including malicious data consumer.

- **Game 1:** With this game, we will define the ***trapdoor indistinguishability*** in the CPA model to provide the ***trapdoor privacy***. This game is played between a challenger $\mathcal{C}$ and $\mathcal{A}_1$.
  1. Given a security parameter λ, $\mathcal{C}$ generates the list of global system parameter **param**, and the public keys of **DO**, **DC**, and **CS** as $\mathbf{PK_O}$ and $\mathbf{PK_C}$, and $\mathbf{PK_{CS}}$, respectively. Now, $\mathcal{A}_1$ invoked (**param**, $\mathbf{PK_O}$, $\mathbf{PK_C}$, $\mathbf{PK_{CS}}$) as input to this game.
  2. $\mathcal{A}_1$ is allowed to issue various queries adaptively to the following oracles for polynomially several times.
     - **Trapdoor Oracle** $\mathcal{O}_T$: Assume that a keyword w, the corresponding trapdoor T_w computed by the oracle $\mathcal{O}_T$ with respect to $\mathbf{PK_{CS}}$, $\mathbf{PK_O}$, $\mathbf{SK_C}$, and sends it to $\mathcal{A}_1$.
     - **Ciphertext Oracle** $\mathcal{O}_C$: Assume a keyword w, the corresponding ciphertext C computed by the oracle $\mathcal{O}_C$ with respect to $\mathbf{SK_O}$ and $\mathbf{PK_C}$, and sends it to $\mathcal{A}_1$.
  3. Now $\mathcal{A}_1$ chooses two keywords (w_0^*, w_1^*), and they have not been quarried for trapdoor and ciphertext earlier. $\mathcal{A}_1$ submits (w_0^*, w_1^*) to $\mathcal{C}$ as the challenge keywords. $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ and computes the trapdoor $T_{w_b^*} \leftarrow (w_b^*, \mathbf{PK}_O, \mathbf{SK}_C, \mathbf{PK}_{CS})$, and returns it to $\mathcal{A}_1$.
  4. $\mathcal{A}_1$ will continue to issue queries to $\mathcal{O}_T$ and $\mathcal{O}_C$ but the condition is that neither w_0^* nor w_1^* has not been queried for obtaining corresponding trapdoors.
  5. $\mathcal{A}_1$ outputs it guess $b' \in \{0,1\}$, and it wins the game if $b' = b$ holds.

  We define the advantages of $\mathcal{A}_1$ to break the trapdoor indistinguishability in the CPA model within a polynomial time t is $\mathbf{Adv}_{\mathcal{A}_1,PEKS}^{T_{w_b}^*-ind}(t) = |\mathbf{Pr}[b' = b] - \frac{1}{2}|$.

  Definition 5 (*Trapdoor Indistinguishability*)**.** A PEKS scheme satisfies trapdoor indistinguishability in the CPA model if for any PPT adversary $\mathcal{A}_1$, $\mathbf{Adv}_{\mathcal{A}_1,PEKS}^{T_{w_b}^*-ind}(t) \leq \epsilon$.

- **Game 2:** With this game, we define the ***ciphertext indistinguishability*** in the CPA model to provide the ***ciphertext security***. This game is played between $\mathcal{A}_2$ and $\mathcal{C}$.
  1. Given a security parameter λ, $\mathcal{C}$ will generate the list of global system parameter **param**, the public keys of **DO**, **DC**, and **CS** as $\mathbf{PK_O}$ and $\mathbf{PK_C}$, and $\mathbf{PK_{CS}}$, respectively. $\mathcal{A}_2$ invokes (**param**, $\mathbf{PK_O}$, $\mathbf{PK_C}$) as input to this game.
  2. As in **Game 1**, $\mathcal{A}_2$ issues the queries to $\mathcal{O}_T$ and $\mathcal{O}_C$.
  3. Now $\mathcal{A}_2$ chooses two keywords (w_0^*, w_1^*), and they have not been quarried for trapdoor and ciphertext earlier. $\mathcal{A}_2$ submits (w_0^*, w_1^*) to $\mathcal{C}$ as the challenge keywords. $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$, computes the ciphertext $C_{w_b^*} \leftarrow (w_b^*, \mathbf{PK}_O, \mathbf{SK}_C)$, and returns it to $\mathcal{A}_2$.
  4. $\mathcal{A}_2$ will continue to issue queries to $\mathcal{O}_T$ and $\mathcal{O}_C$, but the condition is that neither w_0^* nor w_1^* has not been quarried before.
  5. $\mathcal{A}_2$ outputs its guess $b' \in \{0,1\}$, and it wins the game if $b' = b$.

  We define the advantages of $\mathcal{A}_2$ to break the ciphertext indistinguishability in the CPA model within polynomial time t is $\mathbf{Adv}_{\mathcal{A}_2,PEKS}^{C_{w_b}^*-ind}(t) = |\mathbf{Pr}[b' = b] - \frac{1}{2}|$

  Definition 6 (*Ciphertext Indistinguishability*)**.** A PEKS scheme satisfies ciphertext indistinguishability in the CPA model if for any PPT adversary $\mathcal{A}_2$, $\mathbf{Adv}_{\mathcal{A}_2,PEKS}^{C_{w_b}^*-ind}(t) \leq \epsilon$.

4. Proposed PEKS scheme

The proposed PEKS scheme is depicted in Fig. 3. It is designed to offer a secure client–server storage system in cloud environments. The proposed scheme has three entities: (i) data owner (**DO**), (ii) data consumer (**DC**), and (iii) a cloud server (**CS**). **DO** stores her important data in encrypted form to **CS**. **DO** also upload the encrypted keywords to **CS** using a public channel. **DO** uses her private key $\mathbf{SK_O} = d_o$ and the public keys $\mathbf{PK_C} = g^{d_c}$ and $\mathbf{PK_{CS}} = g^{d_s}$ of **DC**, and **CS** to encrypt the keywords $\mathcal{W} = \{w_1, w_2, w_3, \dots, w_n\}$. After encrypting the keywords, **DO** uploads it to **CS** using a public channel. **DC** will ask **CS** to search the encrypted data using one of $\{w_1, w_2, w_3, \dots, w_n\}$ to find the required data from **CS**. To do so, **DC** creates a trapdoor T_{w_i} for the keyword w_i, and sends it to **CS** using a public channel. **DC** uses his private key $\mathbf{SK_C} = d_c$, the public keys $\mathbf{PK_O} = g^{d_o}$, and $\mathbf{PK_{CS}} = g^{d_s}$ of **DO** and **CS** to create the trapdoor T_{w_i}. **CS** then verifies the correctness of T_{w_i}. After finishing the search, **CS** sends the search result to **DO**. Nevertheless, during the search, **CS** does not know the content of T_{w_i}. The list of notations used in this paper is provided in Table 1.

As mentioned above, the main aim of designing the proposed scheme is to prevent keyword-guessing attacks and provide trapdoor indistinguishability. To prevent keyword-guessing attacks, we use the private key $\mathbf{SK_O} = d_o$ of **DO** during the encryption process as a part of the input. The proposed scheme has ensured that **CS** could not encrypt any keyword. Instead, the encryption algorithm needs the private key of **DO**. Therefore, **CS** could not relaunch a keyword-guessing attack as it

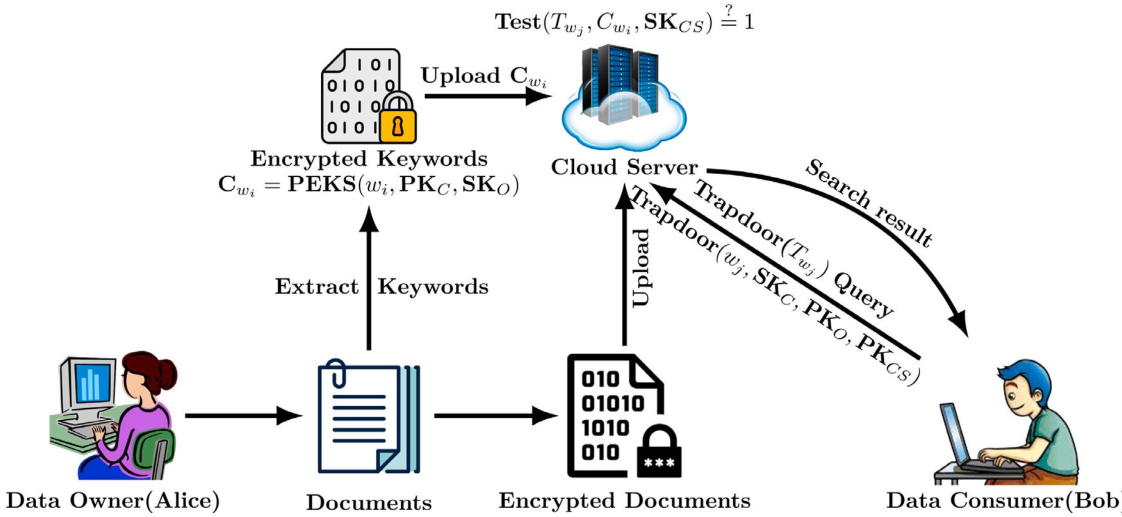

Fig. 3. Schematic flow diagram of the proposed PEKS scheme.

Table 1
Notations.

Notation	Meaning
λ	Security parameter
p	Large prime number of λ-bit
$\mathbf{G}_1$, $\mathbf{G}_2$	Multiplicative cyclic group of order p
g	Generator of $\mathbf{G}_1$
$\mathbf{e}$	Bilinear mapping
$\mathcal{D}_w$	Keyword space
DO/DC/CS	Data owner/Data consumer/Cloud server
$d_o/d_c/d_s$	Private key of **DO/DC/CS**
$g^{d_o}/g^{d_c}/g^{d_s}$	Public key of **DO/DC/CS**
$\mathcal{C}/\mathcal{A}_i$	Challenger/Adversary
$\mathcal{O}_T/\mathcal{O}_C$	Trapdoor/Ciphertext oracle
C/T	Ciphertext/Trapdoor
H_1, H_2, H_3	Cryptographic hash function

could not create ciphertext without the secret key of **DO**. Furthermore, we have designed the trapdoor so that an outside attacker cannot get information from it and prevent information leakage.

1. **Setup**(2^λ): This deterministic polynomial time-bounded algorithm is executed by **CS**. Given a security parameter λ, let p be a large prime number of size λ-bit. Assume that $\mathbf{G}_1$, and $\mathbf{G}_2$ be two multiplicative cyclic groups of prime order p. Let $\mathbf{e} : \mathbf{G}_1 \times \mathbf{G}_1 \rightarrow \mathbf{G}_2$ is a bilinear mapping. **CS** chooses a generator g of $\mathbf{G}_1$, and three hash functions, $\mathbf{H}_1 : \{0,1\}^* \rightarrow \mathbf{G}_1$, $\mathbf{H}_2 : \{0,1\}^* \rightarrow \mathbf{G}_1$, $\mathbf{H}_3 : \mathbf{G}_2 \rightarrow \{0,1\}^\eta$, where η signifies the length of the digest, and it depends on the specific hash function. **CS** publishes the parameters **param** $= \{\mathbf{G}_1, \mathbf{G}_2, g, p, e, \mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3, \mathcal{D}_w\}$.
2. **KeyGen**(2^λ): This deterministic polynomial time-bounded algorithm executed by **DO**, **DC**, and **CS** to generate their secret and public key pairs. **DO** chooses a number $d_o \in \mathbb{Z}_p^*$ uniformly at random as a secret key, i.e., $\mathbf{SK_O} = d_o$, and computes the public key as $\mathbf{PK_O} = g^{d_o}$. **DC** chooses a number $d_c \in \mathbb{Z}_p^*$ uniformly at random as secret key, i.e., $\mathbf{SK_C} = d_c$, and computes the public key as $\mathbf{PK_C} = g^{d_c}$. **CS** chooses a number $d_s \in \mathbb{Z}_p^*$ uniformly at random as a secret key, i.e., $\mathbf{SK_{CS}} = d_s$ and computes the public key as $\mathbf{PK_{CS}} = g^{d_s}$.
3. **PEKS**$(\mathbf{param}, \mathbf{PK}_C, \mathbf{SK}_O, \mathcal{W})$: This probabilistic polynomial-time bounded algorithm executed by **DO**. Given a list of keywords $\mathcal{W} = \{w_1, w_2, w_3, \dots, w_n\}$, for each w_i, **DO** chooses a number $r_i \in \mathbb{Z}_p^*$ uniformly at random, and computes:

$$C_1 = (PK_C)^{d_o r_i} = g^{d_o d_c r_i},$$
$$C_2 = (PK_O)^{r_i} = g^{d_o r_i},$$
$$C_{w_i} = H_3(e(C_1, H_2(w_i))) = H_3(e(g, H_2(w_i)^{d_c d_o r_i})), \text{ and}$$
$$C = \{C_1, C_2, C_{w_1}, C_{w_2}, \dots, C_{w_n}\}.$$

DO sends the ciphertext C to CS over a public channel, and CS stores it in the storage.
4. **Trapdoor**$(\mathbf{param}, \mathbf{PK}_O, \mathbf{SK}_C, \mathbf{PK}_{CS}, \mathcal{W})$: This probabilistic polynomial time-bounded algorithm executed by **DO**. **DC** chooses a keyword w_j, a number $r_j \in \mathbb{Z}_p^*$ uniformly at random, and computes the followings:

$$T_1 = H_1((PK_O)^{d_c r_j}) = H_1(g^{d_c d_o r_j}),$$
$$T_2 = (PK_{CS})^{d_c r_j} = g^{d_c d_s r_j},$$
$$T_3 = g^{r_j},$$
$$T_{w_j} = (H_2(w_j) \cdot T_1) = (H_2(w_j) \cdot H_1(g^{d_c d_o r_j})), \text{ and}$$
$$T = (T_1, T_2, T_3, T_{w_j}).$$

DC sends the trapdoor T to CS over a public channel.
5. **Test**$(param, T_{w_j}, C_{w_i}, SK_{CS})$: This deterministic polynomial time-bounded algorithm executed by **CS**. It takes **param**, $C = \{C_1, C_2, C_{w_1}, C_{w_2}, \dots, C_{w_n}\}$, and $T = (T_1, T_2, T_3, T_{w_j})$ as input, and computes $\phi = e(C_1, T_{w_j})/e(C_1, T_1)$. Next, **CS** checks whether $H_3(\phi) \cdot e(C_1, T_3)^{d_s} = C_{w_i} \cdot e(C_2, T_2)$ holds. If it is satisfied, **CS** returns "YES"; otherwise, returns "NO".

