What is the buzz term in the current field of computer science?

A. Cloud Computing
B. Grid Computing
C. Distributed Computing
D. Parallel Computing

What is the buzz term in the current field of computer science?

A. Cloud Computing
B. Grid Computing
C. Distributed Computing
D. Parallel Computing

Cloud Computing is a recent trend in IT that moves computing and data away from desktop and portable PCs into remote large data centers.

Cloud Computing can provide three kinds of services:

Infrastructure-as-a-Service (IaaS): Such as Amazon's Elastic Compute Cloud (EC2)

Platform-as-a-Service (PaaS): Such as Google App Engine

Software-as-a-Service (SaaS): Such as Google Docs

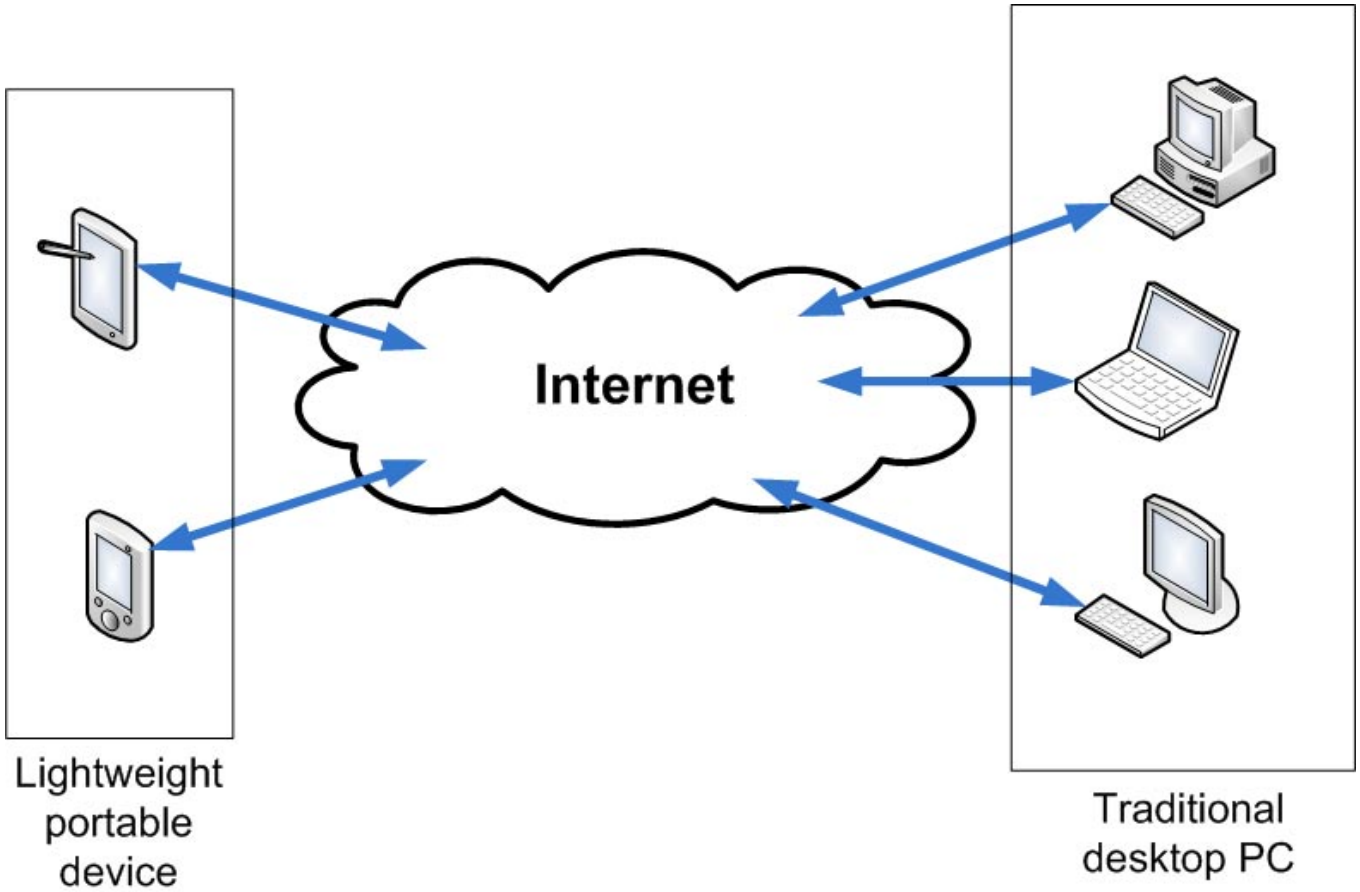


Figure: Benefit of Cloud Computing.

