On the Security of a Searchable Anonymous Attribute Based Encryption

Payal Chaudhari[1,2] and Manik Lal Das[1](✉)

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

Abstract. Ciphertext Policy Attribute-based Encryption(CP-ABE) is a public key primitive in which a user is able to decrypt a ciphertext if the attributes associated with secret key and the access policy connected with ciphertext matches. Although CP-ABE provides both confidentiality and fine-grained access control to the data stored in public cloud, anonymous CP-ABE adds interesting feature of sender and/or receiver anonymity. In this paper, we discuss a recent work on anonymous CP-ABE [1], which aims to provide secure and efficient data retrieval anonymously. We show that the scheme has major security weakness and does not ensure anonymity feature, which is the main claim of the scheme. We then present an improved scheme for mitigating the weaknesses of the scheme. The improved scheme retains the security claims of the original scheme [1] without adding any computation and communication overhead.

Keywords: Attribute based encryption · Anonymity · Confidentiality · Access structure

1 Introduction

Cloud computing is a comprehensive model, which provides on-demand computing resources such as storage, network, applications and services. Many enterprizes and individuals outsource their data to the cloud storage servers in order to reduce the cost for resource management. While making this flexibility to manage data in third party server, the security and privacy of data are major concerns. The outsourced data may contain sensitive information, such as Electronic Health Records(EHRs), financial details, personal photos etc. Therefore, data must be protected in the cloud storage server, so that unauthorized data access and data privacy protection need to be handled appropriately based on application requirement. There have been several approaches to securing data in cloud server. However, data encryption is a widely used primitive for securing data from authorized users. Before storing the data in cloud server, the data owner can encrypt the data so that the cloud server cannot learn anything from the stored data. Once the encrypted data are stored in the cloud server, two requirements become apparent for user convenience - *Access control* and

D. Giri et al. (Eds.): ICMC 2017, CCIS 655, pp. 16–25, 2017.
DOI: 10.1007/978-981-10-4642-1_2

Search over encrypted data. To provide a solution for secure and fine-grained data access, Sahai and Waters introduced the concept of attribute-based encryption (ABE) [2]. Ciphertext-Policy ABE(CP-ABE) [3] enables data encryption as per the access policy, where the access policy describes the combination of required attributes. User's secret key contains the attribute values which the user possesses. If the user's key matches with the access policy then he can decrypt the documents.

Although ABE scheme supports fine-grained access control [4], it discloses sender and/or receiver identity by which an adversary can guess the meaning or purpose of the message by seeing the attributes. Therefore, protecting sender and/or receiver identity while using ABE has been found a challenging research problem. In order to address this problem, anonymous ABE (AABE) schemes have been proposed in literature [5–9]. In anonymous CP-ABE, access policy is concealed inside the ciphertext components. A user tries to decrypt a ciphertext using the secret key made up with his attributes. If his attributes fulfills the access policy, then the decryption operation is successful. If the attributes included in the secret key do not match with the access policy, then the user can neither decrypt the ciphertext nor he can uncover the access policy hidden inside the ciphertext.

In 2013, Koo *et al.* [1] have proposed a searchable anonymous ABE scheme, where search on encrypted data is done on data owner's identity and data retriever's attributes. The scheme claimed that a user in the system can search on encrypted data stored in cloud with preserving sender and receiver anonymity. In this paper, we show that Koo *et al.*'s scheme fails to achieve the receiver anonymity [10]. We then propose an improved scheme, which mitigates the security flaw and retains the claimed security strength without adding any overhead.

The remaining of the paper is organized as follows. In Sect. 2, we give some preliminaries. In Sect. 3, we discuss Koo *et al*'s scheme. In Sect. 4, we show the security weaknesses of Koo *et al.*'s scheme. In Sect. 5, we present an improved scheme and provide its analysis in Sect. 6. We conclude the paper in Sect. 7.

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: $e(g, g) \neq 1$.
- Efficiency: The function e is efficiently computable.

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.

2.2 Access Tree

Access structure is represented in form of an access tree T. Each non-leaf node of the tree behaves as a threshold gate. It is defined as a tuple of its children and a threshold value. Let num_x denotes the number of children of a node x and k_x represents the threshold value of the node x, then $0 < k_x \leq num_x$. For an OR gate $k_x = 1$, and for an AND gate, $k_x = num_x$. Each leaf node x represents an attribute and threshold value $k_x = 1$. Each child of a parent will have unique index number from set [1,*num*] in an ordered fashion. To assist in traversing the access trees in cryptographic operations, following functions are being used.

