Cryptanalysis of Searchable Anonymous Attribute Based Encryption

Payal Chaudhari[1,2] and Manik Lal Das[2]

[1] LDRP-ITR, Gandhinagar, India.
[2] DA-IICT, Gandhinagar, India.
payal.ldrp@gmail.com; maniklal_das@daiict.ac.in

Abstract. Ciphertext Policy Attribute Based Encryption (CP - ABE) is a public key primitive in which a user is only able to decrypt a ciphertext if the attributes associated with secret key and the access policy connected with ciphertext matches. CP-ABE provides both confidentiality and access control to the data stored in public cloud. Anonymous CP-ABE is an adaptation of ABE where in addition to data confidentiality and access control, receiver anonymity is also provided. Recently, Koo *et al.* (2013) proposed a scheme in the area of anonymous CP-ABE. We found security flaws in *Koo et al.*'s scheme. Their scheme does not support anonymity of the intended receiver, which was the main security claim in [8]. In this paper, we show how one can identify the receiver in Koo *et al.*'s scheme.

Keywords: Attribute Based Encryption, Anonymity, Bilinear Pairing, Access Structure

1 Introduction

Cloud computing is a comprehensive model which provides virtually unlimited configurable computing resources such as storage, network, applications and services. Users can acquire these resources on-demand basis. Many enterprises and individuals are outsourcing their data to the cloud storage servers in order to reduce the capital and human power investment in building and maintain their own data centers. The outsourced data may contain much sensitive information, such as Personal Health Records (PHRs), financial details, personal photos etc. Therefore the cloud servers or unauthorized users are encouraged to access and obtain such sensitive information. Without ensuring security against these potential risks, users may hesitate to outsource their data to cloud servers.

To protect data security and privacy, encryption technology appears to be a promising solution. Before outsourcing the data, the data owners first encrypt their documents. However, once the encrypted data are outsourced to cloud servers, two major requirements becomes apparent for user convenience: Access control and Search over encrypted data. To provide a solution for secure and fine-grained data sharing and decentralized access control, Sahai and Waters first

introduced the concept of attribute-based encryption (ABE) [9]. ABE is much more beneficial than other encryption technologies as it provides one-to-many encryption instead of one-to-one. There are two variants of ABE: Ciphertext-Policy ABE(CP-ABE)[1] and Key-Policy ABE (KP-ABE)[2]. In CP-ABE data encryption is done as per access policy. Here access policy describes the combination of required attributes. User's secret key contains the attribute values which a user possesses. If the users' key matches with access policy then he will be able to decrypt the documents. In KP-ABE access policy is attached with user's secret key and attributes' secret shares are listed with encrypted documents. Sometimes to enlist the receiver's attributes along with ciphertext in clear form, discloses the receiver's identity and also in worst case the statistical information about the encrypted documents. Therefore the design of anonymous attribute based encryption has been formulated [3–7].

Recently Koo *et al.* have proposed a searchable anonymous ABE scheme where search on encrypted data stored in cloud is done based on data owner's ID. The scheme claims for data confidentiality, sender anonymity and receiver anonymity. We show that Koo *et al.*'s scheme fails to achieve the receiver anonymity.

Rest of the paper is organized as follows. In section 2 we define some preliminaries which will be used in the scheme presented in [8]. In section 3 we present the construction of the scheme proposed by Koo et al. in [8]. In section 4 the cryptanalysis of the scheme [8] is discussed. We conclude our paper in section 5.

2 Preliminaries

2.1 Bilinear Mapping

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_1 and e be a bilinear map, $e : G_0 \times G_0 \rightarrow G_1$. The bilinear map e has the following properties:

- Bilinearity: for all $u,v \in G_0$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(u,v)^{ab}$.
- Non-degeneracy: $e(g,g) \neq 1$.