The execution of **PEKS**$(\cdot)$, **Trapdoor**$(\cdot)$, and **Test**$(\cdot)$ algorithms are further explained in Figs. 4, 5 , and 6, respectively.

5. Analysis of the proposed PEKS scheme

5.1. Correctness

Theorem 1. *The proposed PEKS scheme is complete.*

Proof. We are assuming that the ciphertext C is valid for key word w_i and trapdoor T is valid for w_j. The correctness of the proposed **Test** algorithm can be verified as

$$\begin{aligned}\phi &= \frac{e(g^{d_c d_o r_i}, H_2(w_j) \cdot H_1(g^{d_o d_c r_j}))}{e(g^{d_c d_o r_i}, H_1(g^{d_o d_c r_j}))} \\ &= \frac{e(g^{d_c d_o r_i}, H_2(w_j)) \cdot e(g^{d_c d_o r_i}, H_1(g^{d_o d_c r_j}))}{e(g^{d_c d_o r_i}, H_1(g^{d_o d_c r_j}))} \\ &= e(g^{d_c d_o r_i}, H_2(w_j)) \end{aligned} \quad (1)$$

Data Owner(DO)	Cloud Server(CS)	Data Consumer(DC)
$\mathbf{PEKS}(\mathbf{param}, \mathbf{PK}_C, \mathbf{SK}_O, \mathcal{W})$ Choose $\{w_1, w_2, w_3 \cdots w_n\}$ Choose $r_i \in \mathbb{Z}_p^*$ Compute $C_1 = (PK_C)^{d_o r_i} = g^{d_o d_c r_i}$ $C_2 = (PK_O)^{r_i} = g^{d_o r_i}$ $C_{w_i} = H_3(e(C_1, H_2(w_i)))$ $\xrightarrow{\langle C = [C_1, C_2, C_3, C_{w_1} \cdots C_{w_n}] \rangle}$ **(Public Channel)**		
	Store $C = [C_1, C_2, C_3, C_{w_1} \cdots C_{w_n}]$	
		$\mathbf{Trapdoor}(\mathbf{param}, \mathbf{PK}_O, \mathbf{SK}_C, \mathbf{PK}_{CS}, \mathcal{W})$ Choose w_j, and $r_j \in \mathbb{Z}_p^*$ Compute $T_1 = H_1((PK_O)^{d_c r_j}) = H_1(g^{d_c d_o r_j})$ $T_2 = (PK_{CS})^{d_c r_j} = g^{d_c d_s r_j}$ $T_3 = g^{r_j}$ $T_{w_j} = (H_2(w_j) \cdot T_1) = (H_2(w_j) \cdot H_1(g^{d_c d_o r_j}))$ $T = (T_1, T_2, T_3, T_{w_j})$ $\xleftarrow{\langle T = [T_1, T_2, T_3, T_{w_j}] \rangle}$ **(Public Channel)**
	$\mathbf{Test}(\mathbf{param}, T_{w_j}, C_{w_i}, SK_{CS})$ Compute $\phi = e(C_1, T_{w_j})/e(C_1, T_1)$ **If**$(H_3(\phi) \cdot e(C_1, T_3)^{d_s} = C_{w_i} \cdot e(C_2, T_2))$ **Return** "YES" **Else** **Return** "NO"	

Fig. 4. The proposed PEKS scheme.

$$H_3(\phi) = H_3(e(g^{d_c d_o r_i}, H_2(w_j))) \\ = H_3(e(C_1, H_2(w_j))) \tag{2}$$

$$H_3(\phi) \cdot e(C_1, T_3)^{d_s} = H_3(e(C_1, H_2(w_j))) \cdot e(g^{d_c d_o r_i}, g^{r_j})^{d_s} \\ = H_3(e(C_1, H_2(w_j))) \cdot e(g,\ g)^{d_c d_s d_o r_i r_j} \tag{3}$$

$$C_{w_i} \cdot e(C_2, T_2) = H_3(e(C_1, H_2(w_i))) \cdot e(g^{d_o r_1}, g^{d_s d_c r_j}) \\ = H_3(e(C_1, H_2(w_i))) \cdot e(g, g)^{d_o d_c d_s r_i r_j} \tag{4}$$

From the Eqs. (3) and (4), we found that if $w_i = w_j$, then $H_3(\phi) \cdot e(C_1, T_3)^{d_s} = C_{w_i} \cdot e(C_2, T_2)$. Therefore, the proposed PEKS scheme is correct.

5.2. Trapdoor indistinguishability

We will now prove the security of a trapdoor under the hardness assumption of the HDH problem. Further, we summarize the trapdoor indistinguishability in Fig. 5.

Theorem 2. *Assume that $H_i(\cdot)$, $i = 1, 2, 3$ behaves like a random oracle. We define $\mathbf{Adv}_{\mathcal{A}_1, PEKS}^{T_{w_b}^* - ind}(t')$ is the probability of success of any PPT adversary $\mathcal{A}_1$ within polynomial time-bound t' to breach the trapdoor indistinguishability in the CPA model of the proposed PEKS scheme by executing at most q_{H_1}, q_C, q_T queries to hash oracle $\mathcal{O}_{H_1}$, ciphertext oracle $\mathcal{O}_C$ and trapdoor oracle $\mathcal{O}_T$, respectively. Therefore, we have:*

$$\mathbf{Adv}_{\mathcal{A}_1, PEKS}^{T_{w_b}^* - ind}(t') \leq \mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)$$

$$t' \leq t + \mathcal{O}\big((q_{H_1} + q_C + q_T) \cdot t_{exp}\big)$$

where $\mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)$ is the probability of success of $\mathcal{A}_1$ to break the security of HDH problem within a polynomial time-bound t, and t_{exp} signifies the time complexity to compute a modular exponentiation operation in $\mathbb{Z}_p^*$

Proof. To prove this theorem, we will follow the analysis given in [36]. As described in **Game 1**, we are assuming that there is a malicious outside attacker $\mathcal{A}_1$ who is trying to breach the trapdoor indistinguishability in the CPA model of the proposed PEKS scheme within a polynomial time bound t'. For this purpose, $\mathcal{A}_1$ is playing a challenge–response game with a challenger $\mathcal{C}$ who invokes a PPT algorithm $\mathcal{B}$ to simulate the proposed PEKS scheme based on various queries asked by $\mathcal{A}_1$.

- **Setup:** $\mathcal{C}$ invokes $\mathcal{B}$ and generates a HDH problem instance $\mathscr{C}_{HDH} = \{\mathbf{G}_1, \mathbf{G}_2, p, e, g, g^\alpha, g^\beta, g^\gamma, H_1(g^c), \zeta\}$. We assume that H_i, $i = 1, 2, 3$ behaves like a random oracle, and ζ is either equal to $H_1(g^{\alpha\beta})$ or a random elements $H_1(g^c) \in \mathbf{G}_1$. $\mathcal{C}$ chooses $\alpha, \beta, \gamma \in \mathbf{G}_1$, and sets $\mathbf{SK}_C = \alpha$, $\mathbf{SK}_O = \beta$ and $\mathbf{SK}_C = \gamma$ as the secret keys of **DC**, **DO**, and **CS**, respectively. $\mathcal{C}$ also computes the corresponding the public keys of **DC**, **DO** and **CS** as $\mathbf{PK}_C = g^\alpha$, $\mathbf{PK}_O = g^\beta$, and $\mathbf{PK}_{CS} = g^\gamma$, respectively. For simplicity, we made the following assumptions:
    1. $\mathcal{A}_1$ can make at most q_{H_1}, q_C, q_T queries to hash oracle $\mathcal{O}_{H_1}$, ciphertext oracle $\mathcal{O}_C$, and trapdoor oracle $\mathcal{O}_T$, respectively.
    2. $\mathcal{A}_1$ can issue only one query for a keyword w to an oracle, i.e., it cannot repeat a query to the oracle.
    3. $\mathcal{A}_1$ first issues query for a keyword w to $\mathcal{O}_{H_1}$, then it will issue query for a keyword w to $\mathcal{O}_C$ or $\mathcal{O}_T$.
- **Trapdoor queries:** Suppose $\mathcal{A}_1$ issues a trapdoor query for a chosen keyword w_k, and $\mathcal{C}$ will respond in the following way
    1. $\mathcal{C}$ chooses $a_k \in \mathbb{Z}_p^*$ uniformly at random, and calculates $T_1^* = H_1(g^{\beta\alpha})^{a_k}$, $T_2^* = g^{\gamma \times a_k}$ $T_3^* = g^{a_k}$ and $T_{w_k}^* = (H_2(w_k) \cdot T_1^*)$.
    2. $\mathcal{C}$ responds to $\mathcal{A}_1$ with the trapdoor $T^* = [T_1^*, T_2^*, T_3^*, T_{w_k}^*]$ for the keyword w_k.
- **Challenge:** Let $\mathcal{A}_1$ wish to be challenged on the keywords w_0^* and w_1^*. $\mathcal{C}$ produced the challenge trapdoor $T = [T_1, T_2, T_3, T_{w_b}]$ as follows.
    1. $\mathcal{C}$ chooses a bit $b \in \{0, 1\}$ uniformly at random, and set $T_1 = \zeta^b$, $T_2 = g^{\gamma \times b}$, $T_3 = g^b$ and $T_{w_b} = (H_2(w_b) \cdot T_1)$, where ζ is the component of HDH problem instance.

$\mathscr{C}_{HDH}$	$\mathcal{C}$	$\mathcal{A}_1$
$\mathbf{param} = \{\mathbf{G}_1, \mathbf{G}_2, p, e, g, g^{\alpha}g^{\beta}, g^{\gamma}, H_1(g^c), \zeta\}$ $\xrightarrow{\langle \mathbf{param} \rangle}$		
	Select $\alpha, \beta, \gamma \in \mathbf{G}_1$ Set $\mathbf{SK}_C = \alpha$ $\mathbf{PK}_C = g^{\alpha}$ $\mathbf{PK}_O = g^{\beta}$ $\mathbf{PK}_{CS} = g^{\gamma}$ $\xrightarrow{\langle g^{\alpha}, g^{\beta}, g^{\gamma} \rangle}$	
		Choose $\mathbf{PK}_O, \mathbf{PK}_{CS}, \mathbf{PK}_C$ and w_k for $\mathcal{O}_T$ and $\mathcal{O}_C$ $\xleftarrow{\langle w_k \rangle}$
	Choose $a_k \in \mathbb{Z}_p^*$ Set $T_1^* = H_1(g^{\beta\alpha})^{a_k}$ $T_2^* = g^{\gamma \times a_k}$ $T_3^* = g^{a_k}$ $T_{w_k}^* = (H_2(w_k) \cdot T_1^*)$ $\xrightarrow{\langle T^* = [T_1^*, T_2^*, T_3^*, T_{w_k}^*] \rangle}$	
		$\xleftarrow{\langle w_0^*, w_1^* \rangle}$
	Choose $b \in \{0,1\}$ Set $T_1 = \zeta^b$ $T_2 = g^{\gamma \times b}$ $T_3 = g^b$ $T_{w_b} = (H_2(w_b) \cdot T_1)$ $\xrightarrow{\langle T = [T_1, T_2, T_3, T_{w_b}] \rangle}$ If $\zeta = H_1(g^{\alpha\beta})$ T is a valid trapdoor	
		Choose $(\mathbf{PK}_O, \mathbf{PK}_{CS}$ and $w_j) \neq$ $(g^{\beta}, g^{\gamma}, w_0^*)$ $(g^{\beta}, g^{\gamma}, w_1^*)$ for $\mathcal{O}_T$ Choose$(\mathbf{PK}_O, \mathbf{PK}_C$ and $w_j) \neq$ $(g^{\beta}, g^{\alpha}, w_0^*)$ $(g^{\beta}, g^{\alpha}, w_1^*)$ for $\mathcal{O}_C$ $\xleftarrow{\langle w_j \rangle}$
	$\xrightarrow{\langle C_j, T_j \rangle}$	
		Select $b' \in \{0,1\}$ $\xleftarrow{b'}$
	If $(b' = b)$ $\mathcal{A}_1$ win's the game Output $\zeta = H_1(g^{\alpha\beta})$ Otherwise Output $\zeta \neq H_1(g^{\alpha\beta})$	

Fig. 5. Trapdoor indistinguishability.

