cryptography

MDPI

Article

Privacy-Preserving and Efficient Public Key Encryption with Keyword Search Based on CP-ABE in Cloud

Yunhong Zhou, Shihui Zheng and Licheng Wang *

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; yunhongzhou@bupt.edu.cn (Y.Z.); shihuizh@bupt.edu.cn (S.Z.)
* Correspondence: wanglc@bupt.edu.cn; Tel.: +86-13581632945

Received: 16 August 2020; Accepted: 8 October 2020; Published: 13 October 2020

Abstract: In the area of searchable encryption, public key encryption with keyword search (PEKS) has been a critically important and promising technique which provides secure search over encrypted data in cloud computing. PEKS can protect user data privacy without affecting the usage of the data stored in the untrusted cloud server environment. However, most of the existing PEKS schemes concentrate on data users' rich search functionalities, regardless of their search permission. Attribute-based encryption technology is a good method to solve the security issues, which provides fine-grained access control to the encrypted data. In this paper, we propose a privacy-preserving and efficient public key encryption with keyword search scheme by using the ciphertext-policy attribute-based encryption (CP-ABE) technique to support both fine-grained access control and keyword search over encrypted data simultaneously. We formalize the security definition, and prove that our scheme achieves selective indistinguishability security against an adaptive chosen keyword attack. Finally, we present the performance analysis in terms of theoretical analysis and experimental analysis, and demonstrate the efficiency of our scheme.

Keywords: cloud computing; public key encryption with keyword search; data privacy; access control; ciphertext-policy attribute-based encryption

1. Introduction

With the rise and popularity of cloud computing technology, more and more data users are motivated to outsource their data to cloud for storage, and start to enjoy the various advantages brought by cloud computing, such as large storage capacity, flexible accessibility and vast computationality. By outsourcing data to the cloud, data users are relieved from the heavy burden of local storage and management costs. However, as the cloud server is not fully trusted and the outsourced data may contain sensitive information, sensitive data privacy and security in the cloud naturally become a primary concern for users. In order to address this issue, sensitive data should be encrypted before outsourcing to the cloud. Typical practical applications include cloud-based storage systems, such as electronic medical record systems where electronic health record data-sharing can help doctors to effectively assess patients' conditions and make correct treatments. To prevent the leakage of patient information, we store the encrypted medical data on the cloud. Although encryption can ensure the confidentiality of data, it brings the inconvenience of data access and processing. For example, when doctors want to query the required medical data, the cloud server needs to perform search functions without knowing the content of the data. However, the conventional information retrieval methods based on plaintext can not directly be used on ciphertext. Therefore, how to search over encrypted data is of importance and becomes a new challenge.

Homomorphic encryption [1,2] can operate on encrypted data, it can be used to search the keyword over encrypted data theoretically. However, constructions based on homomorphic encryption needs large key sizes and brings huge communication costs. To better solve the keyword search problem over encrypted data, searchable encryption (SE) was proposed as an efficient solution. SE not only enables users to search queries over encrypted data stored on an untrusted server but also protects data privacy and search privacy. Compared with homomorphic encryption, SE is designed for keyword search, which is more efficient.

Song et al. [3] seminally proposed the notion of searchable encryption, which was the first concrete symmetric searchable encryption (SSE) construction. Subsequently, a number of efficient SSE schemes [4–10] have been proposed. However, data users have to securely share key for encryption in SSE schemes due to the property of symmetric cryptography. Therefore, most SSE schemes inevitably suffer from secret key distribution and management problems. Later, at Eurocrypt'04, Boneh et al. [11] first introduced the concept of public key searchable encryption based on the public key encryption algorithm, namely public key encryption with keyword search (PEKS). PEKS solved the secret key management and distribution yield by SSE, so PEKS has aroused great attention in research and has a broader application prospect.

PEKS is a critically important and promising technique, and this paper concentrates on PEKS. At present, various PEKS schemes have been proposed in the literature [12–20], and PEKS has been rapidly developed [21]. Most existing PEKS schemes focus on data users' rich search functionalities, such as conjunctive keyword search [14] and multi-keyword search [20]. In the above schemes, all search users can be regarded as authorized users, who can have unrestricted access to the system. However, it is not suitable for practical requirements. For example, only the mail receiver is allowed to search on the encrypted emails in the cloud-based email system, while other users have no such permissions. To solve this problem, CP-ABE technology is widely adopted as a viable tool to achieve flexible data access control over encrypted data, which can gain one-to-many encryption instead of one-to-one.

Zheng et al. [22] constructed an attribute-based keyword search scheme on the basis of CP-ABE, however, computation and storage costs grow linearly with the number of system attributes. Yin et al. [23] improved the work of Zheng, but encryption and key generation operations also lead to much higher computational overheads. Thus, in this paper, we propose a new privacy-preserving and efficient public key encryption with keyword search scheme based on ciphertext-policy attribute-based encryption (PEKS-CPABE) to support both fine-grained access control and keyword search over encrypted data simultaneously. Specifically, the main contributions of this paper can be summarized as follows.

- We propose an efficient and privacy-preserving PEKS-CPABE scheme, which enables the data owner to control the search and use of its outsourced encrypted data in the cloud according to its access control policy. Our scheme can achieve keyword search queries over encrypted data with fine-grained access control;
- We formalize the security definition, and prove that our scheme achieves selective indistinguishability security against an adaptive chosen keyword attack and can also ensure keyword privacy;
- We present the performance analysis in terms of theoretical analysis and experimental analysis, and demonstrate the efficiency of our scheme. Finally, the experimental results show that our PEKS-CPABE scheme performs better than the CP-ABKS scheme proposed in [22] and CP-ABSE scheme proposed in [23].

The remainder of this paper is organized as follows: Section 2 presents an overview of related work. Section 3 reviews some necessary and basic preliminary notions used in this paper. Section 4 defines the system model, algorithm description and security model of our proposed scheme. Section 5 introduces the concrete construction of our proposed scheme. Section 6 provides detailed security

proof of our proposed scheme. Section 7 shows the performance analysis of our proposed scheme in terms of theoretical analysis and experimental analysis. Finally, Section 8 summarizes this paper.