However, there are some security problems in cloud computing. For example, when users store the private data in the cloud computing, how can they protect the secrecy of the data without sacrificing some functionalities, such as searchability?

However, there are some security problems in cloud computing. For example, when users store the private data in the cloud computing, how can they protect the secrecy of the data without sacrificing some functionalities, such as searchability?

Note that the ACL (access control list) based approach is ruled out immediately, since it is always assumed that the data center is fully trusted, while it is semi-trusted in the cloud computing.

Secure Storage in the Cloud Computing

Reporter: Jun Shao

January 26, 2010

Security Requirements

One Creator vs. One Searcher

Multi-Creator vs. One Searcher

Multi-Creator vs. Multi-Searcher

Security requirements

Document confidentiality The document can only be accessed by the authorized user.

Security requirements

Document confidentiality The document can only be accessed by the authorized user.

Inference resistance The unauthorized user cannot decide which two keywords in one document.

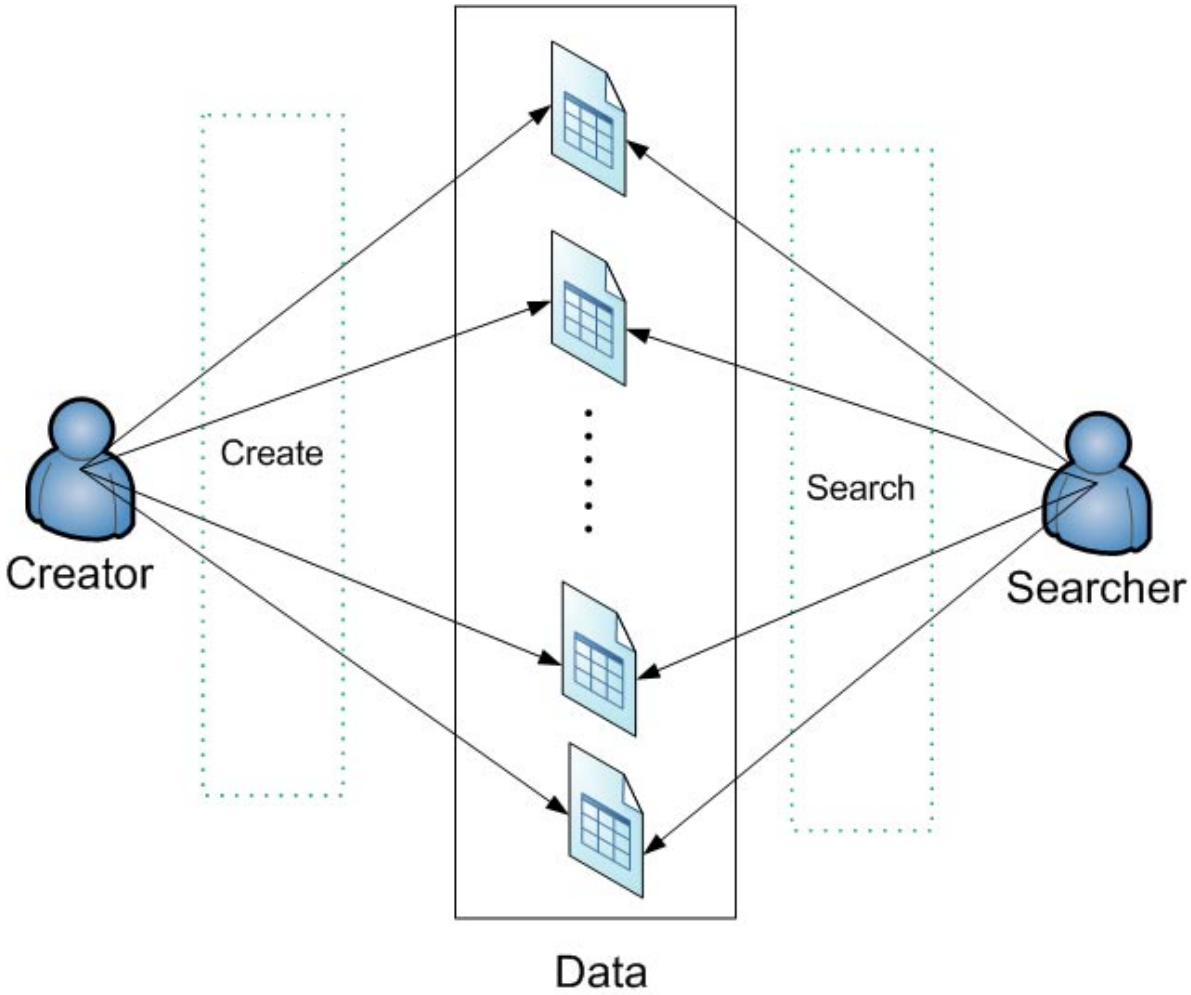
Security requirements

Document confidentiality The document can only be accessed by the authorized user.

Inference resistance The unauthorized user cannot decide which two keywords in one document.

Policy privacy The unauthorized user cannot decide the access policy of documents.

One Creator vs. One Searcher



One Creator vs. One Searcher

The existing solutions are usually based on symmetric encryption with keyword search (SEKS), which is proposed by Song, Wagner, and Perrig.

D. Song, D. Wagner, and A. Perrig.
Practical techniques for searches on encrypted data.
In *S & P 2000*, pages 44–55, 2000.