2. C sends the challenge trapdoor $T = [T_1, T_2, T_3, T_{w_b}]$ to $\mathcal{A}_1$.

- **Output:** If $\zeta = H_1(g^{\alpha\beta})$, then T is valid trapdoor for w_b under the randomness of b. If $\mathcal{A}_1$ issues the trapdoor queries for another keyword w_j, where $w_j \neq w_0^*$, $w_j \neq w_1^*$, then $\mathcal{A}_1$ produces its output $b' \in \{0,1\}$. Here b' is indicating that whether the challenge trapdoor T is for $\mathbf{Trapdoor}(\mathbf{pram}, \mathbf{SK}_C, \mathbf{PK}_O, \mathbf{PK}_{CS}, w_0)$ or $\mathbf{Trapdoor}(\mathbf{pram}, \mathbf{SK}_C, \mathbf{PK}_O, \mathbf{PK}_{CS}, w_1)$. If $b = b'$, C outputs 1, i.e., $\zeta = H_1(g^{\alpha\beta})$, otherwise outputs 0, i.e., $\zeta \neq H_1(g^{\alpha\beta})$. When the input tuples are taken from $\mathscr{C}_{HDH}$, i.e., $\zeta = H(g^{\alpha\beta})$, then in this case $\mathcal{A}_1$ must satisfy $|\mathbf{Pr}[b = b'] - \frac{1}{2}| > \mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)$. On the other hand, if the input tuples are taken from $\mathscr{C}_{HDH}$, i.e., ζ is chosen uniformly at random over $\mathbf{G}_1$, then $T_{w_i} = (H_2(w_k) \cdot \zeta)$ is an independent element of $\mathbf{G}_1$. In this case, $\mathbf{Pr}[b = b'] = \frac{1}{2}$ since (g, g^a, g^b) and ζ are independent in $\mathbf{G}_1$. Thus, we can write

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}_1,PEKS}^{T_{w_b}^*-ind}(t') &= |\mathbf{Pr}[\mathcal{B}(g, g^{\alpha}, g^{\beta}, H(g^{\alpha\beta})) = 1] - \mathbf{Pr}[\mathcal{B}(g, g^{\alpha}, g^{\beta}, \zeta) = 1]| \\ &\leqslant |\Big(\frac{1}{2} \pm \mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)\Big) - \frac{1}{2}| \\ &\leqslant \mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)\end{aligned}$$

We can say, $\mathcal{A}_1$ can break trapdoor indistinguishability of the proposed scheme in the CPA model if

$$\mathbf{Adv}_{\mathcal{A}_1,PEKS}^{T_{w_b}^*-ind}(t') \leq \mathbf{Adv}_{\mathcal{A}_1}^{HDH}(t)$$

$$t' \leq t + \mathcal{O}\big((q_{H_1} + q_C + q_T) \cdot t_{exp}\big)$$

5.3. Ciphertext indistinguishability

We will prove ciphertext indistinguishability against the CPA model of the proposed PEKS scheme under the DBDH assumptions. The security analysis followed in this paper is similar to the method proposed in [3]. The ciphertext indistinguishability analysis is further described in Fig. 6.

Theorem 3. *Assume that $H_i(\cdot)$, $i = 1, 2, 3$ behaves like a random oracle. We define $\mathbf{Adv}_{\mathcal{A}_2,PEKS}^{C_{w_b}^*-ind}(t')$ is the probability of success of any PPT adversary $\mathcal{A}_2$ within polynomial time-bound t' to breach the ciphertext indistinguishability against CPA of the proposed PEKS scheme by executing*

$\mathscr{C}_{DBDH}$	$\mathcal{C}$	$\mathcal{A}_2$
$\mathbf{param} = \{\mathbf{G}_1, \mathbf{G}_2, p, e, g, g^x, g^y, g^z, Z\}$ $\xrightarrow{\langle \mathbf{param} \rangle}$		
	$\mathbf{PK}_C = g^y$ $\mathbf{PK}_O = g^x$ $\mathbf{PK}_{CS} = g^z$ $\xrightarrow{\langle g^x, g^y, g^z \rangle}$	
		Choose $\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}$ and w_i for $\mathcal{O}_T$ and $\mathcal{O}_C$ $\xleftarrow{\langle \mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_i \rangle}$
	Choose $a_i \in \mathbb{Z}_p^*$ Compute h_i $\xrightarrow{\langle C_i, T_i \rangle}$	
		$\xleftarrow{\langle w_0, w_1 \rangle}$
	Choose $b \in \{0, 1\}$ Compute a ciphertext $C^*_{w_b}$ $\xrightarrow{\langle C^*_{w_b} \rangle}$	
		Choose $(\mathbf{PK}_O, \mathbf{PK}_{CS}$ and $w_k) \neq (g^x, w_0)\ (g^x, w_1)$ for $\mathcal{O}_T$ Choose $(\mathbf{PK}_C, \mathbf{PK}_O$ and $w_k) \neq (g^y, w_0)\ (g^y, w_1)$ for $\mathcal{O}_C$ $\xleftarrow{\langle w_k \rangle}$
	$\xrightarrow{\langle C_{w_k}, T_{w_k} \rangle}$	
		Select $b' \in \{0, 1\}$ $\xleftarrow{b'}$
	If $(b' = b)$ $\mathcal{A}_2$ win's the game Output $Z = e(g, g)^{xyz}$ Otherwise Output $Z \neq e(g, g)^{xyz}$	

Fig. 6. Ciphertext indistinguishability.

at most q_{H_2}, q_{H_3}, q_C, q_T queries to hash oracles $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$, ciphertext oracle $\mathcal{O}_C$, and trapdoor oracle $\mathcal{O}_T$, respectively. Therefore, we have:

$$\mathbf{Adv}_{\mathcal{A}_2, PEKS}^{C^*_{w_b}-ind}(t') \leq \frac{1}{e} \cdot \frac{1}{q_T} \cdot \frac{1}{q_{H_3}} \cdot \mathbf{Adv}_{\mathcal{A}_1}^{DBDH}(t)$$

$$t' \leq t + \mathcal{O}\big((q_{H_3} + q_C + q_T) \cdot t_{exp}\big)$$

where $\mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t)$ is the probability of success of $\mathcal{A}_2$ to break the security of DBDH problem within a polynomial time-bound t, and t_{exp} signifies the time complexity to compute a modular exponentiation operation in $\mathbb{Z}_p^*$.

Proof. We assuming that $\mathcal{A}_2$ is trying to breach the ciphertext indistinguishability against the CPA model of the proposed PEKS scheme within a polynomial time bound t'. For this purpose, $\mathcal{A}_2$ is playing a challenge–response game with a challenger $\mathcal{C}$. $\mathcal{C}$ invokes a polynomial time-bounded algorithm $\mathcal{B}$ to generates a DBDH instance $\mathscr{C}_{DBDH}$. We also assume that $\mathcal{A}_2$ is executing at most q_{H_2}, q_{H_3}, q_C, q_T queries to hash oracles $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$, ciphertext oracle $\mathcal{O}_C$, and trapdoor oracle $\mathcal{O}_T$, respectively to simulate the proposed PEKS scheme.

- **Setup:** $\mathcal{B}$ takes a DBDH problem instance $\mathscr{C}_{DBDH} = \{\mathbf{G}_1, \mathbf{G}_2, e, p, g, g^x, g^y, g^z, Z\}$, where $x, y, z \xleftarrow{R} \mathbb{Z}_p^*$, and Z is either equal to $e(g, g)^{xyz}$ or a random element of $\mathbf{G}_2$. $\mathcal{B}$ chooses a bit b such that $b = 1$ if $Z = e(g, g)^{xyz}$ or $b = 0$ if Z is random element. $\mathcal{C}$ sets $\mathbf{param} = (\mathbf{G}_1, \mathbf{G}_2, e, p, g)$ and $u_1 = \mathbf{PK}_O = g^x, u_2 = \mathbf{PK}_C = g^y, u_3 = \mathbf{PK}_{CS} = g^z$, so we can say that $\mathbf{SK}_O = x$, $\mathbf{SK}_C = y$, $\mathbf{SK}_{CS} = z$. Now, $\mathcal{C}$ executing $\mathcal{B}$ with the input $(\mathbf{param}, \mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS})$, and responding based on different queries made by $\mathcal{A}_2$. For simplicity, we made the following assumptions:
  1. $\mathcal{A}_2$ can make at most q_{H_2}, q_{H_3}, q_C, q_T queries to hash oracles $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$, ciphertext oracle $\mathcal{O}_C$, and trapdoor oracle $\mathcal{O}_T$, respectively.
  2. $\mathcal{A}_2$ can issue only one query for a keyword w to an oracle, i.e., it cannot repeat a query to the oracle.
  3. $\mathcal{A}_2$ first issues query for a keyword w to $\mathcal{O}_{H_2}$, then it will issue query for a keyword w to $\mathcal{O}_C$ or $\mathcal{O}_T$.

  Now $\mathcal{C}$ simulates the oracles as follows:
- H_2**-queries:** $\mathcal{A}_2$ can query the oracle H_2 at any time. To respond to this query, $\mathcal{C}$ maintains a list L_{H_2}. Initially, L_{H_2} is empty. L_{H_2} includes the tuples like $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_i), h_i, a_i, c_i\rangle$. When $\mathcal{A}_2$ starts query to H_2 for the input w_i, $\mathcal{C}$ responds in the following way:
  - If w_i is present in L_{H_2} in a tuple $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_i), h_i, a_i, c_i\rangle$, $\mathcal{C}$ responds with h_i.
  - Otherwise, $\mathcal{C}$ selects a coin $c_i \in \{0, 1\}$ uniformly at random with $\mathbf{Pr}[c_i = 0] = \frac{1}{(q_T+1)}$.
  - $\mathcal{C}$ chooses $a_i \in \mathbb{Z}_p^*$ uniformly at random, and computes
    - $h_i \leftarrow g^{a_i} \cdot g^z \in \mathbf{G}_1$ if $c_i = 0$, or
    - $h_i \leftarrow g^{a_i} \in \mathbf{G}_1$ if $c_i = 1$.
  - Now, $\mathcal{C}$ adds the tuple $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_i), h_i, a_i, c_i\rangle$ to L_{H_2}, and responds to $\mathcal{A}_2$ by setting $H_2(w_i) = h_i$. Here h_i is selected uniformly at random from $\mathbf{G}_1$, and independent of $\mathcal{A}_2$.
- H_3**-queries:** To responds to H_3-queries, $\mathcal{C}$ maintains an initial-empty list L_{H_3}. If there exists a $t \in \mathbf{G}_2$, $\mathcal{C}$ responds to query of $H_3(t)$ by choosing a random value $V \in \{0, 1\}^\eta$ for each t and sets $H_3(t) = V$, and adds the pair (t, V) to L_{H_3}.
- **Trapdoor queries:** When $\mathcal{A}_2$ issues a trapdoor query for a keyword w_i, $\mathcal{C}$ responds as follows:

– Assume that a tuple $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_j), h_j, a_j, c_j\rangle$ is found in L_{H_2}. If $w_i = w_j$, C selects $h_j \in \mathbf{G}_1$, sets $H_2(w_i) = H_2(w_j) = h_j$.
  * If $c_i = 1$ and $h_i = g^{a_i}$, C sets $T_1 = H_1(u_1^{ya_j})$, $T_2 = u_3^{ya_j}$, $T_3 = g^{a_j}$ $T_{w_j} = h_j \cdot T_1$.
  * If $c_i = 0$, then C reports failure and terminates.

– Here we can see that $T_{w_j} = h_j \cdot T_1 = H_2(w_j) \cdot T_1$. Hence $T = (T_1, T_2, T_3, T_{w_j})$ is a valid trapdoor for w_j, then C returns $T = (T_1, T_2, T_3, T_{w_j})$ to $\mathcal{A}_2$.

