Juni Khyat
(UGC Care Group I Listed Journal)

ISSN: 2278-4632
Vol-10 Issue-6 No. 3 June 2020

A Privacy-Protection for Multi-Owner Sharing System using CP-ABKS in Cloud

[1]Dr. P. Penchalaiah [2]G.Gangadhar

[1]*Associate professor, Dept. of Master of Computer Applications, Narayana Engineering College, Gudur, AP.*

[2] *PG Scholar, Dept. of Master of Computer Applications, Narayana Engineering College, Gudur, AP.*

Abstract—Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) facilitates search queries and supports fine-grained access control over encrypted data in the cloud. However, prior CP-ABKS schemes were designed to support unshared multi-owner setting, and cannot be directly applied in the shared multi-owner setting (where each record is accredited by a fixed number of data owners), without incurring high computational and storage costs. In addition, due to privacy concerns on access policies, most existing schemes are vulnerable to off-line keyword-guessing attacks if the keyword space is of polynomial size. Furthermore, it is difficult to identify malicious users who leak the secret keys when more than one data user has the same subset of attributes. In this paper, we present a privacy-preserving CP-ABKS system with hidden access policy in Shared Multi-owner setting (basic ABKS-SM system), and demonstrate how it is improved to support malicious user tracing (modified ABKS-SM system). We then prove that the proposed ABKS-SM systems achieve selective security and resist off-line keyword-guessing attack in the generic bilinear group model. We also evaluate their performance using real-world datasets.

Keywords:Ciphertext-Policy Attribute-Based Encryption, Shared Multi-Owner Setting, Hidden Access Policy, User Tracing, Off-Line Keyword-Guessing Attack.

1. INTRODUCTION

Cloud computing [1], [2] is widely used by both individuals and organizations (including government agencies), for example to store and process large volume of data (e.g., text, image, and video), which are typically encrypted prior to outsourcing [3], [4], [5]. Searchable Encryption (SE) schemes [6], [7], [8], [9] enable data users to securely search and selectively retrieve records of interest over encrypted data (outsourced to the cloud), according to user-specified keywords. There are, however, other desirable properties when dealing with encrypted data outsourced to the cloud. For example, when encrypting significant volume of data, conventional encryption approaches suffer from limitations due to having multiple copies of ciphertexts (e.g., in public key encryption schemes) and complex and expensive key management (e.g., In symmetric encryption schemes). Ciphertext-Policy Attribute-Based

Encryption (CP-ABE) schemes are designed to mitigate these two limitations, as well as enhancing access permissions in multi-user setting and facilitating one-to-many encryption (rather than one-to-one) [10], [11], [12], [13].

However, in standard CP-ABE schemes, an access policy in plaintext is associated with a ciphertext may result in leakage of sensitive information. For example, inane-health system, hospital A encrypts a patient’s electronic medical record (EMR) using CP-ABE with an access policy, such as (“ID: 1788” AND “Hospital: Hospital A”) OR (“Doctor: Cardiologist” AND “Hospital: Hospital B”) – see Fig. 1. Hence, one can easily infer from the user attribute set (“Cardiologist”, “Hospital B”) that patient (“ID:1788”) in Hospital A likely suffers from a heart condition. Such privacy leakage is clearly not appropriate, particularly if the medical condition is more sensitive (e.g., sexually transmitted diseases such as chlamydia, gonorrhoea, and human papillomavirus infections). In addition, medical organizations are subject to exacting regulatory oversight in most developed jurisdictions. Hence, there have been efforts to design CP-ABE scheme with hidden access policies [14].

Therehavealsobeeneffortstodesignschemesthatallow a data owner to delegate his/her search capability in a fine-grained manner, which allows other data users to search, retrieve and decrypt encrypted data of interest. Examples include Ciphertext-Policy Attribute-Based Keyword Search(CP-ABKS).

2. RELATED WORK

The first symmetric SE scheme and asymmetrical SE scheme were presented by Songetal.[6] and Bonehetal.[7],respectively. Subsequent SE schemes were designed to support a range of features, such as single keyword search multi-keyword search [8] and ranked keyword search.

CP-ABE was designed to allow fine-grained access control over ciphertexts, and CP-ABKS was designed to support both fine-grained access control and keyword search simultaneously. For example, Zheng et al. presented the CP-ABKS scheme that enables data owners to grant fine-grained search permissions, Sun et al. presented an owner-enforced CP-ABKS scheme that supports user revocation and is shown to be selectively secure against chosen-keyword attack. However, the computational costs of these two schemes grow linearly as the number of system attributes increases. This is not scalable in practice. To minimize computational costs and ciphertext size required in such schemes, Li et al. implemented a keyword search function in attribute-based encryption (ABE) scheme, by outsourcing key-issuing and decryption operations. Dong et al. also designed an efficient CP-ABKS scheme via an online/offline approach when considering resource constrained mobile devices.