2. Related Work

In 2004, Boneh et al. [11] devised the first public key searchable encryption scheme based on a standard public key encryption algorithm. In this scheme, multiple users are allowed to encrypt data using the public key, but only the private key holder can search the keyword over encrypted data. Inspired by Boneh et al. pioneering work, more and more researchers have been working on the public key searchable encryption to achieve various functionalities. Park et al. [13] first proposed a PEKS scheme to support conjunctive keyword search on encrypted data. Subsequently, Boneh and Waters [14] extended PEKS to support conjunctive, subset, and range queries on encrypted data. Later on, Lv et al. [18] proposed an expressive and secure PEKS scheme based on composite-order groups to support conjunctive, disjunctive and negation search operations that can be extended to support a range search. Huang and Li [19] proposed a public key authenticated encryption with keyword search scheme that can resist the inside keyword guessing attack, in which the data user can use the secret key to authenticate the data. Recently, Zhang et al. [20] constructed a novel PEKS scheme supporting semantic multi-keyword search by leveraging an efficient inner product encryption technology. These schemes guarantee that data users are able to efficiently perform search queries and prevent the untrusted cloud server from infringing data confidentiality and search privacy.

Attribute-based encryption (ABE), as a useful data encryption tool to address the problem of fine-grained data sharing and decentralized access control, was first proposed by Sahai and Waters [24]. This kind of new cryptographic technology enables users to achieve access control over encrypted data by using an access control policy associated with the private key or ciphertext. There are two types of attribute-based encryption, depending on whether the private key or ciphertext is associated with the access control policy, namely key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). Goyal et al. [25] proposed the first KP-ABE scheme, where the private key is associated with an access control policy and the ciphertext is associated with attributes. A user can decrypt a ciphertext only if the attributes that are used for encryption satisfy the access policy on the user private key. Bethencourt et al. [26] proposed the first CP-ABE scheme, where the ciphertext is associated with an access control policy and the key is associated with attributes. A user can decrypt a ciphertext only if the attributes on the user private key satisfy the access policy associated with the ciphertext. Compared with KP-ABE, CP-ABE is a preferred choice for designing an access control mechanism, because it is conceptually closer to traditional role-based access control.

Based on the special features of ABE, Zheng et al. [22] proposed a ciphertext-policy attribute-based keyword search (CP-ABKS) scheme, in which keywords are encrypted with an access control policy so that only data users with a legitimate credential can generate the trapdoor used to search over encrypted data. Dong et al. [27] introduced an enhanced attribute-based keyword search scheme via an online/offline approach. Their schemes mainly consider resource-constrained mobile devices, and allow data owners and data users to perform related operations online and offline. However, two encryption and decryption operations incur huge computational costs. Li et al. [28] proposed an attribute-based keyword search scheme with outsourcing key-issuing and outsourcing decryption, which is shown to be secure against chosen-plaintext attack, but it introduces three cloud storage providers and needs a number of expenses. Sun et al. [29] designed an authorized keyword search scheme with user revocation by utilizing CP-ABE technology, however, the computational costs in the trapdoor generation grow linearly with the number of system attributes. Qiu et al. [30] proposed a hidden policy attribute-based keyword search scheme to protect the privacy of the access control policy, which incurs abundant computational costs at the same time. Recently, Yin et al. [23] proposed a ciphertext-policy attribute-based searchable encryption scheme, and improved the scheme of Zheng et al. Inspired by both the work of Zheng and Yin, in this paper, we propose a new privacy-preserving and efficient public key encryption with a keyword search scheme based on

ciphertext-policy attribute-based encryption (PEKS-CPABE) to support both fine-grained access control and keyword search over encrypted data simultaneously.

3. Preliminaries

In this section, we review some necessary and basic preliminary notions used in this paper.

3.1. Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups with the same prime order, p. g is a generator of $\mathbb{G}$. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a computable bilinear map. The map $\hat{e}$ has the following properties:

- Bilinearity: Given $a, b \in \mathbb{Z}_p^*$ and for all $g \in \mathbb{G}$, there exists $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$;
- Non-degeneracy: $\hat{e}(g, g) \neq 1$;
- Computability: Given $a, b \in \mathbb{Z}_p^*$ and for all $g \in \mathbb{G}$, there is an efficient algorithm to compute $\hat{e}(g^a, g^b) \in \mathbb{G}_T$.

3.2. Decisional Bilinear Diffie-Hellman Assumption

Decisional Bilinear Diffie–Hellman (DBDH) Assumption is given as follows: Let $\mathbb{G}$ be a cyclic group with prime order p. g is a generator of $\mathbb{G}$. $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. We define the advantage as ϵ of a PPT adversary $\mathcal{A}$ to solve the DBDH problem as $Adv_{\mathcal{A}}^{DBDH}(\lambda) = |Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 1] - Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^z) = 1]| < \epsilon$, where $a, b, c, z \in \mathbb{Z}_p^*$. We say that the DBDH assumption holds if the PPT adversary has a negligible advantage ϵ in solving the DBDH problem.

3.3. Access Structure

Access structure is defined by the concepts of an authorized access subset and unauthorized access subset. Let $P = \{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^P$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ which is a non-empty subset for $\{P_1, P_2, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^P \backslash \{\varnothing\}$. The set in $\mathbb{A}$ is called an authorized set, and the set not in $\mathbb{A}$ is called an unauthorized set.

3.4. Access Tree

An access tree is usually used to represent an access control policy. Let T be an access tree, and each non-leaf represents a threshold gate described by a threshold value and the number of its children nodes. Let num_x represent the number of children of node x, and the children be labeled from the left to the right as $\{1, \cdots, num_x\}$. Let k_x represent the threshold value of x with $1 \leq k_x \leq num_x$. If $k_x = 1$, the logic gate of node x is an "OR" gate. If $k_x = num_x$, the logic gate of node x is an "AND" gate. In an access tree T, a leaf is associated with an attribute, so each leaf node of T is described by an attribute and a threshold value $k_x = 1$.