• **Challenge:** $\mathcal{A}_2$ selects two keywords w_0, w_1, and challenged on them. C does as follows:
  – To get $h_0, h_1 \in \mathbf{G}_1$, i.e., $H_2(w_0) = h_0$ and $H_2(w_1) = h_1$, C executes H_2 queries. For $i = 0, 1$, let $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_j), h_j, a_j, c_j\rangle$ be the corresponding tuples on L_{H_2}. C reports failure and terminates if $c_0 = 1$ and $c_1 = 1$.
  – We know that at least one of c_0, c_1 would be equal to 0. C chooses $b \in \{0, 1\}$ uniformly at random such that $c_b = 0$. Note that no randomness is needed if $c_b = 0$ because there is only one choice.
  – C generates the **PEKS** $= [u_2{}^{xa_b}, u_1{}^{a_b}, J]$, where $J \in \{1, 0\}^\eta$ is chosen uniformly at random. The challenge J is defined as $H_3(e(u_2{}^{xa_b}, H_2(w_b))) = J$, where

$$J = H_3(e(u_2{}^{xa_b}, g^{a_b} g^z)) = H_3(e(g^{yxa_b}, g^{a_b+z}))$$
$$= H_3(e(g, g)^{xya_b(a_b+z)})$$

  With this definition, we can say that C is a valid PEKS ciphertext for w_b.

As of now $\mathcal{A}_2$ made queries for the keywords w_0 and w_1. $\mathcal{A}_2$ can continue to ask trapdoor queries for keywords w_i provided $w_i \neq w_0$, $w_i \neq w_1$, and C responds to these queries as described before.

• **Output:** Finally, $\mathcal{A}_2$ produces the output according to its guess bit $b' \in \{0, 1\}$ which is indicating that whether the challenge ciphertext C is the result for w_0 or w_1. At this point, C chooses a pair (t, V) uniformly at random from L_{H_2}. Now the pair (t, V) would be $t/e(u_1^{a_b}, u_2^{a_b})$ as it is guessing for $e(g, g)^{xyz}$, where a_b is the same value used in the challenge step. There is a reason behind this work, as we will show, $\mathcal{A}_2$ must have issued a query for either $H_3(e(u_2{}^{xa_b}, H_2(w_0)))$ or $H_3(e(u_2{}^{xa_b}, H_2(w_1)))$. Hence, in L_{H_2} if there exists a pair whose left hand side is equal to $t = e(u_2{}^{xa_b}, H_2(w_b)) = e(g, g)^{xya_b(a_b+z)}$ with the probability of $\frac{1}{2}$. If C choose the pair (t, V) from L_{H_2} then $t/e(u_1^{a_b}, u_2^{a_b}) = e(g, g)^{xyz}$ holds. We define the following three events:

  – ξ_1: C does not abort any trapdoor query.
  – ξ_2: C does not abort during challenge phase.
  – ξ_3: In the real attack, $\mathcal{A}_2$ issues an H_3 query for either one of $H_3(e(u_2{}^{xa_b}, H_2(w_0)))$ or $H_3(e(u_2{}^{xa_b}, H_2(w_1)))$.

We now follow the same steps as in [3,37] to show that ξ_1 and ξ_2 occur with sufficiently high probability.

  – We now claim that $\mathbf{Pr}[\xi_1] \geq \frac{1}{e}$. Without the loss of the generality, we are assuming that $\mathcal{A}_2$ does not query for the same keyword twice. C can issue a trapdoor query is at most q_T. So the probability that a trapdoor query asked by C to abort is $\frac{1}{(q_T+1)}$, and thus $\Pr[\neg\xi_1] \geq \frac{1}{(q_T+1)}$. We assume that T_i is ith query made by $\mathcal{A}_2$, and the corresponding tuples in L_{H_2} is $\langle(\mathbf{PK}_O, \mathbf{PK}_C, \mathbf{PK}_{CS}, w_i), h_i, a_i, c_i\rangle$. Since the only value that could be given to $\mathcal{A}_2$ is $H_2(w_i)$ which depends on c_i. Since $\mathcal{A}_2$ can make maximum q_T trapdoor quarries, so the probability that C does not abort as a result of all trapdoor queries is at least $(1 - \frac{1}{(q_T+1)})^{q_T} \geq \frac{1}{e}$ [3,37].
  – We now claim that $\mathbf{Pr}[\xi_2] \geq \frac{1}{q_T}$. If $\mathcal{A}_2$ is able to issue the trapdoor queries for w_0, w_1 with $c_0 = c_1 = 1$ where $\langle(\mathbf{PK}_{CS}, w_0), h_0, a_0, c_0\rangle$ and $\langle(\mathbf{PK}_{CS}, w_1), h_1, a_1, c_1\rangle$ are the tuples of L_{H_2}, then C will abort this game during the challenge phase. Since $\mathcal{A}_2$ is yet to issue trapdoor queries for w_0, w_1, so both c_0, and c_1 are independent of $\mathcal{A}_2$'s current view. Since $\mathbf{Pr}[c_i = 0] = \frac{1}{(q_T+1)}$ for $i = 0, 1$, we can say that $\mathbf{Pr}[c_0 = 1 \wedge c_1 = 1] = \left(1 - \frac{1}{(q_T+1)}\right)^2 \geq \frac{1}{q_T}$ as both c_0 and c_1 are independent of each other. Hence, $\mathbf{Pr}[\xi_2] \geq \frac{1}{q_T}$.
  – Note that $\mathcal{A}_2$ can never issue a trapdoor query for challenge keywords w_0, and w_1, we can say that $\mathbf{Pr}[\xi_1 \wedge \xi_2] \geqslant \frac{1}{e} \cdot \frac{1}{q_T}$ as both the events ξ_1 and ξ_2 are independent.
  – We now claim that $\mathbf{Pr}[\xi_3] \geqslant \frac{1}{q_T}$. Assume that in real attack, $\mathcal{A}_2$ is given the public key $[u_1, u_2]$ and $\mathcal{A}_2$ challenges on the keywords w_0 and w_1. In response, $\mathcal{A}_2$ is given a challenged ciphertext $C = [u_2{}^{xa_b}, u_1{}^{a_b}, J]$. When ξ_3 occurs, we know that the bit $b \in \{0, 1\}$ indicates that whether C is **PEKS** of w_0 or w_1, and it is independents of $\mathcal{A}_2$'s view. So, the output b' of $\mathcal{A}_2$ satisfies $b = b'$ and the maximum probability is $\frac{1}{2}$. We know that in the real attack, $|\mathbf{Pr}[b = b'] - \frac{1}{2}| \geqslant \mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t)$.

$$\begin{aligned}
\mathbf{Pr}[b = b'] &= \mathbf{Pr}[b = b' \wedge \xi_3] + \mathbf{Pr}[b = b' \wedge \neg\xi_3] \\
&= \mathbf{Pr}[b = b'|\xi_3]\mathbf{Pr}[\xi_3] + \mathbf{Pr}[b = b'|\neg\xi_3]\mathbf{Pr}[\neg\xi_3] \\
&\leqslant \mathbf{Pr}[b = b'|\xi_3]\mathbf{Pr}[\xi_3] + \mathbf{Pr}[\neg\xi_3] \\
&= \frac{1}{2}\mathbf{Pr}[\xi_3] + \mathbf{Pr}[\neg\xi_3] \\
&= \frac{1}{2}(\mathbf{Pr}[\xi_3] + 2\mathbf{Pr}[\neg\xi_3]) \\
&= \frac{1}{2}(1 - \mathbf{Pr}[\neg\xi_3] + 2\mathbf{Pr}[\neg\xi_3]) \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Pr}[\neg\xi_3]
\end{aligned}$$

  It shows that $\mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t) \leqslant |\mathbf{Pr}[b = b'] - \frac{1}{2}| \leqslant \frac{1}{2}\mathbf{Pr}[\neg\xi_3]$. Therefore, we have: $\mathbf{Pr}[\neg\xi_3] \geqslant 2 \cdot \mathbf{Adv}_{\mathcal{A}_1}^{DBDH}(t)$.

Now we assume that C does not aborts, and simulates a real attack game perfectly till the moment when $\mathcal{A}_2$ issues a query for either $H_3(e(u_2{}^{xa_b}, H_2(w_0)))$ or $H_3(e(u_2{}^{xa_b}, H_2(w_1)))$. Hence, $\mathcal{A}_2$ will be issuing a query for either $H_3(e(u_2{}^{xa_b}, H_2(w_0)))$ or $H_3(e(u_2{}^{xa_b}, H_2(w_1)))$ by the end of the simulation with a probability of at least $2 \cdot \mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t)$. So $\mathcal{A}_2$ issues query for $H_3(e(u_2{}^{xa_b}, H_2(w_b)))$ with a probability equal to $\mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t)$. C will choose the right pair with probability at least $\frac{1}{q_{H_3}}$. Therefore, the probability of correct answer would be $\frac{1}{q_{H_3}} \cdot \mathbf{Adv}_{\mathcal{A}_1}^{DBDH}(t)$. Since, C does not abort the simulation of any trapdoor query with a probability of $\frac{1}{e} \cdot \frac{1}{q_T}$, and thus, we have

$$\mathbf{Adv}_{\mathcal{A}_2, PEKS}^{C^*_{w_b}-ind}(t') \leq \frac{1}{e} \cdot \frac{1}{q_T} \cdot \frac{1}{q_{H_3}} \cdot \mathbf{Adv}_{\mathcal{A}_2}^{DBDH}(t)$$

$$t' \leq t + \mathcal{O}\big((q_{H_3} + q_C + q_T) \cdot t_{exp}\big)$$

6. Testbed experiments and results

In this section, we implement a real test-bed experiment for the proposed scheme using PBC library [38], and C socket programming. Each Data Owner (**DO**), Cloud Server (**CS**), and Data Consumer (**DC**) was executed on a machine (shown in Fig. 7) with 12th Gen Intel(R) Core(TM) i5-12400, 16 GB RAM and 512 GB SSD as secondary storage. Each machine described here queryUbuntu server 22.04 LTS and is connected to the same LAN.

The devices were accessed using the "Secure Shell or Secure Socket Shell (SSH)" from another device connected to the same network, and the execution results are shown in Fig. 8. The terminal window on the left side shows the logs from **DO**, the center window shows the logs

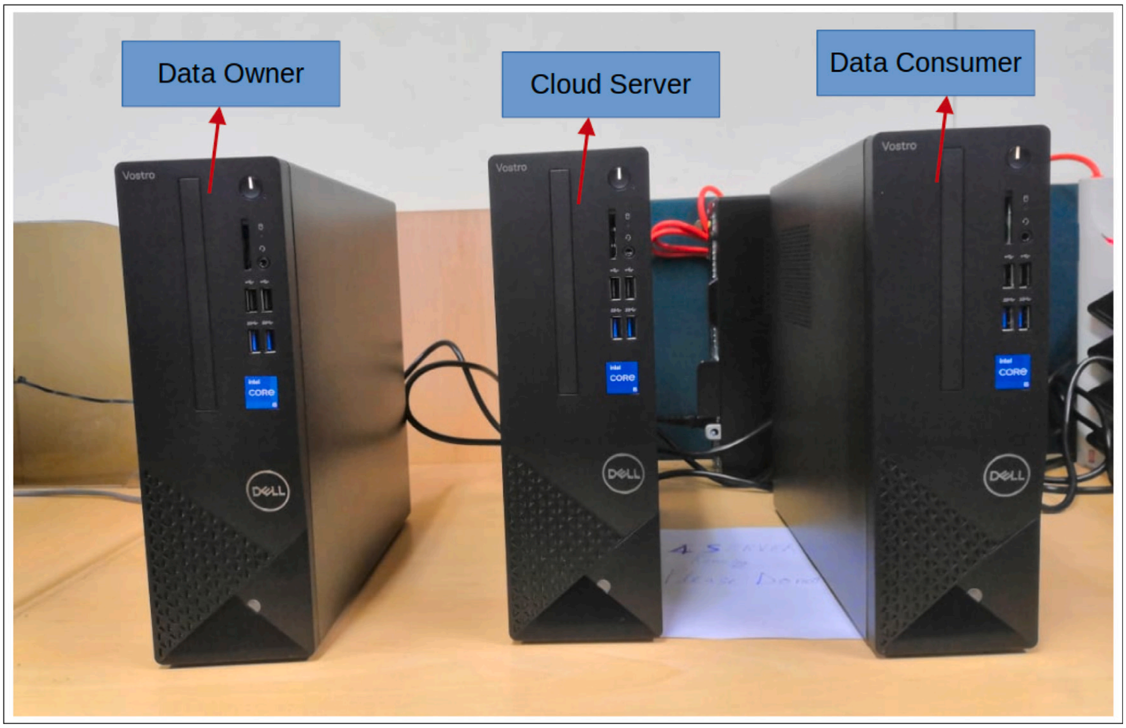

Fig. 7. Experimental setup.