We say that G_0 is a bilinear group if the group operation in G_0 and the bilinear map $e : G_0 \times G_0 \rightarrow G_1$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 Access Tree

Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. The access tree $\mathcal{T}$ also defines an

ordering between the children of every node, that is, the children of a node are numbered from 1 to *num*. Each child of a parent will have unique index number. To facilitate working with the access trees, following functions are being used.

- $parent(x)$ = parent of the node x in the tree.
- $att(x)$ = attribute associated with x if and only if x is a leaf node.
- $index(x)$ = index number of node x as a child of its parent node. The value will be between 1 to *num*.

The encryption algorithm first choose a polynomial q_x for each node x (including the leaves) in the tree $\mathcal{T}$. These polynomials are chosen in a top-down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x$ - 1. Starting with the root node R the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent}(x)(index(x))$ and chooses d_x other points randomly to completely define q_x.

3 Scheme presented by Koo *et al.*[8]

Koo *et al.* [8] have presented a scheme for author based document search. The user will be able to pass a query to search for the document which matches with his own attributes and sender's ID. We found that the scheme is vulnerable to an adversary.

3.1 System Model

In the scheme presented by Koo *et. al.* four entities are playing the role.

- Trusted Authority : It is responsible for the setup of the system and generating user specific secret keys.
- Cloud service provider (CSP): This is a semitrusted entity where the users outosurce their data in encrypted form
- Data owner: They encrypt and store their data in cloud stotage
- Data Retriever/Receiver : These are the users who queries the CSP for encrypted data in the cloud storage system and retrieves the data only if his attributes satisfies the access policy specified by the data owner.

3.2 Phases of the scheme [8]

System setup The setup algorithm performed by a trusted authority will choose a bilinear group G of prime order p with generator g. Next it chooses two random exponents α, $\beta \in Z_p$, and a cryptographic hash function $H : \{0,1\}^* \rightarrow G$, which behaves like a random oracle. Then public parameter PK and master secret MK are computed as: $PK = (G, g, h = g^{\beta}, \omega = e(g,g)^{\alpha})$, $MK = (\beta, g^{\alpha})$.

Key Generation In this phase, the trusted authority generates anonymous key A_O for data owner and private key SK for receiver.

- For the data owner with ID_0, the trusted authority runs anonymous key generation algorithm $KeyGen_O$ and returns the result, $A_O = H(ID_0)^\beta$, to the data owner.
- For a receiver $u_i \in U$ with identity ID_i, the trusted authority runs $KeyGen_i$. It chooses a random $r \in Z_p$ for individual user and $r_j \in Z_p$ for each attribute $\lambda_j \in \Lambda_i$ where Λ_i denotes the attribute set belonging to user u_i. Then the private key SK is computed as:

$$SK = (D = g^{(\alpha+r)/\beta}, \{D_j = g^r H(\lambda_j)^{r_j}, D'_j = g^{r_j}, D''_j = H(\lambda_j)^\beta\}_{\lambda_j \in \Lambda_i})$$

Encryption Before outsourcing the data content to cloud storage, the data owner having the ID_O generates its pseudonym by running **PseudoGen(**PK**,** ID_O**)**. In PseudoGen algorithm, the data owner chooses a random $t \in Z_p$ and generates its pseudonym $P_O = H(ID_O)^t$ by itself and publicizes this pseudonym.

Then the data owner encrypts data M by running **Encrypt** algorithm, which will take as input the public parameter PK, its pseudonym P_O, a message M to be encrypted under the access tree $\mathcal{T}$, and output the ciphertext CT_0 . After then, attribute scrambling procedure, **AttrScm**, is applied to the ciphertext CT_0 generating new ciphertext CT to be located in the cloud storage.