Let $parent(x)$ denote the parent of node x, $att(x)$ denote the attribute associated with leaf node x, and $lvs(T)$ denote the set of leaves of the access tree T. Let $index(x)$ denote the label of the node x, and T_x represent the subtree of T rooted at node x (thus, $T_{root} = T$). If an attribute set γ meets the access tree T_x, we denotes it as $T_x(\gamma) = 1$. Otherwise, $T_x(\gamma) = 0$. When x is a leaf node, if and only if $att(x) \in \gamma$, then $T_x(\gamma) = 1$. When x is a non-leaf node, we can compute $T_{x'}(\gamma)$ for each child node x' of node x. If at least k_x children of the node x return 1, then $T_x(\gamma) = 1$.

4. System Model

In this section, we present the system model, the algorithm description of our proposed scheme and the security model.

4.1. System Model

The system model for our proposed scheme is shown as Figure 1, which involves four types of entities, namely Trusted Authority (TA), Data Owner (DO), Data User (DU) and Cloud Service Provider (CSP). First, the DO extracts the keywords from each data file and builds a secure-searchable keyword index with an attribute-based access policy before outsourcing them into the CSP. The CSP is responsible for many services, such as data storage, computation and search. When a DU wants to issue a search query over encrypted data, he will generate a search trapdoor according to his interested keyword by using his private key and submit it to the CSP. Having received the trapdoor, the CSP attempts to check whether the specified search keyword matches with the index, without knowing the content of the encrypted data and the search keyword. Finally, the CSP returns the corresponding search results to the DU if and only if the attributes of the data user on the trapdoor satisfy the access policies of secure-searchable indexes, and the search trapdoor matches the keyword index. In addition, a TA is in charge of generating and distributing public keys and master keys.

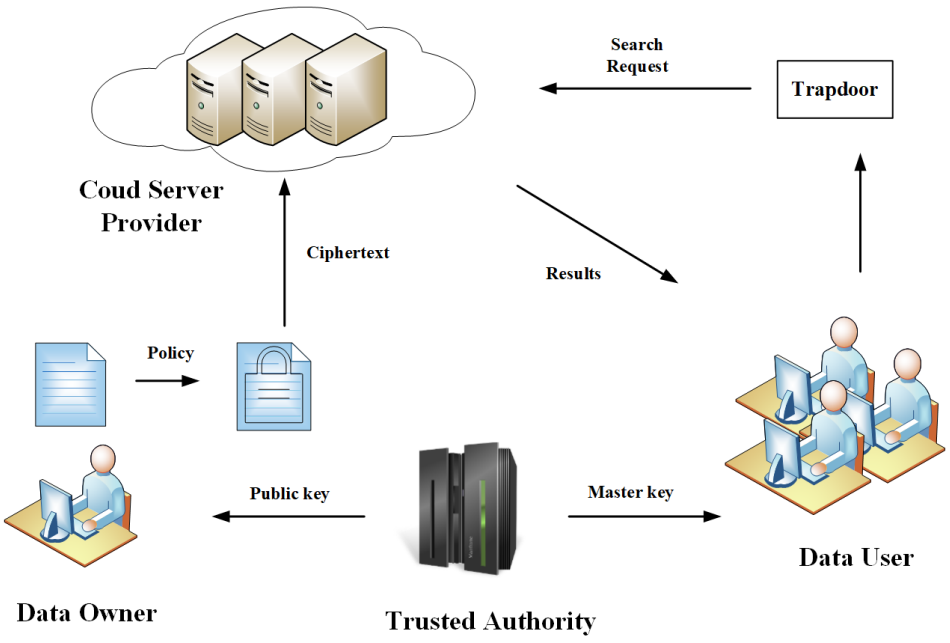

Figure 1. System Model of Our Scheme.

4.2. Algorithm Description of PEKS-CPABE Scheme

In this section, we present an overview of our proposed scheme, which is a tuple of five algorithms.

- **Setup**(1^{λ}): The setup algorithm is invoked by the TA, which takes as input the security parameter λ, and outputs the public parameter pp and the master private key msk;
- **KeyGen**(pp, msk, S_U): The key generation algorithm is invoked by the TA, which takes as input the public parameter pp, the master private key msk and the data user's attribute set S_U, and outputs the private key of the data user SK_U;
- **Encryption**(pp, T, w): The encryption algorithm is invoked by the DO, which takes as input the public parameter pp, the access tree T and the index keyword w, and outputs the encrypted index I_w;
- **TrapGen**(pp, SK_U, w'): The trapdoor generation algorithm is called the DU, which takes as input the public parameter pp, the private key of the data user SK_U and the search query for keyword w', and outputs the trapdoor for query keyword $T_{w'}$;

- **Search**$(pp, I_w, T_{w'}, S_U)$: The search query algorithm is called the CSP, which takes as input the public parameter pp, the encrypted index I_w, the trapdoor for query keyword $T_{w'}$ and the data user's attribute set S_U. If the attributes of the data user on the trapdoor satisfy the access policies of secure searchable indexes, and the search trapdoor matches the keyword index, the algorithm returns 1. Otherwise, it returns 0.

4.3. Security Model

To demonstrate the security of our scheme, we design a security game: indistinguishability security against an adaptive chosen keyword attack (IND-CKA) game.

The IND-CKA security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is defined as follows.