Symmetric encryption with keyword search

Symmetric encryption with keyword search

- a kind of symmetric encryption,
- the data provider encrypts the data according to the keyword,
- the resulting ciphertext can only be decrypted by the key associated to related keyword.

Basic knowledge

Algorithms in symmetric encryption with keyword search

$\texttt{SEKS.KeyGen}(1^\ell) \to sk$: output the secret key sk

$\texttt{SEKS.Trapdoor}(sk, w) \to d$: output the decryption key d associated to the keyword w.

$\texttt{SEKS.Enc}(m, sk, w) \to C$: output the ciphertext associated to the keyword w.

$\texttt{SEKS.Dec}(d, C) \to m$: output the plaintext m.

Description of the system

The secret key of the underlying symmetric encryption with keyword search is shared between the creator and the searcher.

Create: The creator encrypts the document as follows, and sends the resulting ciphertexts to the data center.

$$\text{Encrypted data}||\text{encrypted keywords}$$
$$C_0||(C_1||\cdots||\cdots)$$

where $C_0 = \mathtt{E}_{sk}(m)$, and $C_i = \mathtt{SEKS.Enc}(Y, sk, w_i)$ $(i = 1, \cdots)$, $\mathtt{E}$ is a traditional symmetric encryption, Y is a label meaning "yes", and w_i's are the keywords the document m contains.

Description of the System

Query: The searcher generates the query key d

$$d = \texttt{SEKS.Trapdoor}(sk, w),$$

and sends it to the server.
The server checks

$$Y \stackrel{?}{=} \texttt{SEKS.Dec}(d, C_i) \quad (i \in \{1, \cdots\})$$

Description of the System

Update:

- Adding, the same as Create.
- Deleting, simply find the entry and delete it.
- Modifying, first deleting the old one, and then adding a new one.

Security Analysis

Document confidentiality Due to the security of symmetric encryption $\mathtt{E}$, the one without knowing the secret key cannot get m.