Another limitation of CP-ABKS schemes is that an honest-but-curious cloud service provider may seek to learn additional sensitive information, other than the stored ciphertexts and submitted trapdoors. Also, secret keys (or decryption keys) are defined over different attribute sets, rather than their corresponding identities. Hence, while CPABKS schemes can achieve one-to-many encryption and support expressive access control, they are not capable of identifying data users leaking the secret keys if the ‘culprits’ have the same subset of attributes as other honest data users. Hence, a data user may choose to deliberately trade his/her (partial or entire) decryption privileges for profit without being caught. Thus, traceability should be incorporated in the design of CP-ABKS schemes to facilitate accountability. Based on the traceable CP-ABE technique, we extend the traceability feature in the basic ABKS-SM system to construct the modified ABKS-SM system so that the requirements of real-world applications can be satisfied.

3. PROBLEM DEFENITION

The system and threat models, basic and modified ABKSSM systems definition.

The System Model

ThesystemmodelforthebasicandmodifiedABKS-SMsystemscomprisesfourtypesofentities,namely:multipleData Owners (DOs), Data Users (DUs), Cloud Service Provider (CSP) and Trusted Third-Party (TTP).

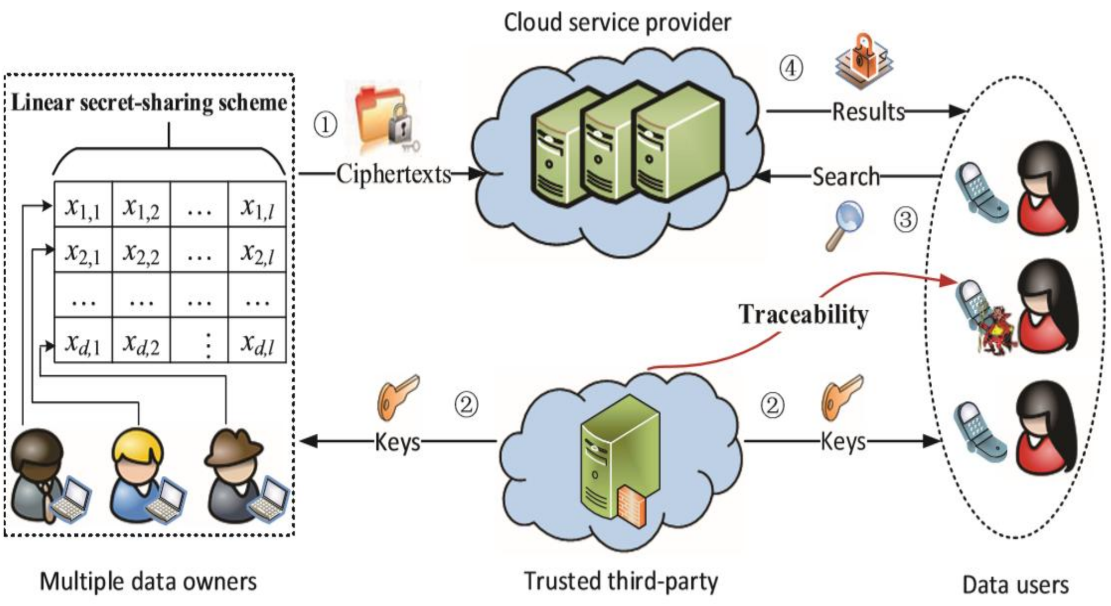
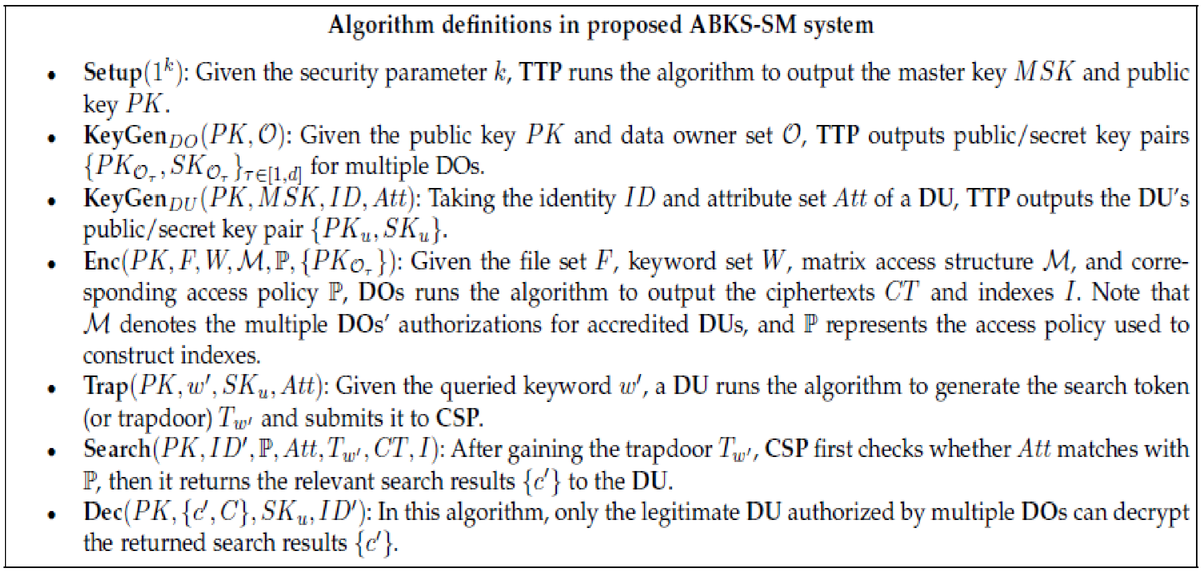