- **Init:** The adversary $\mathcal{A}$ chooses a challenging access tree T^*, which is sent to the challenger $\mathcal{C}$;
- **Setup:** The challenger $\mathcal{C}$ runs the **Setup**(1^λ) algorithms to generate public parameters pp and master key msk. It gives pp to adversary $\mathcal{A}$ and keeps master key msk;
- **Phase 1:** $\mathcal{A}$ can adaptively ask the simulator $\mathcal{B}$ for the trapdoors T_{w_i} of a series of keywords w_i, and issue private key query and trapdoor query as follows:

  Private key query: The adversary $\mathcal{A}$ can adaptively ask the challenger $\mathcal{C}$ for a group of private keys SK_U of some attributes. The challenger $\mathcal{C}$ runs the **KeyGen**(pp, msk, S_U) to generate a set of private keys SK_U, and sends to the adversary $\mathcal{A}$. The only restriction is that the responding private key does not satisfy T^*, and the $\mathcal{C}$ maintains a list L_{SK_U} of private keys;

  Trapdoor query: The adversary $\mathcal{A}$ can adaptively ask the challenger $\mathcal{C}$ for the trapdoors $T_{w'}$ of a series of keywords w'. The challenger $\mathcal{C}$ runs the **Trapdoor**(pp, SK_U, w') to generate the trapdoor, and sends to the adversary $\mathcal{A}$;
- **Challenge:** The adversary $\mathcal{A}$ sends the challenger $\mathcal{C}$ two keywords w_0, w_1 on which it wishes to be challenged. The challenger $\mathcal{C}$ randomly selects a bit $b \in \{0,1\}$ to encrypt and sends the encrypted keyword to the adversary $\mathcal{A}$;
- **Phase 2:** The adversary $\mathcal{A}$ can repeat the query in **Phase 1** and continue to ask for any keyword of his choice, except for the w_0,w_1;
- **Guess:** The adversary $\mathcal{A}$ outputs its guess of b' and wins the game if $b = b'$.

The advantage of an adversary $\mathcal{A}$ in winning the IND-CKA game is

$$Adv_{\mathcal{A}}^{IND-CKA}(\lambda) = |Pr[b = b'] - \tfrac{1}{2}|$$

Definition 1. *Our scheme is said to be IND-CKA secure if a probabilistic polynomial-time adversary wins the IND-CKA game with negligible advantage.*

5. PEKS-CPABE Construction

In this section, we give a concrete construction of the proposed PEKS-CPABE scheme.

- **Setup**(1^λ): The setup algorithm is called by the TA, which takes as input the security parameter λ, and outputs the public parameter pp and the master private key msk, which is processed as follows:

  — The TA first chooses two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p, which is a λ bit prime, g is the generator of $\mathbb{G}$, and selects a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$;

  — Let $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ and $H_2 : \{0,1\}^* \to \mathbb{G}$ be two secure hash functions;

  — Then, the TA selects a random element $\alpha, \beta \in \mathbb{Z}_p^*$ and sets

$$pp = (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^\alpha, g^\beta, H_1, H_2), msk = (\alpha, \beta);$$

- **KeyGen**(pp, msk, S_U): The key generation algorithm is invoked by the TA, which takes as input the public parameter pp, the master private key msk and the data user's attribute set S_U, and outputs the private key of the data user SK_U, which is processed as follows:

  — The TA randomly selects $r \in \mathbb{Z}_p^*$ and computes $K = g^{\frac{\alpha+r}{\beta}}, K' = g^r$;

  — For each attribute $at_j \in S_U$, compute $K_{at_j} = K' g^{H_1(at_j)}$;

  — Set the private key of the data user as

$$SK_U = (K, \{(K_{at_j}) | \forall at_j \in S_U\})$$

- **Encryption**(pp, T, w): The encryption algorithm is invoked by the DO, which takes as input the public parameter pp, the access tree T and the index keyword w, and outputs the encrypted index I_w, which is processed as follows:

  — The DO first computes $I_1 = \hat{e}(g,g)^{\alpha s} \hat{e}(g^{s\beta}, H_2(w))$, $I_2 = g^{s\beta}$;

  — For each node x in the access tree T, the DO chooses a d_x degree polynomial q_x in a top-down manner, where $d_x = k_x - 1$, where k_x is the threshold value of node x;

  — For the root node $root$ of access tree T, the DO randomly picks up a secret key $s \in \mathbb{Z}_p^*$ and sets $q_{root}(0) = s$. Then, the DO randomly chooses d_{root} other points of to define the polynomial q_{root} completely;

  — For other non-root node x, the DO sets $q_x(0) = q_{parent(x)}(index(x))$, and randomly chooses d_x other points to define the polynomial q_x completely;

  — Finally, for each node $x \in lvs(T)$, compute $A_x = g^{q_x(0)}$ and $B_x = g^{H_1(att(x))q_x(0)}$. The encrypted index is given as

$$I_w = (T, I_1, I_2, \{(A_x, B_x) | \forall x \in lvs(T)\})$$

- **TrapGen**(pp, SK_U, w'): The trapdoor generation algorithm is called the DU, which takes as input the public parameter pp, the private key of the data user SK_U and the search query for keyword w', and outputs the trapdoor $T_{w'}$, which is processed as follows:

  — The DU computes $T_0 = K_1 H_2(w') = g^{\frac{\alpha+r}{\beta}} H_2(w')$;

  — For each $at_j \in S_U$, compute $T_{at_j} = K_{at_j}$. The trapdoor is given as

$$T_{w'} = (T_0, \{(T_{at_j}) | \forall at_j \in S_U\})$$

- **Search**$(pp, I_w, T_{w'}, S_U)$: The search query algorithm is called by the CSP, which takes as input the public parameter pp, the encrypted index I_w, the trapdoor for query keyword $T_{w'}$ and the data user's attribute set S_U. The CSP selects a data user's attribute set S_U that satisfies the access tree T contained the encrypted index. If S_U does not exist, the algorithm returns 0. Otherwise, it computes as follows:

  — If the node x is a leaf node in the T, for each attribute $at_j \in S_U$, then

$$E_x = \frac{\hat{e}(T_{at_j}, A_x)}{\hat{e}(g, B_x)} = \frac{\hat{e}(g^r g^{H_1(at_j)}, g^{q_x(0)})}{\hat{e}(g, g^{H_1(att(x))q_x(0)})} = \hat{e}(g,g)^{r q_x(0)}, att(x) = at_j, x \in lvs(T).$$