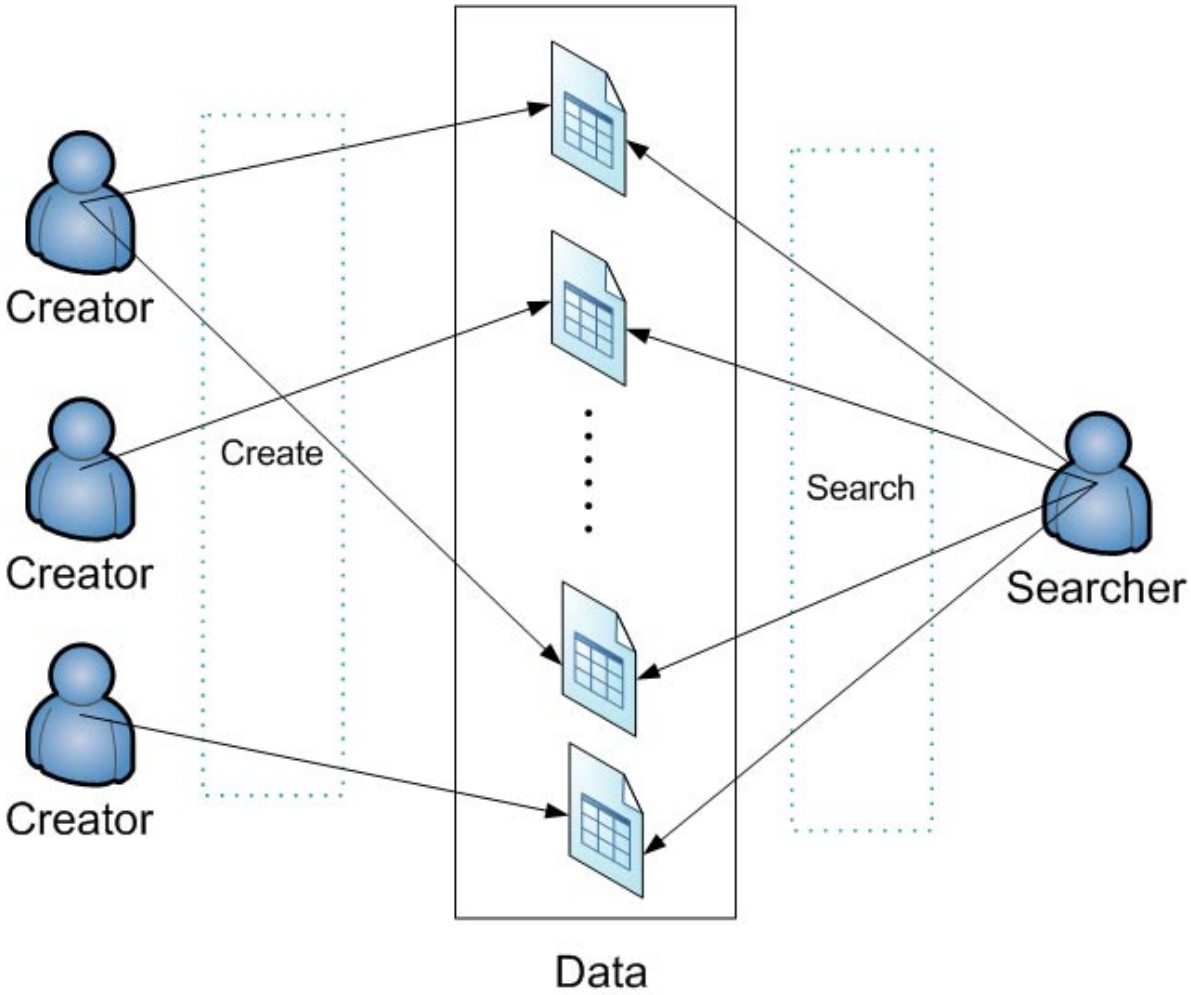
Inference resistance Due to the security of symmetric encryption with keyword search $\mathtt{SKKS}$, the one without knowing the secret key cannot relate d to the real keyword.

Policy privacy No such security.

Limitations

- Sequential scan, time complexity: $O(n)$, n is the total number of entries.
- Once query key is related to the real keyword, the adversary can check whether a specific document (even the new document) contains this keyword.

Multi-Creator vs. One Searcher



Multi-Creator vs. One Searcher

Most of the existing solutions are based on public key encryption with keyword search (PKEKS), which is proposed by Boneh, Crescenzo, Ostrovsky, and Persiano.

D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano.
Public key encryption with keyword search.
In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522, 2004.