- **Data Encryption(Encrypt):** In this phase the algorithm chooses a polynomial q_x for each node x (including the leaves) in a top-down manner, starting from the root node R in the tree $\mathcal{T}$. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of the node, that is, $d_x = k_x$ - 1. On the root node R, the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other nodes x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x. Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is then constructed under the access tree $\mathcal{T}$ and computed as:

$$\begin{aligned} CT' = (\mathcal{T}, \tilde{C} = M\omega^s, C = h^s, C'' = P_O, \\ \{C_y = g^{q_y(0)}, C'_y = H(attr_y)^{q_y(0)}\}_{y \in Y}) \end{aligned} \tag{1}$$

- **Attribute Scrambling(AttrScm):** Once the data content M is encrypted to CT' , the attribute values are hidden in this phase to provide receiver anonymity. In this phase, the data owner obfuscates all attribute values in $\mathcal{T}$ and obtains a new access tree $\mathcal{T}'$ by running **AttrScm(**CT_0**,**A_O**,**S**)**. For the set of attributes $S = \{\lambda_i, \cdots, \lambda_k\}$ where $1 \le i \le k \le |L|$ to be used in the access tree, the data owner generates:

$$\begin{aligned} K_{O,S} &= \{e(A_O^t, H(\lambda_j))\}_{\lambda_j \in S} \\ &= \{e(H(ID_O)^{\beta t}, H(\lambda_j))\}_{\lambda_j \in S} \\ &= \{e(H(ID_O), H(\lambda_j))^{\beta t}\}_{\lambda_j \in S} \end{aligned}$$

and assigns $scm_{att_x} \in K_{O,S}$ to leaf node x in $\mathcal{T}$ instead of λ_x corresponding to $attr_x$. This results in the access tree $\mathcal{T}'$. The new encrypted content CT to be outsourced is made as: $CT = (\mathcal{T}', \tilde{C}, C, C'', \{C_y, C'_y\}_{y \in Y})$

After this phase, the data owner uploads CT to the remote data storage managed by the CSP (Cloud Service Provider).

Access Outsourced Data This part of scheme facilitates the retrieval of encrypted data from cloud storage.

- **Data query (Query):** At the initial round, a retriever can first acquire a pseudonym list of data owners from the CSP or from the data owners themselves.Once the retriever determines to retrieve a data of some data owner with $C'' = P_O$ in the cloud storage and wants to access it, it can generate cryptographic index terms for corresponding attributes as the agreed session key as follows:

$$\begin{aligned} K_{O,\Lambda_i} &= \{e(D''_j, C'')\}_{j \in \Lambda_i} \\ &= \{e(H(ID_O)^t, H(\lambda_j)^\beta)\}_{j \in \Lambda_i} \\ &= \{e(H(ID_O), H(\lambda_j))^{\beta t}\}_{j \in \Lambda_i} \end{aligned}$$

  After then, the retriever sends a subset of those scrambled index information $K_{O,\Lambda'_i} \subseteq K_{O,\Lambda_i}$ as a data request query to the CSP.
- **Data Retrieval (Retrieve):** On request of an access to encrypted contents stored in the cloud storage with scrambled index terms K_{O,Λ'_i}, the CSP determines whether the requested item is stored in the storage and which one is satisfied with the requested index terms.

  The authors have formulated a simple comparison algorithm $\mathcal{C}(\mathcal{T}, K_{O,\Lambda'_i})$ which returns boolean value true or false. Let T_x be a subtree of T rooted at the node x and X' be a set of children whose parent is the node x such that $X' = \{x' \in Y_x$ and parent$(x') = \text{x}\}$. $\mathcal{C}(\mathcal{T}, K_{O,\Lambda'_i})$ is computed recursively as follows. If x is a leaf node, $\mathcal{C}(\mathcal{T}_x, K_{O,\Lambda'_i})$ returns true if and only if $attr_x \in K_{O,\Lambda'_i}$. If x is a non-leaf node in $\mathcal{T}$, $\mathcal{C}(\mathcal{T}, K_{O,\Lambda'_i})$ returns true if and only if at least k_x children return true. For each ciphertext CT_i, where $0 \leq i \leq m$, the CSP simply follows the access tree T^i and determines whether $\mathcal{C}(\mathcal{T}^i, K_{O,\Lambda'_i})$ returns true or not. If there are such ciphertexts, then the algorithm returns true and the CSP sends the corresponding ciphertexts to the retriever.