— If the node x is a non-leaf node in the T, we get the E_x by computing $E_{x'}$ in a recursive manner, where x' is the children nodes of x. Let S_x represent an arbitrary k_x set of children nodes x, if no such set exists, $E_{x'} = \perp$; otherwise, compute as follows

$$\begin{aligned} E_x &= \prod_{x' \in S_x} E_{x'}^{\Delta_{i,S'_x(0)}} \\ &= \prod_{x' \in S_x} (\hat{e}(g,g)^{rq_{x'}(0)})^{\Delta_{i,S'_x(0)}} \\ &= \prod_{x' \in S_x} (\hat{e}(g,g)^{rq_{parent(x')}(index(x'))})^{\Delta_{i,S'_x(0)}} \\ &= \prod_{x' \in S_x} (\hat{e}(g,g)^{rq_x(i)})^{\Delta_{i,S'_x(0)}} \\ &= \hat{e}(g,g)^{rq_x(0)} \end{aligned} \tag{1}$$

where $i = index(x'), S'_x = \{index(x') : x' \in S_x\}$;

— If x is a root node in T, $E_{root} = \hat{e}(g,g)^{rq_{root}(0)} = \hat{e}(g,g)^{rs}$. Finally, if $\hat{e}(I_2, T_0) = E_{root} I_1$, the search algorithm returns 1; otherwise, it returns 0.

Correctness. I_1 and I_2 are generated by the encryption algorithm, and T_0 is generated in the trapdoor generation algorithm. The proposed PEKS-CPABE scheme is correct when the following equation holds.

$$\begin{aligned} I_1 &= \frac{\hat{e}(I_2, T_0)}{E_{root}} \\ &= \frac{\hat{e}(g^{s\beta}, g^{\frac{\alpha+r}{\beta}} H_2(w'))}{\hat{e}(g,g)^{rs}} \\ &= \frac{\hat{e}(g,g)^{s(\alpha+r)} \hat{e}(g^{s\beta}, H_2(w'))}{\hat{e}(g,g)^{rs}} \\ &= \hat{e}(g,g)^{\alpha s} \hat{e}(g^{s\beta}, H_2(w')). \end{aligned} \tag{2}$$

If the query keyword w' matches the encrypted keyword index, namely $w = w'$, then we have the equation $\frac{\hat{e}(I_2,T_0)}{E_{root}} = I_1$ hold.

6. Security Proof

In this section, we give the security proof of the proposed PEKS-CPABE scheme. Based on the aforementioned security model, we provide a detailed security proof that our scheme achieves selective indistinguishability security against an adaptive chosen keyword attack.

Theorem 1. *PEKS-CPABE scheme achieves IND-CKA security on the condition that DBDH problem is intractable.*

Proof. Assume that $\mathcal{A}$ is an adversary that can break our proposed scheme, then we can construct a simulator $\mathcal{B}$, and the goal is to distinguish between the DBDH tuple $(g^a, g^b, g^c, Z = \hat{e}(g,g)^{abc})$ and a random tuple $(g^a, g^b, g^c, Z = \hat{e}(g,g)^z))$ where $a, b, c, z \in \mathbb{Z}_p^*$. The IND-CKA security game between the adversary $\mathcal{A}$ and the simulator $\mathcal{B}$ is conducted as follows.

- **Init:** The adversary $\mathcal{A}$ first chooses an access tree T^* to be challenged, which is sent to the simulator $\mathcal{B}$;
- **Setup:** The simulator $\mathcal{B}$ randomly chooses $\alpha, \beta \in \mathbb{Z}_p^*$, and computes g^α,g^β, $\hat{e}(g,g)^\alpha$. Then, $\mathcal{B}$ returns the public parameter $pp = (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^\alpha, g^\beta, H_1, H_2)$ which is sent to adversary

$\mathcal{A}$. $H_2(w_i)$ is simulated as follows. If the w_i has not been queried before, the simulator $\mathcal{B}$ randomly chooses $\rho_i \in \mathbb{Z}_p^*$, adds (w_i, ρ_i) to the list O_{H_2} and outputs g^{ρ_i}; otherwise, the simulator $\mathcal{B}$ returns g^{ρ_i} by searching ρ_i from O_{H_2};

- **Phase 1:** $\mathcal{A}$ can adaptively ask the simulator $\mathcal{B}$ for the trapdoors T_{w_i} of a series of keywords w_i, and issue the O_{KeyGen} and $O_{Trapdoor}$ oracles as follows.

  O_{KeyGen}: The adversary $\mathcal{A}$ can adaptively ask the simulator $\mathcal{B}$ for a group of private keys $SK_{U_1}, \cdots, SK_{U_n}$ of some attributes $S_{U_1}, \cdots, S_{U_n}$. The attribute sets embedded into corresponding private keys do not satisfy T^*. The simulator $\mathcal{B}$ randomly selects $r \in \mathbb{Z}_p^*$ and computes $K = g^{\frac{\alpha+r}{\beta}}$. For each attribute $at_j \in S_U$, compute $K_{at_j} = g^r g^{H_1(at_j)}$. The simulator $\mathcal{B}$ sends the private key $SK_U = (K, K_{at_j})$ to $\mathcal{A}$, and maintains a list L_{SK_U} of private keys.