Fig. 1. Basic (or modified) ABKS-SM System Model

This system model consists and implements the following modules:

- **DOs:** When taking the shared multi-owner setting into consideration, DOs encrypt the file encryption keys using LSSS, build indexes using access policy based on AND-Gates, and upload the ciphertexts to CSP.
- **DUs:** Authorized DU can search encrypted files of interest and gain access to the plaintext once he/she has been attributed by multiple DOs. Note that the DU can decrypt the final search results if he/she has obtained relevant authorizations assigned by multiple DOs.
- **CSP:** The cloud server provides many services, such as data storage, computation and retrieval. When a DU issues a search query by submitting a search, token generated according to his/her interested keyword, the CSP will attempt to match it with the indexes and return the relevant search results to the DU.
- **TTP:** Firstly, it is responsible for initializing the system and generating the public/secret key pairs for cloud clients including DOs and DUs. Secondly, it can trace the DU who leaks the secret key to unauthorized entities.



Algorithm definitions in proposed ABKS-SM system

- **Setup**(1^k): Given the security parameter k, TTP runs the algorithm to output the master key MSK and public key PK.
- **KeyGen**$_{DO}(PK, \mathcal{O})$: Given the public key PK and data owner set $\mathcal{O}$, TTP outputs public/secret key pairs $\{PK_{\mathcal{O}_\tau}, SK_{\mathcal{O}_\tau}\}_{\tau\in[1,d]}$ for multiple DOs.
- **KeyGen**$_{DU}(PK, MSK, ID, Att)$: Taking the identity ID and attribute set Att of a DU, TTP outputs the DU's public/secret key pair $\{PK_u, SK_u\}$.
- **Enc**$(PK, F, W, \mathcal{M}, \mathbb{P}, \{PK_{\mathcal{O}_\tau}\})$: Given the file set F, keyword set W, matrix access structure $\mathcal{M}$, and corresponding access policy $\mathbb{P}$, DOs runs the algorithm to output the ciphertexts CT and indexes I. Note that $\mathcal{M}$ denotes the multiple DOs' authorizations for accredited DUs, and $\mathbb{P}$ represents the access policy used to construct indexes.
- **Trap**(PK, w', SK_u, Att): Given the queried keyword w', a DU runs the algorithm to generate the search token (or trapdoor) $T_{w'}$ and submits it to CSP.
- **Search**$(PK, ID', \mathbb{P}, Att, T_{w'}, CT, I)$: After gaining the trapdoor $T_{w'}$, CSP first checks whether Att matches with $\mathbb{P}$, then it returns the relevant search results $\{c'\}$ to the DU.
- **Dec**$(PK, \{c', C\}, SK_u, ID')$: In this algorithm, only the legitimate DU authorized by multiple DOs can decrypt the returned search results $\{c'\}$.

Fig. 2. Definition of Basic ABKS-SM System

Overview of Basic ABKS-SM System

As the modified ABKS-SM system has the similar algorithms as the basic ABKS-SM system except for the traceability algorithm, we just give the algorithm definitions of the basic ABKS-SM system.