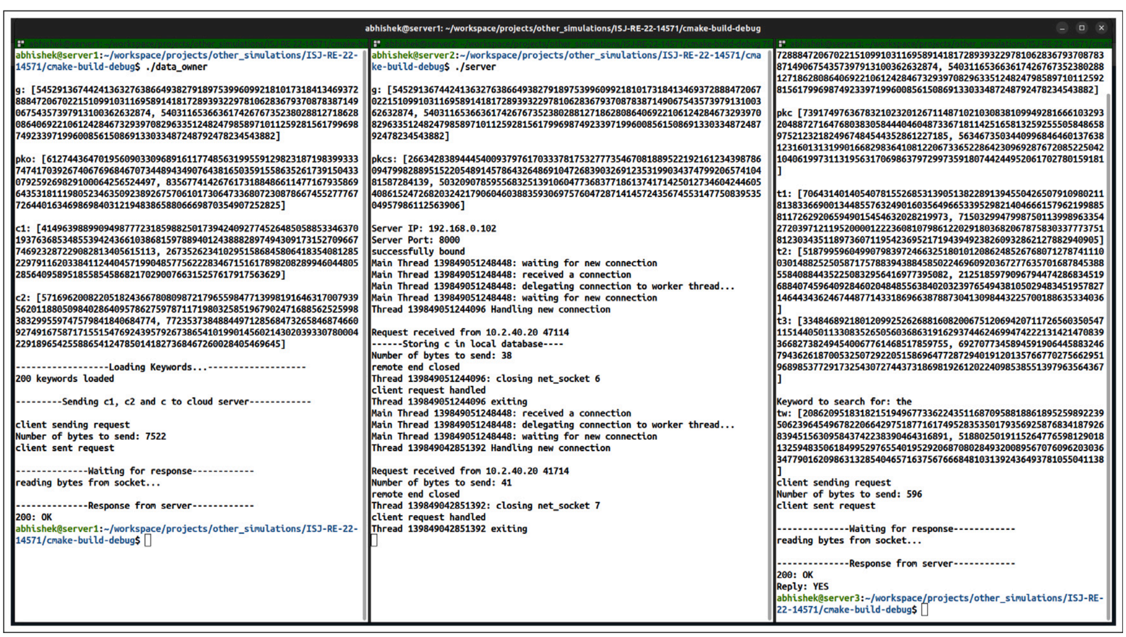

Fig. 8. Testbed experimental results for **DO**, **CS**, and **DC**.

from **CS**, and the right window shows the logs from **DC**. The **CS** starts first, followed by the **DO**, which loads keywords and encrypts them to send them to **DC**, which in turn stores the keywords in a local database. The **DC** then requests **CS** to search for a specific keyword (in encrypted form). The **CS** responds with a 'YES' or 'NO' type of answer depending on whether a keyword is present in its local database of encrypted entries or not. It is assumed that each entity has the required public keys.

A description of each of the three entities is given below.

- **Data Owner (DO)**: It loads n unique keywords from a file containing many keywords. It uses the PBC library to compute C_1, C_2, and $C_{w_i} \forall i \in \{1, 2, 3, \dots, n\}$ as described in the scheme. The **DO** then creates a socket, connects with the server socket listening on the **CS**, and transfers the data computed in the previous step via the socket as per the scheme.
- **Cloud Server (CS):** It creates a server socket and listens for connection requests from **DO** and **DC**. On receiving the data from **DO**, **CS** saves it in a local database (which is just a simple file in this case). On receiving a search request from **DC**, **CS** traverses the local database to check if the keyword is present in the database

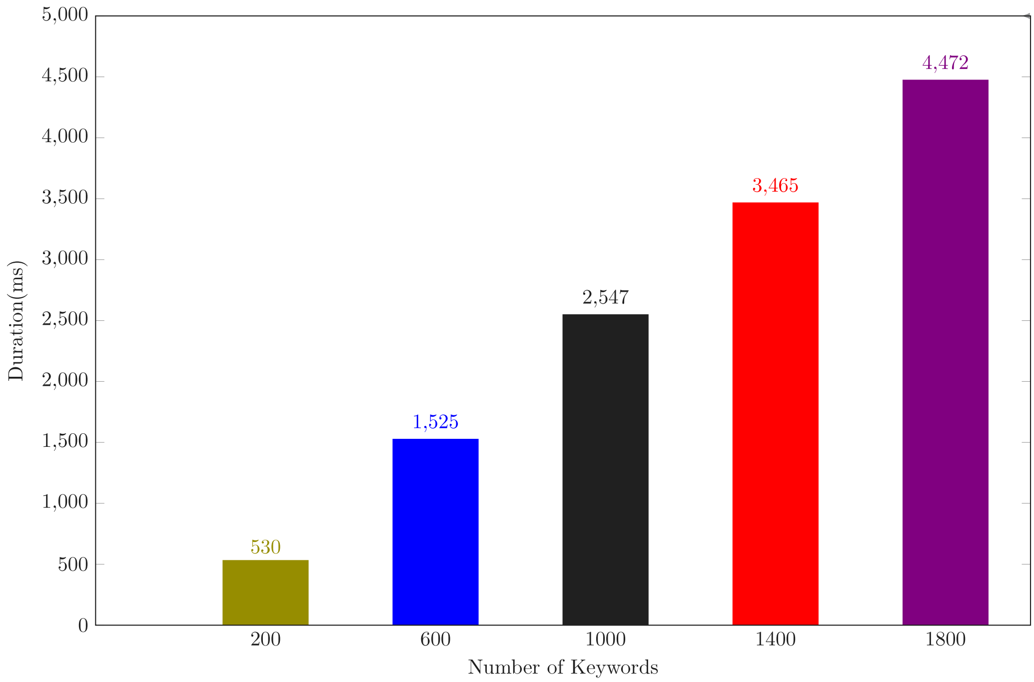

Fig. 9. Testbed experimental results for data owner, cloud server, and data consumer.

or not. It presents its response with a 'YES'; otherwise, it responds with a 'NO''.

- **Data Consumer (DC)**: When **DO** has uploaded the data to **CS**, **DC** can send a search request. It takes a keyword w as input from the user and generates T_1, T_2, T_3, T_w according to the proposed scheme. It creates a socket, connects it to **CS**, and sends the information generated in the previous step to **CS**. **CS** finally waits for a response and displays it to the user upon receiving it.

We also calculated the time taken to generate encrypted keywords while varying the number of keywords, and the results are shown in Fig. 9. We observed that, the keyword generation time increases linearly with the number of keywords.

7. Performance analysis

In this section, we compare our PEKS scheme with the state-of-the-art PEKS schemes.

7.1. Security attributes

Table 2 compares the security properties of the proposed scheme and other existing schemes. We have considered the PEKS schemes, which execute modular exponentiation operations in a multiplicative cyclic group of prime order. The proposed PEKS scheme provides ciphertext indistinguishability. Our scheme encrypts the data and the list of keywords $\mathcal{W} = \{w_1, w_2, w_3, \ldots, w_n\}$. The PEKS encryption algorithm uses the secret key $\mathbf{SK_O} = d_o$ of **DO**, and the public keys $\mathbf{PK_C} = g^{d_c}$ and $\mathbf{PK_{CS}} = g^{d_s}$ of **DC**, and **CS**, respectively to compute the ciphertext $C = \{C_1, C_2, C_3, C_{w_1}, \ldots, C_{w_n}\}$. Assume that $\mathcal{A}_1$ collects C. In the encryption algorithm, we choose a random bit string r_i, unknown to $\mathcal{A}_1$. Also, C will change every time, even if the same keyword is encrypted, because r_i will be chosen uniformly at random every time. The probability of guessing r_i is $\frac{1}{2^\lambda}$. However, $\mathcal{A}_1$ will not be able to compute $H_2(w_i)$ from C_{w_i} because r_i is unknown, and C_3 is secured because of the DBDH assumption. Hence, the proposed PEKS scheme provides ciphertext indistinguishability. In this regard, we point out that the schemes in [3,4] do not provide trapdoor security (See Table 2).

Our scheme also provides trapdoor indistinguishability. We assume that $\mathcal{A}_2$ got a valid trapdoor T, however, he/she cannot verify it. Because the verification of T involves the private key, d_s, of **CS**, and accordingly, only **CS** can verify it. Hence, the proposed scheme provides trapdoor indistinguishability. However, the scheme in [3] does not provide trapdoor indistinguishability [36] (See Table 2).

The proposed PEKS scheme can prevent the keyword-guessing attacks. We thwart this attack using the private key, d_o, of **DO**. Thus, the proposed scheme ensured that **CS** could not encrypt any keyword. Instead, the encryption algorithm needed the private key of **DO**. Therefore, **CS** could not relaunch a keyword-guessing attack as it could not create a valid ciphertext without **DO**'s secret key. The schemes in [3,19,36,36,39] do not provide security against keyword-guessing attacks. In these schemes, **DO**'s private key is not used during encryption, and thus **CS** can perform the keyword-guessing attacks. However, the schemes in [22,40] provide security against keyword-guessing attacks since the scheme in [22] used authenticated encryption, and the scheme in [40] employed **DO**'s signature. If the signature does not match, **CS** could not run **Test** algorithm. The main difference between the proposed scheme and the scheme in [40] is that a dishonest cloud server could not create a valid ciphertext of some chosen keywords. In contrast, in the scheme [40], the cloud server could create a ciphertext but could not run the **Test** algorithm without **DO**'s signature.

Table 2 shows that our PEKS method satisfies the ciphertext indistinguishability (P1) and trapdoor indistinguishability (P2), and resists the keyword-guessing attack (P5). In addition, our PEKS scheme does not require a secure channel (P3) to deliver the trapdoor to **DC**. Furthermore, the proposed scheme involves **DC** as a designated entity to verify the correctness of the trapdoor (P4); this kind of PEKS scheme is called the designated tester PEKS (dPEKS) scheme.

7.2. Communication cost

We used five different security levels (see Table 3) recommended by the NIST for this experimental analysis. We calculated the communication cost for ciphertext and trapdoor computations. In this section, we computed the communication cost of the proposed PEKS scheme, and compared it with the related existing schemes. The comparison is shown in Table 4. We have taken two multiplicative groups $\mathbf{G}_1$ and $\mathbf{G}_2$,

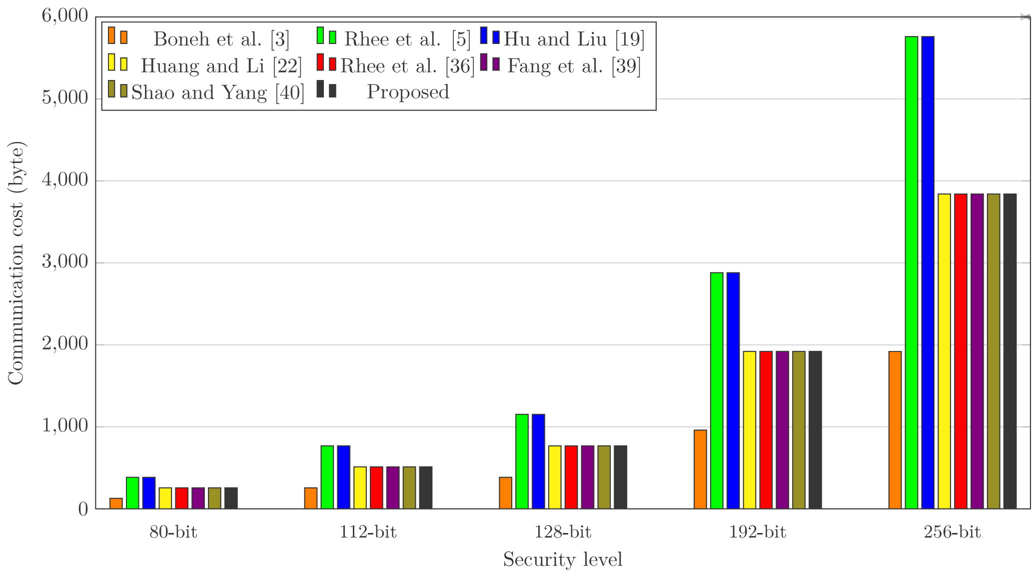

Fig. 10. Communication overheads for public key of different protocols.

Table 2
Comparison of security properties.

Scheme	P1	P2	P3	P4	P5
Boneh et al. [3]	Yes	No	No	No	No
Rhee et al. [5]	Yes	Yes	Yes	Yes	No
Hu and Liu [19]	Yes	Yes	Yes	Yes	No
Huang and Li [22]	Yes	Yes	Yes	No	Yes
Rhee et al. [36]	Yes	Yes	Yes	Yes	No
Fang et al. [39]	Yes	Yes	Yes	Yes	No
Shao and Yang [40]	Yes	Yes	Yes	Yes	Yes
Proposed	Yes	Yes	Yes	Yes	Yes

P1: Provides ciphertext indistinguishability; P2: Provides trapdoor indistinguishability; P3: Secure channel free; P4: dPEKS scheme; P5: Prevent keyword-guessing attacks.

Table 3
Security level versus group element size, digest length, length of p in bits.