Decrypt Upon receiving the requested content in encrypted form, the receiver obtains the plaintext by using decryption algorithm DecryptNode which can be described as a recursive algorithm.
DecryptNode(CT,SK,S): For a leaf node x in access tree the algorithm computes

as follows: If i $(= attr_x) \in S$ then

$$\begin{aligned} DecryptNode(CT, SK, S) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g,g)^{rq_x(0)} \\ &= F_x \end{aligned}$$

If x is a nonleaf node then the algorithm proceeds as follows : $\{\forall z \in$ children of $x\}$, it calls the DecryptNode(CT, SK, z) and stores the output as F_z. Let S_x is the arbitrary k_x sized set of child nodes z such that $F_z \neq \perp$, then next step is computed as

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,s'_x(0)}} \\ &= \prod (e(g,g)^{rq_z(0)})^{\Delta_{i,s'_x(0)}} \\ &= \prod (e(g,g)^{rq_{parent(z)(index(z))}})^{\Delta_{i,s'_x(0)}} \\ &= \prod (e(g,g)^{rq_z(i)})^{\Delta_{i,s'_x(0)}} \\ &= e(g,g)^{rq_z(0)} \end{aligned}$$

(Here, Δ is Lagrange coefficient).
Following this recursive procedure the final result becomes $F_R = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$.
From this, the algorithm can decrypt the ciphertext and restore the original data content M by computing

$$\begin{aligned} \frac{\tilde{C}}{e(C,D)/F_R} &= \frac{M\omega^s}{e(h^s, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} \\ &= \frac{Me(g,g)^{\alpha s}}{e(g^{\beta s}, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} \\ &= M \end{aligned}$$

4 Security weaknesses in Koo *et al.*'s scheme [8]

In this section we discuss the limitation of the proposed scheme and the security flaw we have found in the scheme.

4.1 Limitation

The analysis of the scheme presented in [8] provides following results.

One of the security claim in the scheme is for sender anonymity. When analyzing the scheme for the same we have found the results as discussed below

- As per the scheme[8] the sender anonymity preserves sender's ID even after an intended receiver correctly decrypts a message encrypted by the sender. As per our view, this feature gives the same result as the other Anonymous Attribute Based Encryption schemes where sender does not send their ID at all with the encrypted message.
- The scheme requires every user to get an anonymous encryption key A_O built using the user's ID from trusted authority. But as per our analysis any user who knows the public parameters can encrypt a message and corresponding access tree using the public keys.
- Any adversary who has knowledge of public parameters $PK = (G,\ g,\ h = g^{\beta},\ \omega = e(g,g)^{\alpha})$ will choose a random element t from Z_p and generate his pseudonym as g^t that will be publicized. After encryption of the message as per the access policy depicted in Access tree $\mathcal{T}$, he scramble the attributes as $e(h, H(\lambda_j)^t) = e(g, H(\lambda_j))^{t\beta}$ $(\forall\ \lambda_j \in \mathcal{T})$. Now his ciphertext can be outsourced to public cloud for further search and retrieve procedure.
- In the scheme the authors have not provided any guidance as how an receiver will be able to know the sender's ID correctly. The authors have written that a receiver will get the pseudonym either from the data uploader or from CSP. In case when the data receivers are getting the list of pseudonyms from CSP, there is no way for receiver to identify as which pseudonym corresponds to which data owner. In other case when the data owner publicizes his pseudonym, then there is no option for data retriever to identify whether he receives the pseudonym from a legitimate data owner or it is from a fake user. Both the options creates a security flaw which we are discussing in section 4.2.
- Let every thing is going on a fine way assuming there is no adversary doing malfunctioning. Even then when a retriever wants to search for the documents for which his attributes satisfy the access policy, then he need to scramble his attributes with each of the pseudonym, and send all these requests to CSP. This practice increases the computation on both the retriever and CSP side.