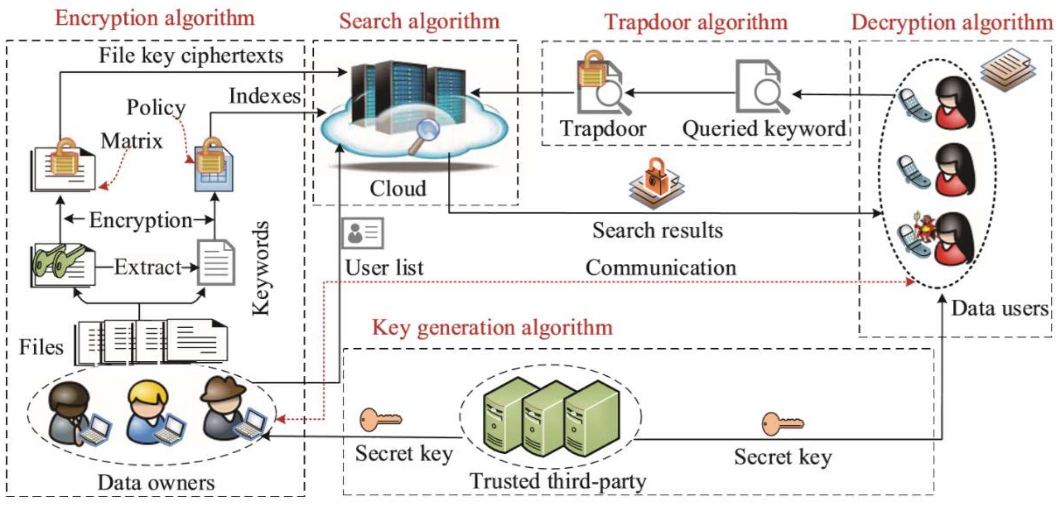


Fig. 3. Architecture of basic ABKS-SM system

We also present the architecture of basic ABKS-SM system – see Fig. 3. The Setup algorithm performs the system initialization, such as generating the public keys and master keys. The Keygen algorithm includes KeyGenDO and KeyGenDU sub algorithms, which generate public/secret key pairs for multiple DOs and DUs, respectively. As for Enc algorithm, multiple DOs first extract keywords from the files be for eout putting the file key ciphertexts and encrypted indexes, by using LSSS and access policy respectively. In Trap algorithm, a DU submits the trapdoor generated according to his/her queried keyword to CSP. After that, the CSP conducts Search algorithm and sends the authorized searchresultstoDU.Beforedecryptingtheencryptedsearch results in Dec algorithm, DU needs to obtain the relevant authorizations by interacting with DOs, which is shown by the red dotted line in Fig. 5. After being accredited by multiple DOs, DU obtains the plaintext results.

Security Requirements

Similar to security requirements in typical private information retrieval schemes, both the basic and modified ABKS-SM systems should ensure the following privacy requirements:

• Data privacy. DUs can access the shared data, if and only if, they have valid authorization from multiple DOs.

• Privacy for DUs. CSP is convinced that DU's search queries are authorized by DOs, without learning any potential information about the queried content.

• Privacy for DOs. Even if a part of DOs is corrupted, the adversary cannot forge valid authorizations from remaining DOs as there exist no interactions and additional computing operations among multiple DOs.

The basic and modified ABKS-SM systems satisfy the above privacy requirements if they achieve selective security in the generic bilinear group model.

4. PROPOSED ABKS-SM SYSTEMS

In this section, we first present the concrete construction of the basic ABKS-SM system, which supports fine-grained keyword search and hidden access policy. Then, we explain how the basic ABKS-SM system is extended to achieve malicious user tracing in the modified ABKS-SM system.

Construction of Basic ABKS-SM System

Unlike existing CP-ABKS schemes, we consider a shared multi-owner setting where each file is co-owned by a group of DOs. In the basic ABKS-SM system, we use conventional symmetric encryption algorithm (AES, DES, etc.), access matrix$M_{d\times l}$

(or (d,l)-LSSS), and access policy P, to respectively encrypt files, file encryption keys and keywords.

Even though a certain DU can issue search queries and obtain the returned search results, he/she cannot decrypt the encrypted data without valid authorizations from multiple DOs. Moreover, in practice, the access policies contain sensitive information and should also be protected. However, existing CP-ABKS schemes with hidden access policies are not practical since any malicious DU having the same attribute set with others, can leak his/her decryption privilege without fear of being caught.

Thus, we further extend the traceability function in the modified ABKS-SMsystem ,and present a concrete construction. Note that we design a two-level access control over outsourced files, which is shown in Fig. 4. As for the first level access control over filed encryption, we design an access matrix Md×l , which is used to encrypt each file encryption key according to DU’s identity list by leveraging LSSS.

For the second-level access control over encrypted files, an access policy is specified to generate indexes according to DU’s attributes by utilizing AND-Gates on multi-valued access structure. Note that DU can request the first-level access control, if and only if, he/she satisfies the second level access control.