Security level (bit)	$\|P\|$, $P \in G_1$ (bit)	$\|g\|$, $g \in G_2$ (bit)	$\|H\|$ (bit)	$\|p\|$, $p \in \mathbb{Z}_p$ (bit)
80	1024	2048	160	160
112	2048	4096	224	224
128	3072	6144	256	256
192	7680	15 360	384	384
256	15 360	30 720	512	512

the size of an element of $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbb{Z}_p$ are |p| bits. Here n signifies the number of keywords. We compared the communication cost regarding the size of the public key, ciphertext, and trapdoor. Fig. 10 shows that the communication cost of our scheme is less than or equal to that of most schemes in [5,19,22,36,39,40]. Only the communication cost of the scheme in [3] is less than our scheme. Fig. 11 shows the communication cost of the PEKS encryption algorithm, and it also shows that the communication cost of our scheme is greater than the schemes in [3,22,36], and less than or equal to the schemes in [5,19,39,40]. Fig. 12 shows the communication cost for the trapdoor generation. The communication cost of the proposed scheme is greater than the schemes in [3,22,36,39], and equal to the scheme in [5,19], and less than the scheme in [40]. However, our scheme satisfies all security and functionality requirements as listed in Table 2, whereas other schemes except [40] do not satisfy all these attributes. The communication cost of [40] is much greater than the proposed scheme.

7.3. Execution cost

To evaluate the efficiency of the state-of-the-art PEKS schemes, we implemented them by using the PBC library [38]. We have used Windows 11 system (64-bit OS) with intel(R) Core(TM) i3-1005G1 CPU @ 1.20 GHz 1.19 GHz and 8 GB RAM. All the operations have been executed 100 times, and a mean of these executions have been considered. The running time of various cryptographic operations is given in Table 5. We consider the following cryptographic operations, where T_E, T_P, T_H, and T_M to denote the average execution cost for one modular exponential operation in $\mathbf{G}_2$, one bilinear paring operation in $\mathbf{G}_1 \times \mathbf{G}_1$, one hash operation, and one modular multiplication, respectively. Table 6 provides the execution costs of various PEKS schemes for the following phases: (i) **PEKS**, (ii) **Trapdoor**, and (iii) **Test**. Here we used Type-1 paring. Fig. 13 shows the execution time of **PEKS** algorithm of the existing schemes. We vary the number of keywords (n) from 50 to 2000. Fig. 13 shows that the time needed to execute the **PEKS** algorithm. It is also showing that the running time of **PEKS** algorithm of the proposed scheme is less than the time needed in [3,5,22,36,39,40]. Our PEKS scheme and the scheme in [22] are secure from keyword-guessing attacks. Though the scheme in [40] satisfies all security parameters like the proposed scheme listed in Table 2, their execution time is much higher than ours, as shown in Fig. 10. Still, all other works listed in Table 2 are not secure from keyword-guessing attacks as the data owner's private key is not used to create ciphertext. So a dishonest server can relaunch keyword-guessing again and again. In Fig. 14, we plot the execution time of **Trapdoor** algorithm of various schemes. We observed that the proposed **Trapdoor** algorithm needs less time than all other schemes [3,5,19,22,36,39,40]. The main reason for having a low execution time is that we mostly use the hash function to generate a trapdoor. The execution time of the hash function is much less than that of bilinear pairing and modular exponentiation operations. Fig. 15 shows that the running time of the proposed **Test** algorithm is nearly equal to that of the scheme in [3,39], and less than the schemes in [5,19,22,36,40]. However, the schemes in [3,22] do not have a designated tester, i.e., anyone can run the **Test** algorithm wherein our scheme is only the desired tester-based method, i.e., the cloud server can run this algorithm. In our scheme, the trapdoor is not fixed, i.e., **Trapdoor** algorithm will generate different trapdoors every time for the same keyword. But the schemes in [3,5,22,36,40–42] always generate the same trapdoor for the same keyword. According to the discussion done in [23], the schemes in [3,5,22,36,41,42] may reveal keywords due to the leakage of encryption pattern.

Table 4
Comparison of communication costs.

Scheme	Security level (bit)	L1	L2	L3
Boneh et al. [3]	80	$\lvert\mathbf{G}_1\rvert = 128$	$\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 148$	$\lvert\mathbf{G}_1\rvert = 128$
	112	$\lvert\mathbf{G}_1\rvert = 256$	$\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 284$	$\lvert\mathbf{G}_1\rvert = 256$
	128	$\lvert\mathbf{G}_1\rvert = 384$	$\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 416$	$\lvert\mathbf{G}_1\rvert = 384$
	192	$\lvert\mathbf{G}_1\rvert = 960$	$\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 1008$	$\lvert\mathbf{G}_1\rvert = 960$
	256	$\lvert\mathbf{G}_1\rvert = 1920$	$\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 1984$	$\lvert\mathbf{G}_1\rvert = 1920$
Rhee et al. [5]	80	$3\lvert\mathbf{G}_1\rvert = 384$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 276$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 296$
	112	$3\lvert\mathbf{G}_1\rvert = 768$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 540$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 568$
	128	$3\lvert\mathbf{G}_1\rvert = 1152$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 800$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 832$
	192	$3\lvert\mathbf{G}_1\rvert = 2880$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 1968$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 2016$
	256	$3\lvert\mathbf{G}_1\rvert = 5760$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 3904$	$\lvert\mathbf{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert = 3968$
Hu and Liu [19]	80	$3\lvert\mathbf{G}_1\rvert = 384$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 276$	$2\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 276$
	112	$3\lvert\mathbf{G}_1\rvert = 768$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 540$	$2\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 540$
	128	$3\lvert\mathbf{G}_1\rvert = 1152$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 800$	$2\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 800$
	192	$3\lvert\mathbf{G}_1\rvert = 2280$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 1968$	$2\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 1968$
	256	$3\lvert\mathbf{G}_1\rvert = 5760$	$\lvert\mathbf{G}_2\rvert + \lvert\mathbb{Z}_p\rvert = 3904$	$2\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 3904$
Huang and Li [22]	80	$2\lvert\mathbf{G}_1\rvert = 256$	$2\lvert\mathbf{G}_1\rvert = 256$	$\lvert\mathbf{G}_2\rvert = 256$
	112	$2\lvert\mathbf{G}_1\rvert = 512$	$2\lvert\mathbf{G}_1\rvert = 512$	$\lvert\mathbf{G}_2\rvert = 512$
	128	$2\lvert\mathbf{G}_1\rvert = 768$	$2\lvert\mathbf{G}_1\rvert = 768$	$\lvert\mathbf{G}_2\rvert = 768$
	192	$2\lvert\mathbf{G}_1\rvert = 1920$	$2\lvert\mathbf{G}_1\rvert = 1920$	$\lvert\mathbf{G}_2\rvert = 1920$
	256	$2\lvert\mathbf{G}_1\rvert = 3840$	$2\lvert\mathbf{G}_1\rvert = 3840$	$\lvert\mathbf{G}_2\rvert = 3840$
Rhee et al. [36]	80	$2\lvert\mathbf{G}_1\rvert = 256$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 148$	$2\lvert\mathbf{G}_1\rvert = 256$
	112	$2\lvert\mathbf{G}_1\rvert = 512$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 284$	$2\lvert\mathbf{G}_1\rvert = 512$
	128	$2\lvert\mathbf{G}_1\rvert = 768$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 416$	$2\lvert\mathbf{G}_1\rvert = 768$
	192	$2\lvert\mathbf{G}_1\rvert = 1920$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 1008$	$2\lvert\mathbf{G}_1\rvert = 1920$
	256	$2\lvert\mathbf{G}_1\rvert = 3840$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 1985$	$2\lvert\mathbf{G}_1\rvert = 3840$
Fang et al. [39]	80	$2\lvert\mathbf{G}_1\rvert = 256$	$2\lvert\mathbf{G}_1\rvert + 2\lvert\mathbf{G}_2\rvert = 768$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 148$
	112	$2\lvert\mathbf{G}_1\rvert = 512$	$2\lvert\mathbf{G}_1\rvert + 2\lvert\mathbf{G}_2\rvert = 1536$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 284$
	128	$2\lvert\mathbf{G}_1\rvert = 768$	$2\lvert\mathbf{G}_1\rvert + 2\lvert\mathbf{G}_2\rvert = 2304$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 416$
	192	$2\lvert\mathbf{G}_1\rvert = 1920$	$2\lvert\mathbf{G}_1\rvert + 2\lvert\mathbf{G}_2\rvert = 5760$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 1008$
	256	$2\lvert\mathbf{G}_1\rvert = 3840$	$2\lvert\mathbf{G}_1\rvert + 2\lvert\mathbf{G}_2\rvert = 11\,520$	$\lvert\mathbf{G}_1\rvert + \lvert\mathbb{Z}_p\rvert = 1984$
Shao and Yang [40]	80	$2\lvert\mathbf{G}_1\rvert = 256$	$5\lvert\mathbf{G}_1\rvert + 3\lvert\mathbf{G}_2\rvert = 1408$	$3\lvert\mathbf{G}_1\rvert = 384$
	112	$2\lvert\mathbf{G}_1\rvert = 512$	$5\lvert\mathbf{G}_1\rvert + 3\lvert\mathbf{G}_2\rvert = 2816$	$3\lvert\mathbf{G}_1\rvert = 768$
	128	$2\lvert\mathbf{G}_1\rvert = 768$	$5\lvert\mathbf{G}_1\rvert + 3\lvert\mathbf{G}_2\rvert = 4224$	$3\lvert\mathbf{G}_1\rvert = 1152$
	192	$2\lvert\mathbf{G}_1\rvert = 1920$	$5\lvert\mathbf{G}_1\rvert + 3\lvert\mathbf{G}_2\rvert = 10\,560$	$3\lvert\mathbf{G}_1\rvert = 2880$
	256	$2\lvert\mathbf{G}_1\rvert = 3840$	$5\lvert\mathbf{G}_1\rvert + 3\lvert\mathbf{G}_2\rvert = 21\,120$	$3\lvert\mathbf{G}_1\rvert = 5760$
Proposed	80	$2\lvert\mathbf{G}_1\rvert = 256$	$2\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 276$	$2\lvert\mathbf{G}_1\rvert + 2\lvert H\rvert = 296$
	112	$2\lvert\mathbf{G}_1\rvert = 512$	$2\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 540$	$2\lvert\mathbf{G}_1\rvert + 2\lvert H\rvert = 568$
	128	$2\lvert\mathbf{G}_1\rvert = 768$	$2\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 800$	$2\lvert\mathbf{G}_1\rvert + 2\lvert H\rvert = 832$
	192	$2\lvert\mathbf{G}_1\rvert = 1920$	$2\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 1968$	$2\lvert\mathbf{G}_1\rvert + 2\lvert H\rvert = 2016$
	256	$2\lvert\mathbf{G}_1\rvert = 3840$	$2\lvert\mathbf{G}_1\rvert + \lvert H\rvert = 3904$	$2\lvert\mathbf{G}_1\rvert + 2\lvert H\rvert = 3968$

L1: Length of the public key (byte); L2: Length of the ciphertext (byte); L3: Length of the trapdoor (byte).

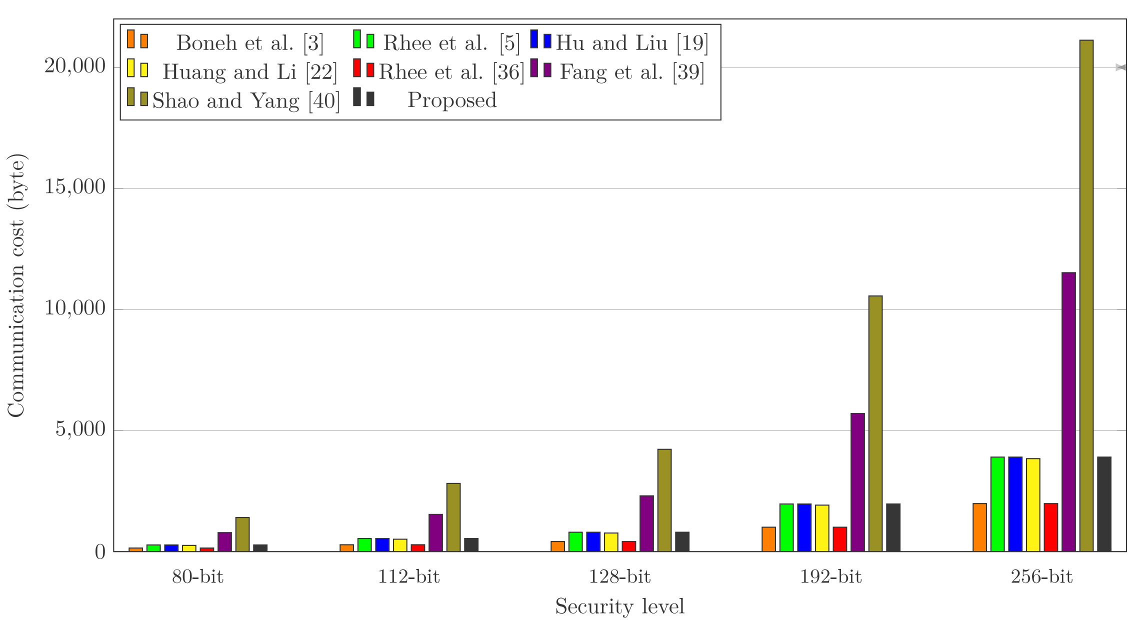

Fig. 11. Communication overheads for ciphertext of different protocols.