Public key encryption with keyword search

Public key encryption with keyword search

- a kind of public key encryption,
- the data provider encrypts the data according to the keyword,
- the resulting ciphertext can only be decrypted by the key associated to related keyword.

Basic knowledge

Algorithms in public key encryption with keyword search

$\texttt{PKEKS.KeyGen}(1^\ell) \to sk$: output the public/secret key pair (pk, sk)

$\texttt{PKEKS.Trapdoor}(sk, w) \to d$: output the decryption key d associated to the keyword w.

$\texttt{PKEKS.Enc}(m, pk, w) \to C$: output the ciphertext associated to the keyword w.

$\texttt{PKEKS.Dec}(d, C) \to m$: output the plaintext m.

Description of the system

The public key of the underlying public key encryption with keyword search is shared among the creators, and the secret key is kept by the searcher.

Create: The creator encrypts the document as follows, and sends the resulting ciphertexts to the data center.

$$\text{Encrypted data}||\text{encrypted keywords}$$
$$C_0||(C_1||\cdots||\cdots)$$

where $C_0 = \mathtt{Enc}_{pk}(m)$, and $C_i = \mathtt{PKEKS.Enc}(Y, pk, w_i)$ $(i = 1, \cdots)$, $\mathtt{Enc}$ is a traditional public key encryption, Y is a label meaning "yes", and w_i's are the keywords the document m contains.

Description of the System

Query: The searcher generates the query key d

$$d = \texttt{PKEKS.Trapdoor}(sk, w),$$

and sends it to the server.
The server checks

$$Y \stackrel{?}{=} \texttt{PKEKS.Dec}(d, C_i) \quad (i \in \{1, \cdots\})$$

Description of the System

Update:

- Adding, the same as Create.
- Deleting, simply find the entry and delete it.
- Modifying, first deleting the old one, and then adding a new one.

Security Analysis

Document confidentiality Due to the security of public key encryption `Enc`, the one without knowing the secret key cannot get m.

Inference resistance Due to the security of public key encryption with keyword search `PKEKS`, the one without knowing the public/secret key pair cannot relate t to the real keyword.

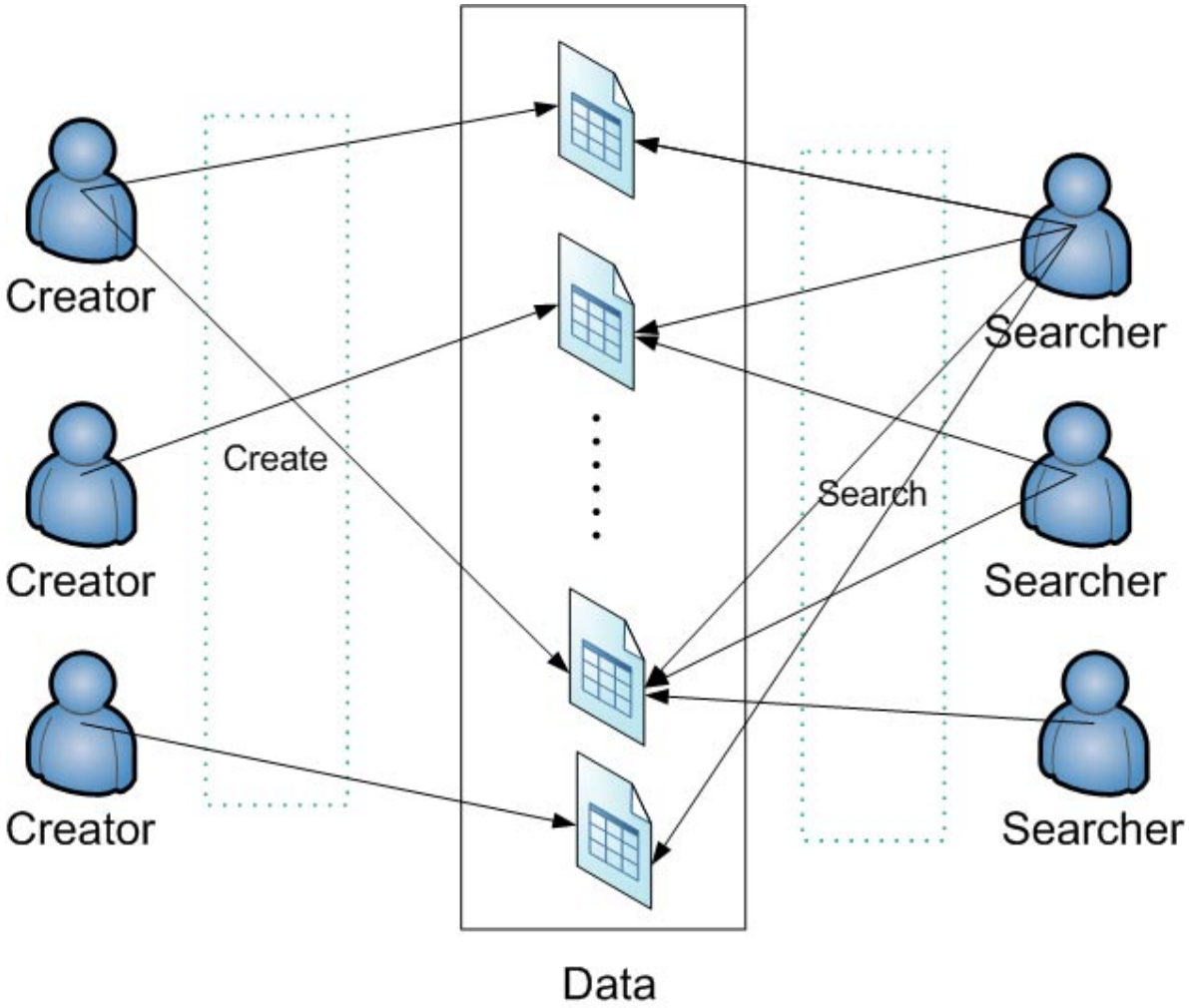
Policy privacy No such security.