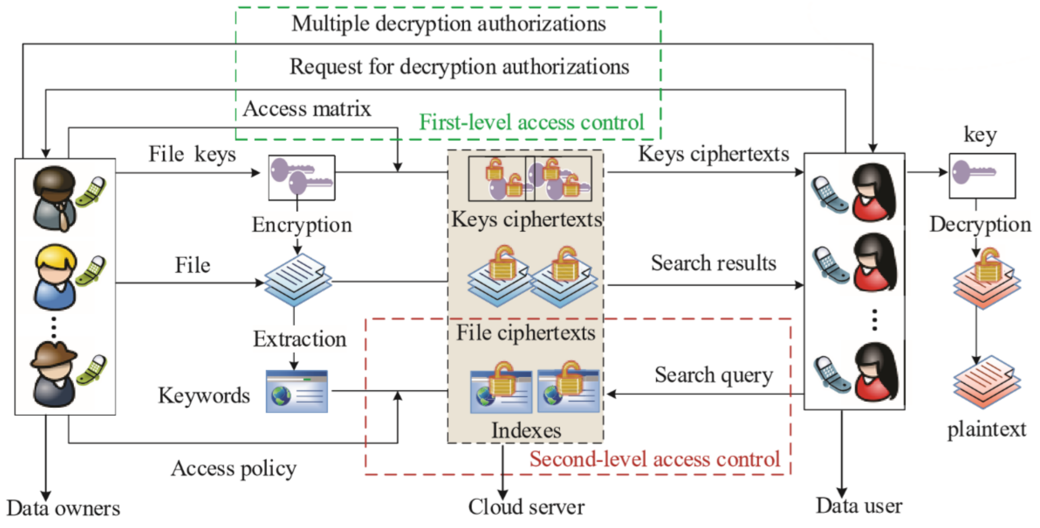
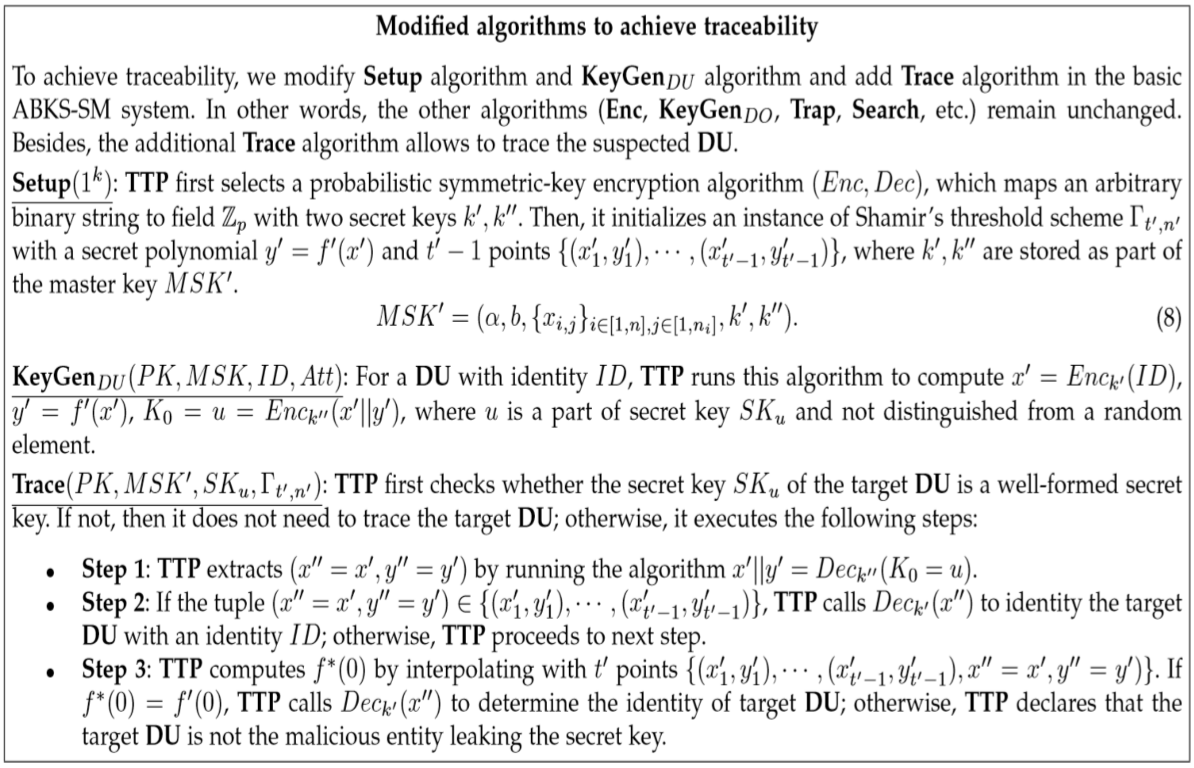


Fig. 4. Two-Level Access Control in Basic ABKS-SM System

ISSN: 2278-4632
Vol-10 Issue-6 No. 3 June 2020

Construction of Modified ABKS-SM System

To trace malicious DU, we will utilize Shamir's threshold scheme Γt′,n′ and a probabilistic encryption algorithm. Thus,themodifiedABKS-SMsystemonlystorest′−1points and the value f′(0) on the polynomial f′(x′) at system initialization, so that the storage cost of user tracing is constant. Due to limited space, we present the content that differs from the original algorithm, as shown in Fig. 5.



Modified algorithms to achieve traceability

To achieve traceability, we modify **Setup** algorithm and **KeyGen**$_{DU}$ algorithm and add **Trace** algorithm in the basic ABKS-SM system. In other words, the other algorithms (**Enc**, **KeyGen**$_{DO}$, **Trap**, **Search**, etc.) remain unchanged. Besides, the additional **Trace** algorithm allows to trace the suspected **DU**.

Setup(1^k): **TTP** first selects a probabilistic symmetric-key encryption algorithm (Enc, Dec), which maps an arbitrary binary string to field $\mathbb{Z}_p$ with two secret keys k', k''. Then, it initializes an instance of Shamir's threshold scheme $\Gamma_{t',n'}$ with a secret polynomial $y' = f'(x')$ and $t' - 1$ points $\{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1})\}$, where k', k'' are stored as part of the master key MSK'.

$$MSK' = (\alpha, b, \{x_{i,j}\}_{i\in[1,n],j\in[1,n_i]}, k', k''). \tag{8}$$

KeyGen$_{DU}(PK, MSK, ID, Att)$: For a **DU** with identity ID, **TTP** runs this algorithm to compute $x' = Enc_{k'}(ID)$, $y' = f'(x')$, $K_0 = u = Enc_{k''}(x'||y')$, where u is a part of secret key SK_u and not distinguished from a random element.