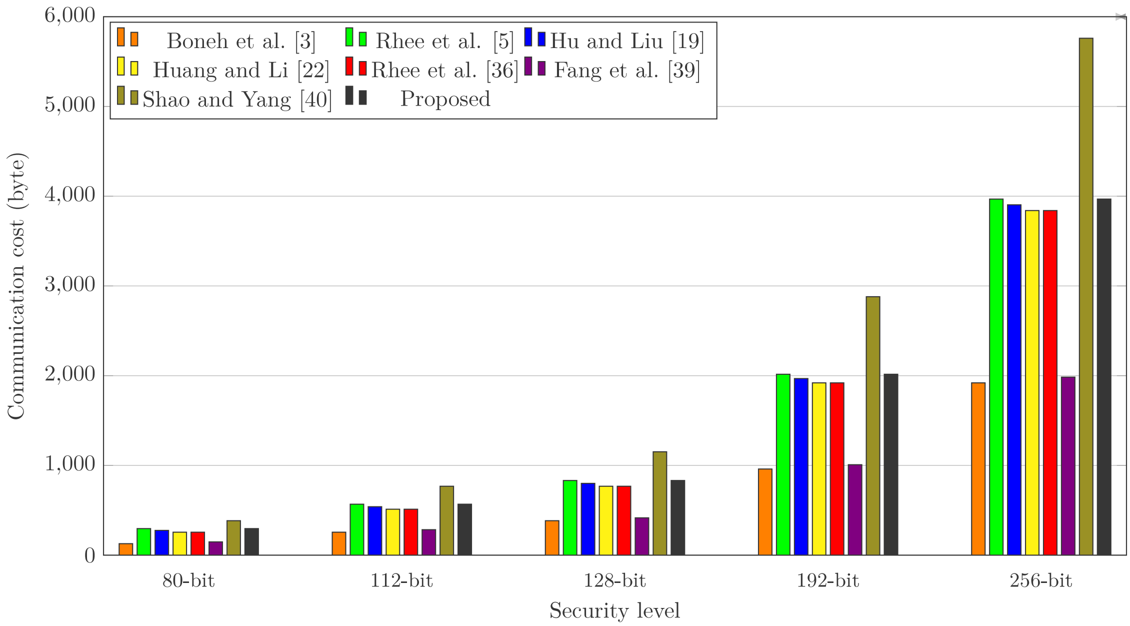

Fig. 12. Communication overheads for trapdoor of different protocols.

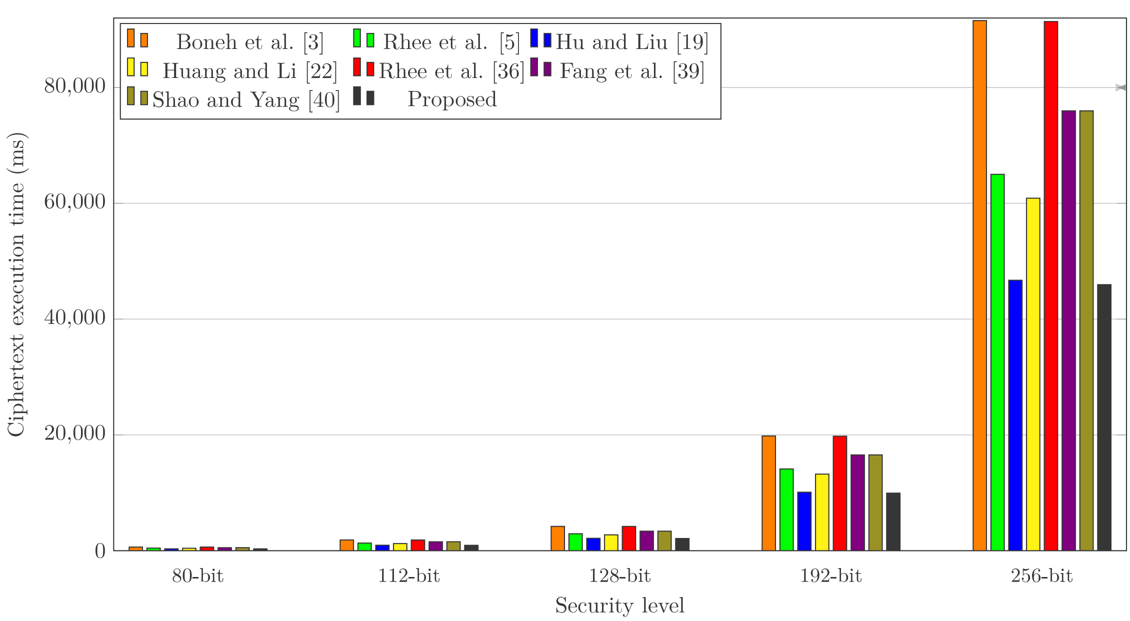

Fig. 13. Comparison of execution time of ciphertext generation (**PEKS** algorithm).

Table 5
Execution time (ms) of various cryptographic operation.

Notation	Description	80-bit	112-bit	128-bit	192-bit	256-bit
T_P	Bilinear pairing	3.1185	9.2226	20.913	98.685	456.2341
T_M	Multiplication operation	1.0391	3.0742	6.971	32.5128	152.2241
T_H	Cryptographic hash function	0.0052	0.0052	0.0075	0.0284	0.105
T_E	Modular exponent operation	1.0491	3.0757	6.3644	33.206	150.9794

8. Conclusions and future work

We propose a new PEKS scheme to outsource the data securely to a cloud server over the Internet. The proposed PEKS scheme can prevent keyword-guessing attacks by enhancing trapdoor security with a designated tester. In this scheme, the cloud server acts as a designated tester. The scheme is different from other existing schemes. In the proposed scheme, the data owner's private key is required to encrypt the list of keywords so that the cloud server cannot encrypt random keywords again, and thus could not relaunch the keyword-guessing attacks. In addition, only a cloud server (designated tester) can run the trapdoor test algorithm in the proposed scheme. The proposed scheme is secure by providing the trapdoor and ciphertext indistinguishability in the CPA model based on the hardness assumption of HDH and DBDH

Table 6
Comparison of execution costs (ms).

Scheme	Security level (bit)	PEKS	Trapdoor	Test
Boneh et al. [3]	80	626.3182	126.516	327.9885
	112	1851.1914	369.708	968.919
	128	4196.0788	764.628	2196.6525
	192	19 806.252	3988.128	10 364.907
	256	91 559.2788	18 130.128	47 915.6055
Rhee et al. [5]	80	446.9156	126.516	417.28
	112	1316.9491	369.708	1230.35
	128	2923.8214	764.628	2728.49
	192	14 116.151	3988.128	13 191.94
	256	64 999.4363	18 130.128	60 731.85
Hu and Liu [19]	80	320.1761	122.2056	416.7652
	112	944.8209	357.2997	1229.8352
	128	2140.9904	739.627	2727.7475
	192	10 104.756	3854.0428	13 189.1284
	256	46 707.8576	17 525.3551	60 721.455
Huang and Li [22]	80	418.3291	313.4191	623.7
	112	1233.4257	925.8557	1844.52
	128	2734.8544	2098.4144	4182.6
	192	13 225.146	9904.546	19 737
	256	60 882.8294	45 784.8894	68 435.115
Rhee et al. [36]	80	625.2691	116.9601	421.9667
	112	1848.1157	341.9212	1245.2025
	128	4189.7144	707.805	2761.9884
	192	19 773.046	3688.0128	13 353.5038
	256	91 408.2994	16 770.4581	61 480.7876
Fang et al. [39]	80	522.7243	209.82	329.6211
	112	1540.4809	615.14	977.6083
	128	3370.5519	1272.88	2213.7525
	192	16 542.9344	6641.2	10 458.3714
	256	75 970.3744	30 195.88	48 356.8871
Shao and Yang [40]	80	522.7243	209.82	537.0292
	112	1540.4809	615.14	1574.7456
	128	3370.5519	1272.88	3265.8595
	192	16 542.9344	6641.2	16 997.7684
	256	75 970.3744	30 195.88	77 314.7414
Proposed	80	314.9882	106.9995	323.2989
	112	929.4514	311.6946	956.0829
	128	2105.5288	644.926	2167.3819
	192	9940.592	3358.8496	10 230.3022
	256	45 946.3688	15 271.3741	47 295.4208

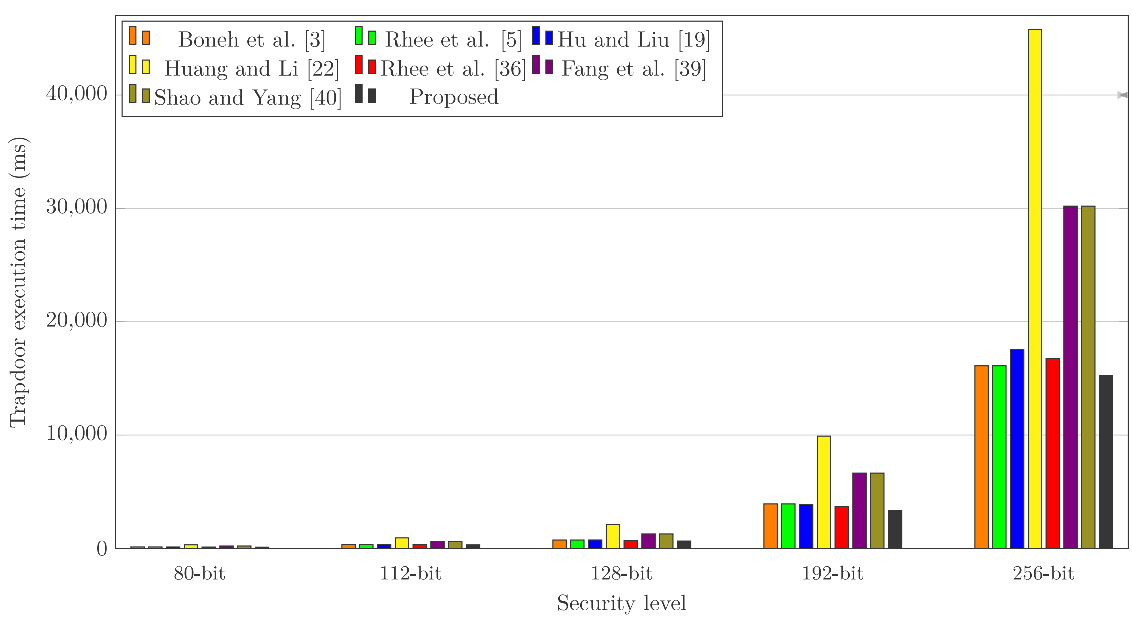

Fig. 14. Comparison of trapdoor generation (**Trapdoor** algorithm).

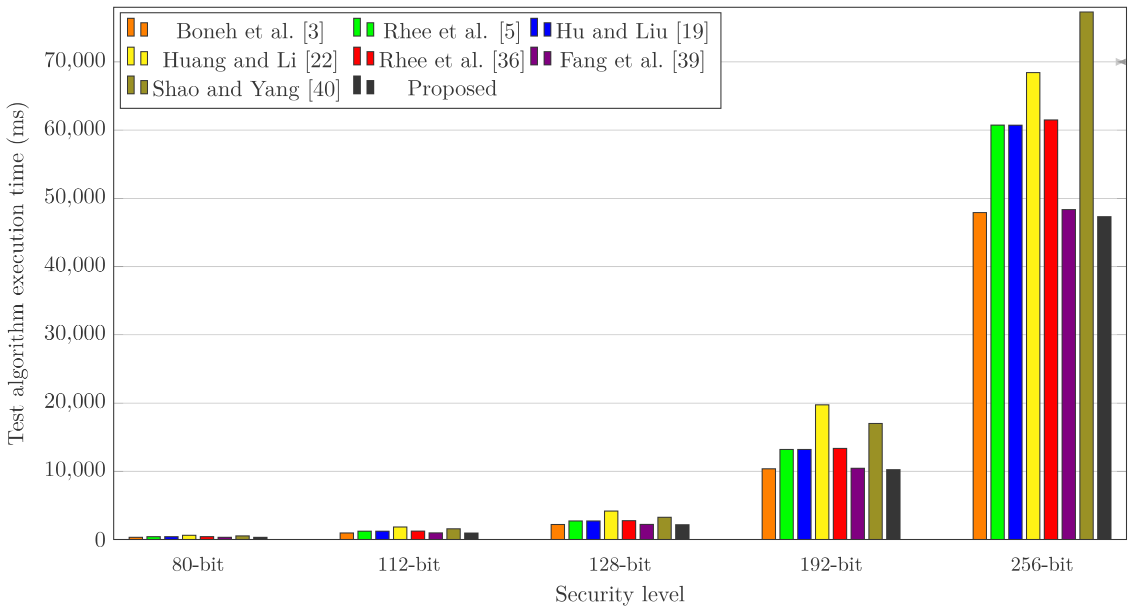

Fig. 15. Comparison of trapdoor verification (**Test** algorithm).

problems. We also show that the proposed scheme is efficient regarding the size of the public keys and ciphertext. Moreover, the real testbed experimental results demonstrate that the proposed scheme is practical for the cloud computing scenario for the keyword search on encrypted data.