Limitations

- Sequential scan, time complexity: $O(n)$, n is the total number of entries.
- Once one creator and the server collude, they can relate d to the keyword by check

$$\texttt{PKEKS.Dec}(d, \texttt{PKEKS.Enc}(Y, pk, w)) \stackrel{?}{=} Y.$$

Multi-Creator vs. Multi-Searcher



Multi-Creator vs. Multi-Searcher

We propose the first system dealing with the case of Multi-Creator vs. Multi-Searcher. Our proposal is based on predication encryption (PE), which is proposed by Katz, Sahai, and Waters.

J. Katz, A. Sahai, and B. Waters.
Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products
In *EUROCRYPT 2008*, volume 4905 of *LNCS*, pages 146–162, 2004.

Predicate encryption

Predicate encryption

- an extension of public key encryption with keyword search
- the encryptor encrypts the data according to the access policy
- the resulting ciphertext can only be decrypted by the decryptor whose attributes satisfy the access policy.

For example, the access policy is (only the reviewer from the Department of Computer Science and Engineering can access the document), and the attributes are ((role=reviewer AND dept.=CSE) OR (role=author AND dept.=EE)).

Basic Knowledge

Algorithms in predicate encryption

$\mathtt{PE.Setup}(1^\ell) \to (mpk, msk, para)$ performed by a third trusted party (different from the encryptor and the decryptor).

$\mathtt{PE.KeyGen}(msk, \mathbb{A}) \to sk$ performed by the third trusted party.

$\mathtt{PE.Dele}(sk_1, \tilde{\mathbb{A}}) \to sk_2$ performed by the one holding sk_1 with attributes $\mathbb{A}$, $\tilde{\mathbb{A}} \subseteq \mathbb{A}$.

$\mathtt{PE.Enc}(m, mpk, \mathcal{P}) \to C$ performed by the encryptor.

$\mathtt{PE.Dec}(sk, C) \to m$ performed by the decryptor, $f(\mathcal{P}, \mathbb{A}) = 1$

Basic Knowledge

Security properties of predicate encryption

Payload-hiding The one holding attributes $\mathbb{A}$ that $f(\mathcal{P}, \mathbb{A}) \neq 1$ cannot access the plaintext of the ciphertext whose access policy is $\mathcal{P}$.

Policy-hiding The one without *msk* cannot figure out the access policy of a document which is not created by him/her.