Trace$(PK, MSK', SK_u, \Gamma_{t',n'})$: **TTP** first checks whether the secret key SK_u of the target **DU** is a well-formed secret key. If not, then it does not need to trace the target **DU**; otherwise, it executes the following steps:

- **Step 1**: **TTP** extracts $(x'' = x', y'' = y')$ by running the algorithm $x'||y' = Dec_{k''}(K_0 = u)$.
- **Step 2**: If the tuple $(x'' = x', y'' = y') \in \{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1})\}$, **TTP** calls $Dec_{k'}(x'')$ to identity the target **DU** with an identity ID; otherwise, **TTP** proceeds to next step.
- **Step 3**: **TTP** computes $f^*(0)$ by interpolating with t' points $\{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1}), x'' = x', y'' = y')\}$. If $f^*(0) = f'(0)$, **TTP** calls $Dec_{k'}(x'')$ to determine the identity of target **DU**; otherwise, **TTP** declares that the target **DU** is not the malicious entity leaking the secret key.

Fig. 5. Definition For Modified Algorithms

5. EXPERIMENTAL RESULTS & DISCUSION

In this section, we provide the performance assessment of the proposed system.We evaluate the performance of the basic ABKSSM system.

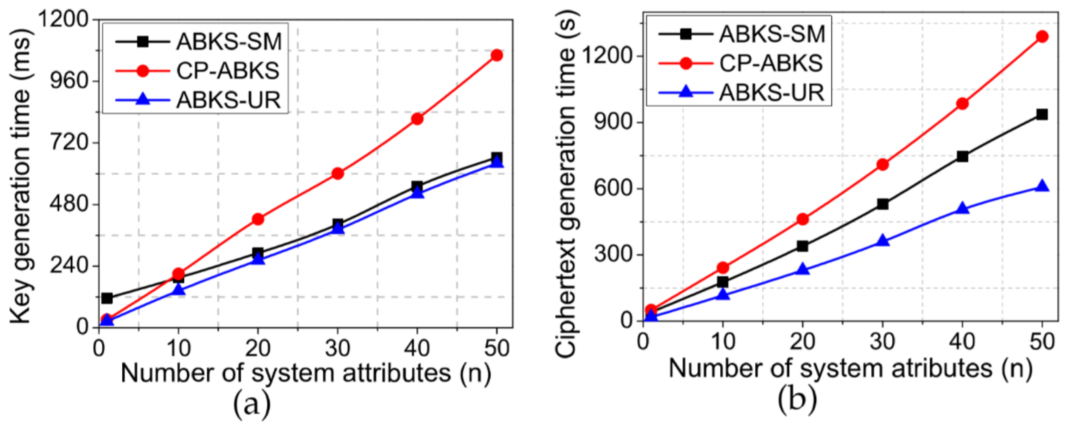


Fig.6.Computationalcostsinvariousalgorithms:(a)KeyGenAlgorithm; (b) Enc Algorithm

In Fig. 6(a), for comparison, we set d = 10 and vary the value of n from 1 to 50. We observe that the key generation time in these three schemes increases with the number of system attributes (n).As the value of d is very small in practice, the computational cost of KeyGen algorithm in the basic ABKS-SM system is only slightly more than that of ABKS-UR scheme. For example, when setting n = 20, the basic ABKS-SM system needs 291ms to generate keys, and both CP-ABKS and ABKS-UR schemes need 423 ms and 263 ms, respectively.

Assuming that each attribute has one possible value and d = 10, n ∈ [1] in Fig. 6(b), the ciphertexts generation time increases with increasing n. As the basic ABKS-SM system needs to encrypt the encryption keys and build indexes simultaneously, when compared with theABKSUR scheme, while the basic ABKS-SM system is still much efficient than the CP-ABKS scheme in terms of ciphertexts generation time due to the consuming hash operations nH′ in. For example, when setting n = 50, the basic ABKSSM system needs 937ms, and the CP-ABKS and ABKS-UR schemes need 1290 ms, 608 ms, respectively.

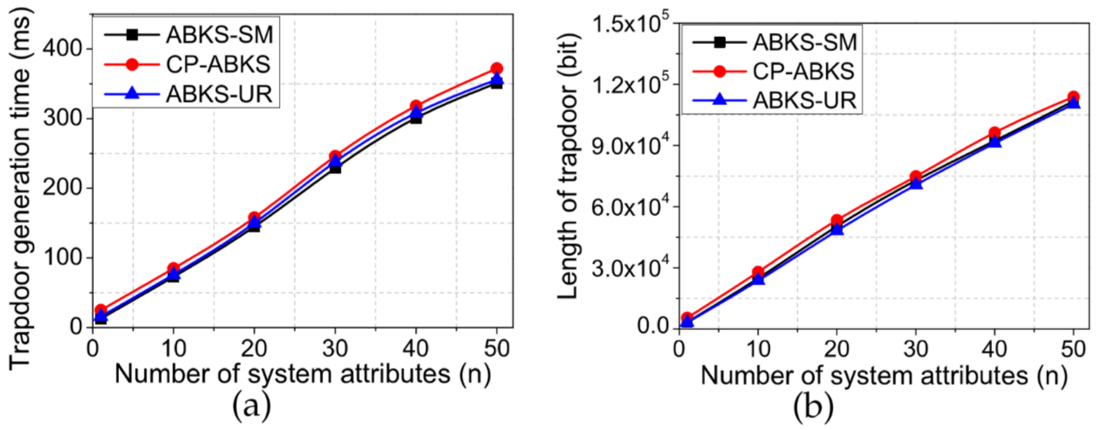


Fig. 7. Performance Analysis InTrap Algorithm: (A) Computational Cost; (B) Storage Cost