In this work, we only considered a single chosen keyword-guessing attack. Our future aim is to design a provably secure PEKS scheme to provide the ciphertext and trapdoor indistinguishability in the CPA model against chosen multi-keyword-guessing attacks.

CRediT authorship contribution statement

Sudeep Ghosh: Conceptualization, Methodology, Validation, Formal analysis, Writing – review & editing, Visualization. **SK Hafizul Islam:** Supervision, Conceptualization, Methodology, Validation, Formal analysis, Writing – review & editing, Visualization. **Abhishek Bisht:** Methodology, Validation, Visualization. **Ashok Kumar Das:** Supervision, Conceptualization, Formal analysis, Writing – review & editing, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article

References

[1] P. Mell, T. Grance, Draft NIST working definition of cloud computing, Ref. June. 3rd 15 (32) (2009) 2.
[2] R. Smith, Computing in the cloud, Res.-Technol. Manag. 52 (5) (2009) 65–68.
[3] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, pp. 506–522.
[4] J. Baek, R. Safavi-Naini, W. Susilo, Public key encryption with keyword search revisited, in: International Conference on Computational Science and Its Applications, Springer, 2008, pp. 1249–1259.
[5] H.S. Rhee, J.H. Park, W. Susilo, D.H. Lee, Improved searchable public key encryption with designated tester, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, 2009, pp. 376–379.
[6] Q. Tang, Public key encryption schemes supporting equality test with authorisation of different granularity, Int. J. Appl. Cryptogr. 2 (4) (2012) 304–321.
[7] S. Ma, Q. Huang, M. Zhang, B. Yang, Efficient public key encryption with equality test supporting flexible authorization, IEEE Trans. Inf. Forensics Secur. 10 (3) (2014) 458–470.
[8] W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, IEEE Trans. Parallel Distrib. Syst. 27 (4) (2014) 1187–1198.
[9] H. Su, Z. Zhu, L. Sun, Online/offline attribute-based encryption with keyword search against keyword guessing attack, in: 2017 3rd IEEE International Conference on Computer and Communications, ICCC, IEEE, 2017, pp. 1487–1492.
[10] A. Arriaga, Q. Tang, P. Ryan, Trapdoor privacy in asymmetric searchable encryption schemes, in: International Conference on Cryptology in Africa, Springer, 2014, pp. 31–50.
[11] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: International Conference on Applied Cryptography and Network Security, Springer, 2004, pp. 31–45.
[12] Z. Lv, C. Hong, M. Zhang, D. Feng, Expressive and secure searchable encryption in the public key setting, in: International Conference on Information Security, Springer, 2014, pp. 364–376.
[13] D.J. Park, K. Kim, P.J. Lee, Public key encryption with conjunctive field keyword search, in: International Workshop on Information Security Applications, Springer, 2004, pp. 73–86.
[14] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2013) 222–233.
[15] H. Li, D. Liu, Y. Dai, T.H. Luan, X.S. Shen, Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage, IEEE Trans. Emerg. Top. Comput. 3 (1) (2014) 127–138.
[16] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, IEEE, 2000, pp. 44–55.
[17] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, M. Steiner, Highly-scalable searchable symmetric encryption with support for boolean queries, in: Annual Cryptology Conference, Springer, 2013, pp. 353–373.
[18] Y. Zhao, X. Chen, H. Ma, Q. Tang, H. Zhu, A new trapdoor-indistinguishable public key encryption with keyword search, J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. 3 (1/2) (2012) 72–81.
[19] C. Hu, P. Liu, A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension, in: International Conference on Computer Science, Environment, Ecoinformatics, and Education, Springer, 2011, pp. 131–136.
[20] C. Hu, P. Liu, An enhanced searchable public key encryption scheme with a designated tester and its extensions, J. Comput. 7 (3) (2012) 716–723.

[21] J. Ni, Y. Yu, Q. Xia, L. Niu, Cryptanalysis of two searchable public key encryption schemes with a designated tester, J. Inf. Comput. Sci. 9 (16) (2012) 4819–4825.
[22] Q. Huang, H. Li, An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks, Inform. Sci. 403 (2017) 1–14.
[23] C. Liu, L. Zhu, M. Wang, Y.-a. Tan, Search pattern leakage in searchable encryption: Attacks and new construction, Inform. Sci. 265 (2014) 176–188.
[24] X. Jiang, J. Yu, J. Yan, R. Hao, Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data, Inform. Sci. 403 (2017) 22–41.
[25] Y. Miao, J. Weng, X. Liu, K.-K.R. Choo, Z. Liu, H. Li, Enabling verifiable multiple keywords search over encrypted cloud data, Inform. Sci. 465 (2018) 21–37.
[26] Z. Liu, T. Li, P. Li, C. Jia, J. Li, Verifiable searchable encryption with aggregate keys for data sharing system, Future Gener. Comput. Syst. 78 (2018) 778–788.
[27] T. Fuhr, P. Paillier, Decryptable searchable encryption, in: International Conference on Provable Security, Springer, 2007, pp. 228–236.
[28] J. Shao, Z. Cao, X. Liang, H. Lin, Proxy re-encryption with keyword search, Inform. Sci. 180 (13) (2010) 2576–2587.
[29] Z. Chen, S. Li, Q. Huang, Y. Wang, S. Zhou, A restricted proxy re-encryption with keyword search for fine-grained data access control in cloud storage, Concurr. Comput.: Pract. Exper. 28 (10) (2016) 2858–2876.
[30] Y. Zhou, Z. Hu, F. Li, Searchable public-key encryption with cryptographic reverse firewalls for cloud storage, IEEE Trans. Cloud Comput. (2021).
[31] B. Chen, L. Wu, L. Li, K.-K.R. Choo, D. He, A parallel and forward private searchable public-key encryption for cloud-based data sharing, IEEE Access 8 (2020) 28009–28020.
[32] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: Annual International Cryptology Conference, Springer, 2001, pp. 213–229.
[33] X. Boyen, The uber-assumption family, in: International Conference on Pairing-Based Cryptography, Springer, 2008, pp. 39–56.
[34] A. Joux, A one round protocol for tripartite Diffie–Hellman, in: International Algorithmic Number Theory Symposium, Springer, 2000, pp. 385–393.
[35] M. Abdalla, M. Bellare, P. Rogaway, The oracle Diffie-Hellman assumptions and an analysis of DHIES, in: Cryptographers' Track at the RSA Conference, Springer, 2001, pp. 143–158.
[36] H.S. Rhee, J.H. Park, W. Susilo, D.H. Lee, Trapdoor security in a searchable public-key encryption scheme with a designated tester, J. Syst. Softw. 83 (5) (2010) 763–771.
[37] J.-S. Coron, On the exact security of full domain hash, in: Annual International Cryptology Conference, Springer, 2000, pp. 229–235.
[38] B. Lynn, PBC library manual 0.5. 11, 2006, https://crypto.stanford.edu/pbc/.
[39] L. Fang, W. Susilo, C. Ge, J. Wang, A secure channel free public key encryption with keyword search scheme without random oracle, in: International Conference on Cryptology and Network Security, Springer, 2009, pp. 248–258.
[40] Z.-Y. Shao, B. Yang, On security against the server in designated tester public key encryption with keyword search, Inform. Process. Lett. 115 (12) (2015) 957–961.
[41] J. Baek, R. Safavi-Naini, W. Susilo, On the integration of public key data encryption and public key encryption with keyword search, in: International Conference on Information Security, Springer, 2006, pp. 217–232.
[42] C. Gu, Y. Zhu, H. Pan, Efficient public key encryption with keyword search schemes from pairings, in: International Conference on Information Security and Cryptology, Springer, 2007, pp. 372–383.

Sudeep Ghosh received the B.Tech. degree in Information Technology from Maulana Abul Kalam Azad University of Technology, West Bengal, India, in 2008. He received the M.E. degree in Information Technology from the Indian Institute of Engineering Science and Technology, Shibpur (IIEST Shibpur), West Bengal, India, in 2011. He is now working as an Assistant Professor in the Department of Information Technology, Guru Nanak Institute of Technology Kolkata, West Bengal 700114, India. He is pursuing Ph.D. degree in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India. His current research interests includes Cryptography and Information Security, Cloud Security, and Blockchain Security.

SK Hafizul Islam received the M.Sc. degree in applied mathematics from Vidyasagar University, Midnapore, India, in 2006, and the M.Tech. degree in Computer Application and the Ph.D. degree in Computer Science and Engineering in 2009 and 2013, respectively, from Indian Institute of Technology [IIT (ISM)] Dhanbad, Jharkhand, India, under the **INSPIRE Fellowship Ph.D. Program (funded by the Department of Science and Technology, Government of India)**. He is currently an Assistant Professor in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani (IIIT Kalyani), West Bengal, India. Before joining the IIIT Kalyani, he was an Assistant Professor with the Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani (BITS Pilani), Rajasthan, India. He has more than ten years of teaching and thirteen years of research experience. He has authored or co-authored 130 research papers in journals and conference proceedings of international reputes. His research interests include Cryptography, Information Security, Neural Cryptography, Lattice-based Cryptography, IoT & Blockchain Security, and Deep Learning. He has edited four books for the publishers **Scrivener-Wiley, Elsevier**, and **CRC Press**. He is an **Associate Editor** for **"IEEE Transactions on Intelligent Transportation Systems", "IEEE Access", "InternationalJournal of Communication Systems (Wiley)", "Telecommunication Systems (Springer)", "IET Wireless Sensor Systems", "Security and Privacy (Wiley)", "Array-Journal (Elsevier)", "Wireless Communications and Mobile Computing (Hindwai)",** and **"Journal of Cloud Computing (Springer)".** He received the University Gold Medal, the S. D. Singha Memorial Endowment Gold Medal, and the **Sabitri Parya Memorial Endowment Gold Medal** from Vidyasagar University in 2006. He also received the **University Gold Medal** from IIT(ISM) Dhanbad in 2009 and the OPERA award from BITS Pilani in 2015. He is a senior member of IEEE and a member of ACM.

Abhishek Bisht completed his B.Tech. degree in Computer Science from University of Petroleum and Energy Studies, Dehradun, India, in the year 2021. At present, he is pursuing his MS degree in Computer Science and Engineering at the Center for Security, Theory and Algorithmic Research, IIIT, Hyderabad, India. His research interests are cyber security, searchable encryption, IoT security and blockchain technology.

Ashok Kumar Das received a Ph.D. degree in computer science and engineering, an M.Tech. degree in computer science and data processing, and an M.Sc. degree in mathematics from IIT Kharagpur, India. He is currently an Associate Professor with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India. He was also working as a visiting faculty with the Virginia Modeling, Analysis and Simulation Center, Old Dominion University, Suffolk, VA 23435, USA. His current research interests include cryptography, network security, security in vehicular ad hoc networks, smart grids, smart homes, Internet of Things (IoT), Internet of Drones, Internet of Vehicles, Cyber–Physical Systems (CPS) and cloud computing, intrusion detection, blockchain and AI/ML security. He has authored over 350 papers in international journals and conferences in the above areas, including over 295 reputed journal papers. He was a recipient of the Institute Silver Medal from IIT Kharagpur. He has been listed in the Web of Science (Clarivate$^{\text{TM}}$) Highly Cited Researcher 2022 in recognition of his exceptional research performance. He was/is on the editorial board of IEEE Systems Journal, Journal of Network and Computer Applications (Elsevier), Computer Communications (Elsevier), Journal of Cloud Computing (Springer), Cyber Security and Applications (Elsevier), IET Communications, KSII Transactions on Internet and Information Systems, and International Journal of Internet Technology and Secured Transactions (Inderscience), and has served as a Program Committee Member in many international conferences. He also served as one of the Technical Program Committee Chairs of the first International Congress on Blockchain and Applications (BLOCKCHAIN'19), Avila, Spain, June 2019, International Conference on Applied Soft Computing and Communication Networks (ACN'20), October 2020, Chennai, India, and second International Congress on Blockchain and Applications (BLOCKCHAIN'20), L'Aquila, Italy, October 2020. His Google Scholar h-index is 76 and i10-index is 217 with over 16,200 citations. He is a senior member of the IEEE.