The result of the analysis as discussed above proves that the only noticeable feature of the current scheme will be receiver anonymity. But in following section we further show that the scheme fails to achieve receiver anonymity also.

4.2 Security Flaw

Here we assume any user of the system or an outsider or the CSP as an attacker, who has knowledge of public parameters .

- An attacker generates a fake pseudonym let's say $P_O = g^t$ where $t \in_R Z_p$ and give it to data receiver. The attacker either directly gives P_O to receiver by publicizing it or the CSP provides this fake pseudonym when the data receiver demands for a list of pseudonyms.
- The receiver will prepare the search query after reforming his attributes as $K_{O,\Lambda_i} = \{\mathrm{e}(P_O,D'')\}_{j\in\Lambda_i} = \{\mathrm{e}(g^t,H(\lambda_j)^{\beta})\}_{j\in\Lambda_i} = \{\mathrm{e}(g,H(\lambda_j))^{\beta t}\}_{j\in\Lambda_i}$.

– Once the data receiver passes the search query, the attacker who is listening the query will fetch the K_{O,A_i} part from the query and compare it with a pre-computed list of scrambled attributes. The list contains an entry for each of the attributes in the universe as $\{e(h, H(\lambda_j))^t\}_{\lambda_j \in L} = \{e(g, H(\lambda_j))^{\beta t}\}_{\lambda_j \in L}$.
– A match with the j^{th} entry in the list indicates that the data receiver possess the corresponding attribute λ_j.

From the search result, the attacker determines the required attributes for decrypting a ciphertext document. Therefore, the receiver anonymity of a ciphertext document is lost.

5 Conclusion

In the paper we have discussed the searchable anonymous attribute based scheme presented by Koo *et al.*[8]. The authors of [8] have claimed to provide sender as well as receiver anonymity in attribute based encryption. However we have shown that if the sender's identity can not be revealed once the document is decrypted then the scheme behaves like other encryption schemes where the sender'ID is not sent along with an encrypted message. We have shown that for encryption of a message a data encryptor does not require the anonymous private key from the trusted authority. Only the knowledge of public parameters is sufficient to encrypt the message. Another major feature of the scheme in [8] as claimed by Koo *et al.* is receiver anonymity. In the paper we have shown how the scheme fails to provide receiver anonymity also. We have discussed the possibility of attack on scheme proposed by Koo *et al.* by an attacker. Our analysis shows that the scheme neither fulfills the security goal for providing receiver anonymity nor its sender anonymity feature adds any value in the scheme.

References

1. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption", *IEEE Symposium on Security and Privacy*, 2007.
2. V. Goyal, O. Pandey, A. Sahai, and B. Waters. "Attribute-based encryption for fine-grained access control of encrypted data", *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89-98, 2006.
3. Y. Zhang, X. Chen, J. Li, D. S. Wong, and H. Li, "Anonymous attribute-based encryption supporting efficient decryption test", *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 511-516, 2013.
4. A. Kapadia, P. P. Tsang, and S. W. Smith, "Attribute-Based Publishing with Hidden Credentials and Hidden Policies", *Network and Distributed System Security Symposium*, vol. 7, pp. 179-192, 2007.
5. S. Yu, K. Ren, and W. Lou, "Attribute-based content distribution with hidden policy" *4th Workshop on Secure Network Protocols, IEEE*, pp. 39-44 , 2008.
6. T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures", *Applied cryptography and network security*, pp. 111-129, 2008.

7. J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability", *Information Security*, pp. 347-362, 2009.
8. D. Koo, J. Hur, H. Yoon, "Secure and efficient data retrieval over encrypted data using attribute - based encryption in cloud storage",*Computers & Electrical Engineering*, pp. 34-46, 2013
9. A. Sahai, B. Waters. "Fuzzy identity-based encryption", *In Advances in CryptologyEUROCRYPT*, pp. 457-473. Springer Berlin Heidelberg, 2005.