  $O_{Trapdoor}$: The simulator $\mathcal{B}$ first searches the O_{KeyGen} oracle to obtain the secret key as

  $(K, \{(K_{at_j}, K'_{at_j}) | \forall at_j \in S_U\})$. Then, $\mathcal{B}$ computes $T_0 = KH_2(w_i) = g^{\frac{\alpha+r^*}{\beta}} g^{\rho_i}$. For each $at_j \in S_U$, compute $T_{at_j} = K_{at_j}$. Finally, $\mathcal{B}$ generates the trapdoor as $T_{w_i} = (T_0, \{(T_{at_j}) | \forall at_j \in S_U\})$, and $\mathcal{B}$ sends T_{w_i} to the adversary $\mathcal{A}$;
- **Challenge:** The adversary $\mathcal{A}$ sends the simulator $\mathcal{B}$ two keywords w_0, w_1 on which it wishes to be challenged. The simulator $\mathcal{B}$ randomly selects a bit $b \in \{0, 1\}$ to encrypt and generates the encrypted keyword index as follows: $I_1^* = Z\hat{e}(g^{a\beta}, H_2(w_b))$, $I_2^* = g^{a\beta}$, $A_x^* = g^{q_x(0)}$,$B_x^* = g^{H_1(att(x))q_x(0)}, \forall x \in lvs(T^*)$. Namely, $I_{w_b} = (T^*, I_1^*, I_2^*, \{(A_x^*, B_x^*) | \forall x \in lvs(T^*)\})$ where $a \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sends I_{w_b} to the adversary $\mathcal{A}$;
- **Phase 2:** The adversary $\mathcal{A}$ can repeat query in **Phase 1** and continue to ask for any keyword of his choice, except for the w_0,w_1;
- **Guess:** The adversary $\mathcal{A}$ outputs its guess b' of b. If $b' = b$, $\mathcal{B}$ outputs 1; otherwise $\mathcal{B}$ randomly outputs 0 or 1.

  There are two conditions, as follows:

1. If $Z = \hat{e}(g, g)^{abc}$, a valid ciphertext I_{w_b} is given to the adversary $\mathcal{A}$, and the adversary $\mathcal{A}$ has an advantage ϵ to win this game.

   The ciphertexts I_{w_b} is valid. $I_1^* = Z\hat{e}(g^{a\beta}, H_2(w_b)) = \hat{e}(g, g)^{abc}\hat{e}(g^{a\beta}, H_2(w_b))$, $I_2^* = g^{a\beta}$. Since α, s are randomly selected, $a, b, c, \in \mathbb{Z}_p^*$, let $a = s, bc = \alpha$, I_1^*, I_2^* can be denoted as $I_1^* = \hat{e}(g, g)^{s\alpha}\hat{e}(g^{s\beta}, H_2(w_b))$, $I_2^* = g^{s\beta}$. This means I_{w_b} is the valid ciphertext. Through ϵ is the advantage that the adversary $\mathcal{A}$ outputs a correct guess, therefore we have that $Pr[\mathcal{B}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^{abc}) = 1] = \frac{1}{2} + \epsilon$;
2. If $Z \neq \hat{e}(g, g)^{abc}$, I_{w_b} is a random ciphertext. $\mathcal{A}$ has nothing to do with the guess, so the adversary $\mathcal{A}$ cannot acquire any advantage in this IND-CKA security game but a random guess. Therefore, we have $Pr[\mathcal{B}(g, g^a, g^b, g^c, Z = \hat{e}(g, g)^z) = 1] = \frac{1}{2}$.

Finally, the overall advantage that $\mathcal{B}$ can solve the DBDH problem in the IND-CKA security is as follows: $Adv_{\mathcal{B}}^{IND-CKA}(\lambda) = \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$

Thus, we have a conclusion that if the probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking IND-CKA security, then we can construct a simulator $\mathcal{B}$ to solve the DBDH problem with the non-negligible advantage $\frac{\epsilon}{2}$. From the above analysis, we know that our proposed PEKS-CPABE scheme is IND-CKA secure on the condition that the DBDH problem is intractable. This completes the proof.

□

7. Performance Analysis

In this section, we show the performance analysis of our proposed PEKS-CPABE scheme in terms of theoretical analysis and experimental analysis, and further compare our proposed scheme with

the state-of-the-art CP-ABKS [22] scheme and CP-ABSE [23] scheme. Table 1 shows the notations of the performance analysis. For the theoretical analysis, we mainly focus on the computation cost and storage cost. For convenience of comparison, we mainly consider several time-consuming operations, as follows: the bilinear pairing operation *Pair* mapping two elements in group $\mathbb{G}$ to group $\mathbb{G}_T$, the hash function operation H mapping the arbitrary string to an element in group $\mathbb{G}$, a modular exponentiation operation $E_{\mathbb{G}}$ in $\mathbb{G}$ and a modular exponentiation operation $E_{\mathbb{G}_T}$ in $\mathbb{G}_T$. We ignore the multiplication operations and hash operations mapping the arbitrary string to an element in group $\mathbb{Z}_p^*$ because of the increased efficiency.

Table 1. Notations for performance analysis.

Notation	Description
$E_{\mathbb{G}}$	The time of a modular exponentiation in $\mathbb{G}$
$E_{\mathbb{G}_T}$	The time of a modular exponentiation in $\mathbb{G}_T$
$Pair$	The time of a bilinear pairing
H	The time of hash operation mapping a bit-string to an element in $\mathbb{G}$
$\lvert\mathbb{G}\rvert$	The number of elements in $\mathbb{G}$
$\lvert\mathbb{G}_T\rvert$	The number of elements in $\mathbb{G}_T$
$\lvert\mathbb{Z}_p^*\rvert$	The number of elements in $\mathbb{Z}_p^*$
S	The number of a data user's attribute
N	The number of attributes that are involved in a data owner's access control policy

7.1. Theoretical Analysis

In Table 2, we evaluate the computation cost of **KeyGen** algorithm, **Encryption** algorithm, **TrapGen** algorithm and **Search** algorithm under the same access control policy tree, respectively. We observe that our scheme is much more efficient than the CP-ABKS [22] scheme and CP-ABSE [23] scheme in the **KeyGen** algorithm and **Encryption** algorithm. Although we notice that our scheme has a higher computational overhead than the CP-ABSE [23] scheme in the **TrapGen** algorithm, the hash function operation time is minimal. In the **Search** algorithm, our scheme performs similarly to that of CP-ABSE [23] scheme, and the PEKS-CPABE scheme has a better performance than the CP-ABKS [22] scheme.