Our system

In our system, we have the following kinds of entities: creators, searchers, the server in cloud computing, a central authority, and an indexing server.

Our system

In our system, we have the following kinds of entities: creators, searchers, the server in cloud computing, a central authority, and an indexing server.

Trust levels:

- *Fully-trusted*: Do not launch any kind of attacks. (the central authority and the indexing server)
- *Honest-but-curious*: Only launch passive attacks. (the server in cloud computing)
- *Can-be-malicious-and-curious*: Can launch both passive and active attacks in arbitrary ways. (the creators and the searchers, however, the creators are assumed to not generate the dump documents.)

Our system (overview)

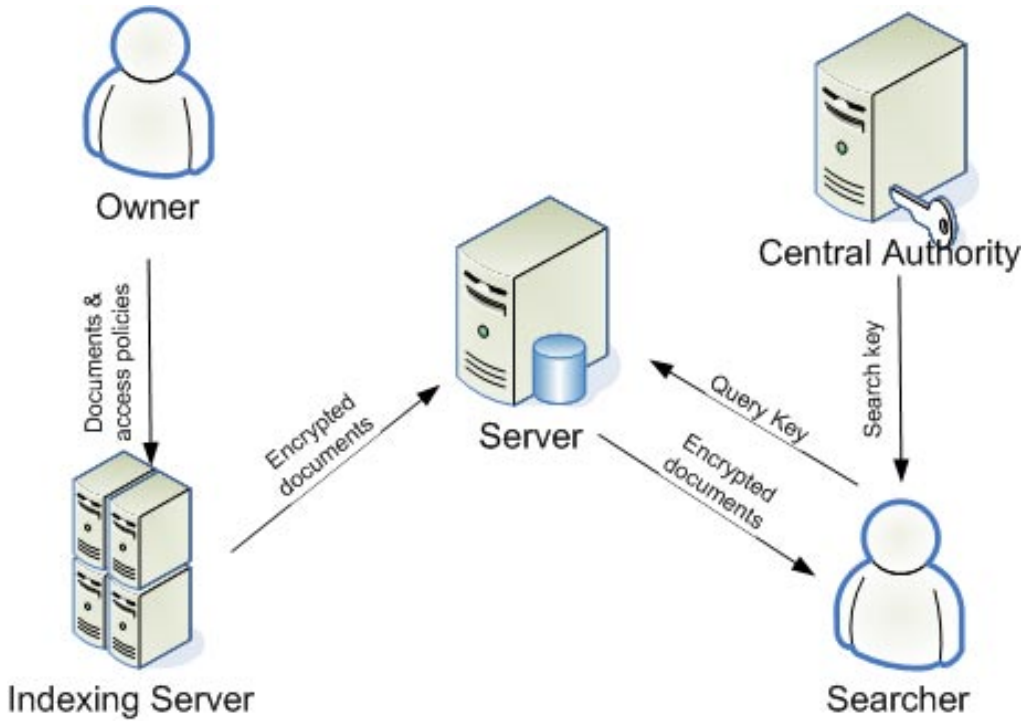


Figure: The overview of our proposal

Our system (Setup)

The central authority runs $\texttt{PE.Setup}(1^\ell)$ to get $(msk, mpk, para)$, and sends mpk to the indexing server *securely*, and publishes $para$.

Our system (Create)

- A creator sends the document and the associated access policy to the indexing server.
- On receiving the data from the creator, the indexing server indexes the document and computes

$$\text{Encrypted data}||\text{encrypted keywords}$$
$$C||C'$$

where $C = \texttt{PE.Enc}(m, mpk, \mathcal{P})$,
$C' = \texttt{PE.Enc}(Y, mpk, \mathcal{P} \wedge \mathbb{W})$, $\mathbb{W}$ is the keyword set contains all the keywords m contains.

- At last, the indexing server sends $C||C'$ to the server in the cloud computing.

Our system (Query)

- (Perform only once) A searcher gets a search key k_s from the central authority.

$$k_s = \mathtt{PE.KeyGen}(msk, \mathbb{A} \wedge \mathcal{W}),$$

where $\mathbb{A}$ is the searcher's attributes, $\mathcal{W}$ is the set of all the keywords.