In Fig. 7(a) and (b), we analyze the performance of Trap algorithm for the schemes being studied, for n ranging from 1 to 50. We observe that the computational and storage costs of this algorithm almost linearly increase with n.

Furthermore, the performance of both ABKS-SM and ABKSUR schemes is similar, and the basic ABKS-SM system has a slightly better performance than CP-ABKS. For example, when setting n = 40, the computational and storage overhead of ABKS-SM is 301 ms and 11.27 KB, and for CP-ABKSand ABKS-UR schemes (318 ms, 11.27 KB) and (308 ms, 11.13 KB), respectively.

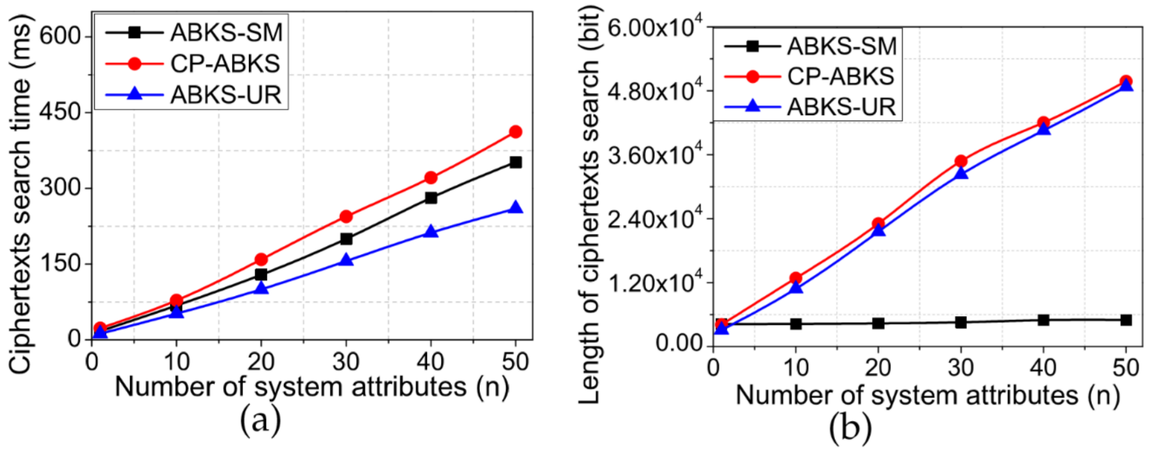


Fig. 8. Performance Analysis inSearch Algorithm: (A) Computational Cost; (B) Storage Cost

Fig. 8 (a) and (b) present the computational and storage costs for Search algorithm, respectively. For the ciphertext search time, the basic ABKS-SM system needs to conduct additional pairing operations nP, unlike the ABKS-UR scheme.

For Search algorithm, the storage cost of basic ABKS-SM system remains almost unchanged whilst the storage costs of CP-ABKS and ABKSUR schemes increase linearly with the number of system attributes(n).For example, whensetting n = 50,thestorage cost of the basic ABKS-SM system is 0.61 KB, and those of the CP-ABKS and ABKS-UR schemes are 6.57 KB and 6.69 KB, respectively.

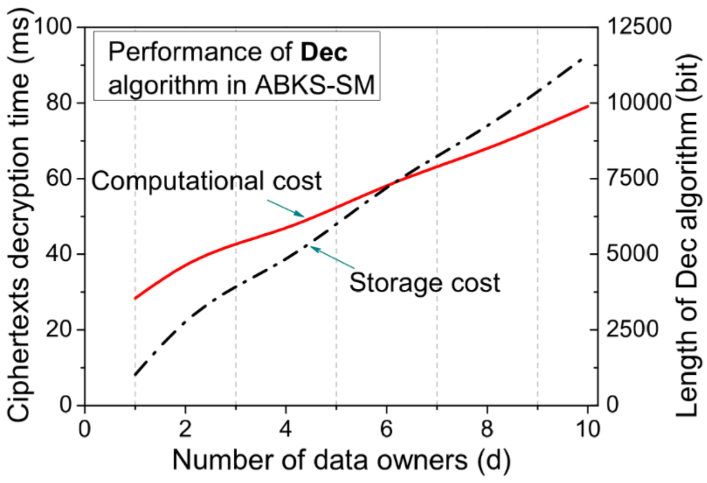


Fig. 9. The performance of Dec algorithm in ABKS-SM

As a queried DU must obtain valid authorizations from multiple DOs before decrypting the search results, we now present the performance evaluation of Dec in the basic ABKS-SM system.