In Table 3, we show the storage cost comparison. We observe that the storage cost of our scheme outperforms the CP-ABKS [22] scheme and CP-ABSE [23] scheme in the **KeyGen** algorithm. In the execution **Encryption** algorithm and **TrapGen** algorithm, the storage cost of our scheme is the same as that of the CP-ABSE [23] scheme, but much less than that of the CP-ABKS [22] scheme. Although the CP-ABSE scheme has better search performance than the CP-ABKS scheme, the cost of **KeyGen** algorithm and **Encryption** algorithm bring in a much higher overhead. Our scheme solves this problem at the same time, with greater efficiency, therefore, with respect to theoretical analysis, our scheme is acceptable in the cloud.

Table 2. The comparison of computation cost.

Algorithm	CP-ABKS [22]	CP-ABSE [23]	PEKS-CPABE (Ours)
KeyGen	$(2S+2)E_{\mathbb{G}} + SH$	$(2S+3)E_{\mathbb{G}} + SH$	$(S+2)E_{\mathbb{G}}$
Encryption	$(2N+4)E_{\mathbb{G}} + NH$	$(2N+2)E_{\mathbb{G}} + E_{\mathbb{G}_T} + 2Pair + NH$	$(2N+1)E_{\mathbb{G}} + E_{\mathbb{G}_T} + 2Pair$
TrapGen	$(2S+4)E_{\mathbb{G}}$	$E_{\mathbb{G}}$	$E_{\mathbb{G}} + H$
Search	$(2N+3)Pair + NE_{\mathbb{G}_T}$	$(2N+1)Pair + NE_{\mathbb{G}_T}$	$(2N+1)Pair + NE_{\mathbb{G}_T}$

Table 3. The comparison of storage cost.

Algorithm	CP-ABKS [22]	CP-ABSE [23]	PEKS-CPABE (Ours)
KeyGen	$(2S+1)log\|\mathbb{G}\|$	$(2S+2)log\|\mathbb{G}\|$	$(S+1)log\|\mathbb{G}\|$
Encryption	$(2N+3)log\|\mathbb{G}\|$	$(2N+1)log\|\mathbb{G}\|+log\|\mathbb{G}_T\|$	$(2N+1)log\|\mathbb{G}\|+log\|\mathbb{G}_T\|$
TrapGen	$(2S+3)log\|\mathbb{G}\|$	$(2S+1)log\|\mathbb{G}\|$	$(S+1)log\|\mathbb{G}\|$

7.2. Experimental Analysis

To evaluate the actual performance of our scheme, we implement CP-ABKS [22], CP-ABSE [23] and our scheme using Java language based on the Java Pairing Based Cryptography Library (JPBC) [31]. Our experimental platform is based on a Windows 10 server with Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz and 8.00GB RAM. The running environment of our experiment is Java Runtime Environment 1.8 (JRE1.8). In our experiment, we instantiated the bilinear map with Type A: $y^2 = x^3 + x$. For comparison convenience, we set the access policy tree as "AND" access tree "at_1 AND at_2 AND, $\cdots$, AND at_N", and the number of a data user's attributes S is equal to the number of attributes that are involved in the access policy, from 1 to 50 with step length 10, namely, $N = S \in [1, 50]$. We conduct each experiment many times to obtain the average execution time under the same access control policy. Figure 2 shows the performance comparison of various schemes.

As shown in Figure 2a, we can find that the time of key generation in our scheme is more efficient than the CP-ABKS scheme and CP-ABSE scheme. For example, when setting $N = 50$, our scheme needs 1061 ms to generate keys, however, both CP-ABKS scheme and CP-ABSE scheme need 4333 and 4314 ms, respectively. As illustrated in Figure 2b, in our scheme, the time cost to generate ciphertexts is the lowest out of the three schemes. For example, when setting $N = 50$, our scheme needs 3074 ms to generate ciphertexts, however, both the CP-ABKS scheme and CP-ABSE scheme need 4375 and 4277 ms, respectively. From Figure 2c,d, we can show that our scheme has the better search performance. In the execution of trapdoor generation and search, our scheme and the CP-ABSE scheme outperform the CP-ABKS scheme. Therefore, our scheme is acceptable in practice and suitable for the cloud.

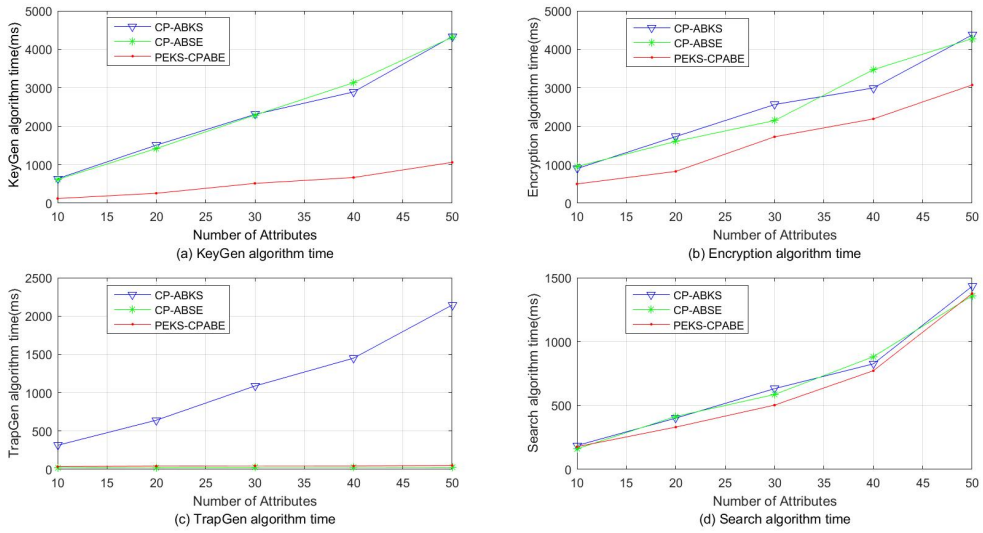

Figure 2. The performance comparison of various schemes.