- The searcher computes a query key k_q associated to his $\mathcal{Q}$, and sends it to the server in cloud computing.

$$k_q = \mathtt{PE.Dele}(k_s, \mathbb{A} \wedge \mathcal{Q})$$

- The server in the cloud computing checks

$$Y \stackrel{?}{=} \mathtt{PE.Dec}(k_q, C').$$

If yes, return C.

Our system (Update)

Document Update The same as that in case of one creator vs. one searcher.

Our system (Update)

Document Update The same as that in case of one creator vs. one searcher.

User Update Add time attributes in the document and k_s.

Our system (Update)

Document Update The same as that in case of one creator vs. one searcher.

User Update Add time attributes in the document and k_s.

$$C = \mathtt{PE.Enc}(m, mpk, \mathcal{P} \wedge t),$$

$$C' = \mathtt{PE.Enc}(L, mpk, \mathcal{P} \wedge \mathcal{W} \wedge t),$$

$$k_s = \mathtt{PE.KeyGen}(msk, \mathbb{A} \wedge \mathcal{W} \wedge \mathcal{T}),$$

An example database

Table: The example documents and their access polices.

Document	Content	Access Polices
A	ACL-based search.	(posn.=employee AND dept.=research) OR (posn.=senior AND dept.=eng.)
B	ACL-based enterprise search.	posn.=senior AND dept.=research
C	PE-based enterprise search.	(posn.=employee AND dept.=research) OR (posn.=senior AND dept.=eng.)

Encrypted example database

Table: The encrypted results of the example documents.

Document	Encrypted Result
A	$C_A = \mathtt{PE.Enc}(m_A, mpk, \mathcal{P}_A)$ $C'_A = \mathtt{PE.Enc}(Y, mpk, \mathcal{P}_A \bigwedge \mathbb{W}_A)$
B	$C_B = \mathtt{PE.Enc}(m_B, mpk, \mathcal{P}_B)$ $C'_B = \mathtt{PE.Enc}(Y, mpk, \mathcal{P}_B \bigwedge \mathbb{W}_B)$
C	$C'_C = \mathtt{PE.Enc}(m_C, mpk, \mathcal{P}_C)$ $C_C = \mathtt{PE.Enc}(Y, mpk, \mathcal{P}_C \bigwedge \mathbb{W}_C)$

$\mathbb{W}_A = (\mathtt{ACL} \wedge \mathtt{ACL\text{-}based} \wedge \mathtt{search})$;
$\mathbb{W}_B = (\mathtt{ACL} \wedge \mathtt{ACL\text{-}based} \wedge \mathtt{enterprise} \wedge \mathtt{search} \wedge \mathtt{enterprise\ search})$;
$\mathbb{W}_C = (\mathtt{PE} \wedge \mathtt{PE\text{-}based} \wedge \mathtt{enterprise} \wedge \mathtt{search} \wedge \mathtt{enterprise\ search})$.

The example searcher

The illustrating document searcher is a senior employee in engineering department, and his query is ((“ACL” AND “search”) OR (“PE” AND “enterprise search”)).

$$k_s = \mathtt{PE.KeyGen}(msk, (\mathtt{posn.} = \mathtt{senior} \wedge \mathtt{dept.} = \mathtt{eng.}) \wedge \mathcal{W})$$

$$k_q = \mathtt{PE.Dele}(k_s, (\mathtt{posn.} = \mathtt{senior} \wedge \mathtt{dept.} = \mathtt{eng.}) \wedge ((\mathtt{ACL} \wedge \mathtt{search}) \vee (\mathtt{PE} \wedge \mathtt{enterprise\ search})))$$

Security

Document confidentiality Due to the underlying PE scheme is payload-hiding, the one without associated sk cannot get m.

Inference resistance Under the assumptions that the underlying PE scheme is payload-hiding, and that the indexing server does not collude with the server in cloud computing or any creator, the adversary cannot relate k_w to the real keyword.

Policy privacy Due to the policy-hiding security of the underlying PE scheme, the adversary cannot figure out the access policy of any document.

Limitations

- Sequential scan.
- A fully trusted indexing server.

Any Question?

Thank you!