The computational over head is shown by the red line and the storage overhead is denoted by the black dash line in Fig. 9. For example, when fixing d = 10, the ciphertext decryption process requires 79.1 ms, and its storage length is 1.42 KB. Thus, the basic ABKSSM system is suitable for deployment on resource-limited devices, such as mobile terminals and sensor nodes.

From these proposed schemes, we observe the performance(i.e.,computationalcosts, storage costs, etc.) for the three different datasets. As the resource-limited mobile devices need to initialize when executing algorithms in the basic ABKS-SM system, CPABKS and ABKS-UR schemes, these devices require slightly higher computational and storage costs in actual tests than those of a simulation. Although

Enc algorithm has a high computational cost, it does not affect user search experience as it is a one-time cost. In other words, the basic ABKSSM system does not incur high computational and storage overheads on resource-limited mobile devices during the execution of Keygen, Trap, Search and Dec algorithms in practice.

6. CONCLUSION

In the paper, we presented a practical attribute-based keyword search scheme supporting hidden access policy in the shared multi-owner setting. Furthermore, we demonstrated how the basic ABKS-SM system can be extended to support traceability (i.e., tracing of malicious DUs) in the modified ABKS-SM system, if desired. The formal security analysis showed that the basic and modified ABKS-SM systems achieve selective security and resist off-line key word guessing attacking the generic bilinear group model.We also demonstrated the utility of the proposed ABKS-SM systems by evaluating their performance using three real-world datasets and, on a test bed, including 11 mobile terminals and a high-performance workstation server.

One limitation of the proposed ABKS-SM systems is that as the number of system attributes increases, so does the computational and storage costs. Thus, we intend to improve the efficiency of the ABKS-SM systems in the future. Also, to facilitate the efficient locating of search results and minimizing the number of irrelevant search results, we will focus on expressive search (e.g., multi-keyword search and fuzzy keyword search) in our future work.

REFERENCES

1. J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, “Flexible and fine-grained attribute-based data storage in cloud computing,” IEEE TransactionsonServicesComputing, vol. 10, no. 5, pp. 785–796, 2017.
2. J. Li, H. Yan, and Y. Zhang, “Certificateless public integrity checking of group shared data on cloud storage,” IEEE Transactions on Services Computing, vol. PP, pp. 1–1, 2018.
3. D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, “Social attribute aware incentive mechanism for device-to-device video distribution,” IEEETransactionsonMultimedia,vol.19,no.8,pp.1908–1920, 2017.

4. D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy-efficient mobile edge collaboration for video distribution," IEEE Transactions on Multimedia, vol. 19, no. 10, pp. 2197–2209, 2017.
5. D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1–1, 2017.
6. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. IEEE Symposium on Security and Privacy (SP 2000), 2000, pp. 44–55.
7. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Proc. International conference on the theory and applications of cryptographic techniques (EUROCRYPT 2004), 2004, pp. 506–522.
8. H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classifiedsub-dictionariesoverencryptedclouddata,"IEEETransactions on Dependable and Secure Computing, vol. 13, no. 3, pp. 312–325, 2016.
9. R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," IEEE transactions on information forensics and security, vol. 11, no. 4, pp. 789–798, 2016.
10. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Proc. IEEE Symposium on Security and Privacy (SP 2007), 2007, pp. 321–334.
11. B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in Proc. InternationalConferenceonPracticeandTheoryinPublicKeyCryptography (PKC 2011), 2011, pp. 53–70.
12. J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute-based encryption," IEEE Transactions on Services Computing, vol. PP, pp. 1–1, 2017.
13. J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collision avoidance cp-abe with efficient attribute revocation for cloud storage," IEEE Systems Journal, vol. 12, no. 2, pp. 1767–1777, 2018.
14. T.V.X.Phuong,G.Yang,andW.Susilo,"Hiddenciphertextpolicy attribute-based encryption under standard assumptions," IEEE Transactions on Information Forensics and Security, vol. 11, no. 1, pp. 35–45, 2016.

Author's Profile:

Dr. P Penchalaiah received Ph.D (Aug-2017) in Computer Science & Applications from Vikrama Simhapuri University, A.P State Govt. University, India. He qualified the UGC-NET & AP-SET exams that ensure minimum standards in teaching profession and research exams for Indian National. He has more than 12 years of experience in Academic and Industry (Software Engineer) as well. His research papers are indexed in international renowned indexing database like SCOPUS, WoS and he authored three books. His research areas of interest are Cryptology, Information Security, Cyber Security, Cyber Forensics, and Data Sciences.

G. Gangadharhas received his B.Sc degree in Computer Science from Dr. CRR Degree College, Sydapuram affiliated to VikramaSimhapuri University, Nellore in 2017 and pursuing PG degree in Mater of Computer Applications (MCA) from Narayana Engineering College, Gudur affiliated to JNTU, Ananthapur, AndhraPradesh, India.