8. Conclusions

In this paper, we propose a privacy-preserving and efficient public key encryption with a keyword search scheme based on CP-ABE to support both fine-grained access control and keyword search over encrypted data simultaneously. Then, we show that our scheme achieves selective indistinguishability

security against an adaptive chosen keyword attack on the condition that the DBDH problem is intractable. Meanwhile, we also analyzed the performance of our proposed scheme from the aspects of theoretical analysis and experimental analysis. At last, the experimental results further demonstrate that our scheme performs better than the CP-ABKS scheme proposed in [22] and CP-ABSE scheme proposed in [23]. Besides, our work only considers single keyword search. For the part of the future work, we try to enhance the search functionality and further explore an attribute-based multi-keyword search.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; software, Y.Z.; validation, S.Z. and L.W.; formal analysis, S.Z. and L.W.; investigation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, S.Z. and L.W; project administration, L.W.; funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (2017YFB0803001) and the National Natural Science Foundation of China (NSFC) (No. 61972050).

Acknowledgments: In addition, we gratefully acknowledge Jing Li and Jie Cheng from the State Grid Information & Telecommunication Branch for their constructive criticism and suggestions in the revision of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PEKS	Public Key Encryption with Keyword Search
ABE	Attribute-based Encryption
CP-ABE	Ciphertext-policy Attribute-based Encryption
KP-ABE	Key-policy Attribute-based Encryption
SE	Searchable Encryption
SSE	Symmetric Searchable Encryption
TA	Trusted Authority
DO	Data Owner
DU	Data User
CSP	Cloud Service Provider

References

1. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
2. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
3. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
4. Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. Comput. Secur.* **2011**, *19*, 895–934. [CrossRef]
5. Kamara, S.; Papamanthou, C.; Roeder, T. Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 965–976.
6. Kamara, S.; Papamanthou, C. Parallel and dynamic searchable symmetric encryption. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 258–274.
7. Li, H.; Yang, Y.; Dai, Y.; Bai, J.; Yu, S.; Xiang, Y. Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. *IEEE Trans. Cloud Comput.* **2017**. [CrossRef]

8. Ghareh Chamani, J.; Papadopoulos, D.; Papamanthou, C.; Jalili, R. New constructions for forward and backward private symmetric searchable encryption. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1038–1055.
9. Sun, S.F.; Yuan, X.; Liu, J.K.; Steinfeld, R.; Sakzad, A.; Vo, V.; Nepal, S. Practical backward-secure searchable encryption from symmetric puncturable encryption. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 763–780.
10. Wang, K.; Dong, X.; Shen, J.; Cao, Z. An Effective Verifiable Symmetric Searchable Encryption Scheme in Cloud Computing. In Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City, Kuala, Malaysia, 18 April 2019; pp. 98–102.
11. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 506–522.
12. Abdalla, M.; Bellare, M.; Catalano, D.; Kiltz, E.; Kohno, T.; Lange, T.; Malone-Lee, J.; Neven, G.; Paillier, P.; Shi, H. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 205–222.
13. Park, D.J.; Kim, K.; Lee, P.J. Public key encryption with conjunctive field keyword search. In *International Workshop on Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 73–86.
14. Boneh, D.; Waters, B. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 535–554.
15. Baek, J.; Safavi-Naini, R.; Susilo, W. Public key encryption with keyword search revisited. In *International Conference on Computational Science and Its Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1249–1259.
16. Tang, Q.; Chen, L. Public-key encryption with registered keyword search. In *European Public Key Infrastructure Workshop*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 163–178.
17. Fang, L.; Susilo, W.; Ge, C.; Wang, J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.* **2013**, *238*, 221–241. [CrossRef]
18. Lv, Z.; Hong, C.; Zhang, M.; Feng, D. Expressive and secure searchable encryption in the public key setting. In *International Conference on Information Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 364–376.
19. Huang, Q.; Li, H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Inf. Sci.* **2017**, *403*, 1–14. [CrossRef]
20. Zhang, Y.; Wang, Y.; Li, Y. Searchable Public Key Encryption Supporting Semantic Multi-Keywords Search. *IEEE Access* **2019**, *7*, 122078–122090. [CrossRef]
21. Zhou, Y.; Li, N.; Tian, Y.; An, D.; Wang, L. Public Key Encryption with Keyword Search in Cloud: A Survey. *Entropy* **2020**, *22*, 421. [CrossRef]
22. Zheng, Q.; Xu, S.; Ateniese, G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 522–530.
23. Yin, H.; Zhang, J.; Xiong, Y.; Ou, L.; Li, F.; Liao, S.; Li, K. CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme. *IEEE Access* **2019**, *7*, 5682–5694. [CrossRef]
24. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 457–473.
25. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
26. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
27. Dong, Q.; Guan, Z.; Chen, Z. Attribute-based keyword search efficiency enhancement via an online/offline approach. In Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, Australia, 14–17 December 2015; pp. 298–305.
28. Li, J.; Lin, X.; Zhang, Y.; Han, J. KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Trans. Serv. Comput.* **2016**, *10*, 715–725. [CrossRef]

29. Sun, W.; Yu, S.; Lou, W.; Hou, Y.T.; Li, H. Protecting Your Right: Verifiable Attribute-Based Keyword Search with Fine-Grained Owner-Enforced Search Authorization in the Cloud. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1187–1198. [CrossRef]
30. Qiu, S.; Liu, J.; Shi, Y.; Zhang, R. Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. *Sci. China Inf. Sci.* **2017**, *60*, 052105. [CrossRef]
31. De Caro, A.; Iovino, V. jPBC: Java pairing based cryptography. In Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC), Corfu, Greece, 28 June 28–1 July 2011; pp. 850–855.