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

The encryption algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree $\mathcal{T}$. The polynomial is chosen in a top-to-bottom fashion, initiating from the root node R. For each node x in the tree, the degree d_x of the polynomial $q_x = k_x - 1$, that is d_x is one less than the threshold value k_x of that node. For the root node R, the algorithm selects a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it picks d_R number of random points to define the polynomial q_R. For every other node x of access tree, it computes $q_x(0) = q_{parent}(x)(index(x))$ and selects d_x number of random points randomly to define a polynomial q_x.

3 Koo *et al.*'s Scheme

Koo *et al.* [1] proposed a scheme for secure and efficient data retrieval using anonymous attribute based encryption. The scheme works with the four entities as follows.

- Trusted Authority (TA), who generates user specific secret keys.
- Cloud service provider(CSP) is a semi-trusted entity where the users stored their data in encrypted form.
- Data owner/encryptor, who encrypts and stores the data in CSP.
- Data retriever/receiver, who issues queries to the CSP to access encrypted data from the cloud storage and retrieves the data only if his attributes satisfies the access policy specified by the data owner.

The scheme consists of five phases - System Setup, Key Generation, Encryption, Data Access and Decryption.

3.1 System Setup

The TA performs the setup. It chooses a bilinear group G of prime order p with generator g. It picks two random exponents α, β from Z_p and also selects a cryptographic hash function H: $\{0,1\}^* \rightarrow G$. TA computes the public parameter PK and master secret MK for the system as: $PK = \langle G, g, \omega = e(g,g)^{\alpha}, h = g^{\beta} \rangle$, $MK = \langle g^{\alpha}, \beta \rangle$.

3.2 Key Generation

Each data owner gets a secret key A_O from TA in which data owner identity is hidden. Each receiver gets a secret key SK from TA for decryption operation.

- For the data owner having identity ID_0, TA computes and returns him the anonymous key, $A_O = H(ID_0)^{\beta}$.
- The TA chooses a random $r \in Z_p$ for each individual user u_i in the system and $r_j \in Z_p$ for each attribute $\lambda_j \in \Lambda_i$. Here Λ_i is the set of attributes that belongs to user u_i. The private key SK is computed as

$$\begin{aligned} SK = \langle D = g^{\frac{(\alpha+r)}{\beta}}, \{D_j = g^r H(\lambda_j)^{r_j}, \\ D'_j = g^{r_j}, D''_j = H(\lambda_j)^{\beta}\}_{\lambda_j \in \Lambda_i}\rangle \end{aligned}$$

3.3 Encryption

Before uploading data content to cloud storage, the data owner having the identity ID_O computes his pseudonym as $P_O = H(ID_O)^t$. Here t is the random value selected by the data owner from Z_p. The data owner publicizes his pseudonym. To encrypt data M, the data owner runs **Encrypt** algorithm, as explained below. The encryption algorithm inputs the public parameter PK, its pseudonym P_O, a message M to be encrypted under the access tree $\mathcal{T}$, and outputs the ciphertext CT_0. After that, the attribute scrambling procedure, **AttrScm**, is applied to the ciphertext CT_0 for generating new ciphertext CT to be located in the cloud storage.

Data Encryption(Encrypt). This 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 as $k_x - 1$. The algorithm chooses a random $s \in Z_p$ and sets one point for polynomial q_R as (0, s). Rest of the d_R points are chosen randomly to completely define the polynomial q_R. For every other node x, the algorithm fixes $q_x(0) = q_{parent(x)}(index(x))$ and selects d_x number of random points to completely define a polynomial q_x. Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is built upon the basis of the access tree $\mathcal{T}$ as $CT' = (\mathcal{T}, \tilde{C} = M\omega^s, C = h^s, C'' = P_O, \{C_y = g^{q_y(0)}, C'_y = H(attr_y)^{q_y(0)}\}_{y \in Y})$.

Attribute Scrambling(AttrScm). In this phase the data owner garbles each attribute value included in $\mathcal{T}$ and obtains a new access tree $\mathcal{T}'$ by running **AttrScm(**CT_0,A_O,S**)**. S is the set of attributes which are included in the access policy of CT_0. $S = \{\lambda_i, \cdots, \lambda_k | 1 \le i \le k \le |L|\}$. For each attribute included in S, the data owner computes

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

and replaces the value of λ_x of every leaf node x related to $attr_x$ in $\mathcal{T}$ with the value of $scm_{att_x} \in K_{O,S}$. This results in the access tree $\mathcal{T}'$. The output of this algorithm is $CT = \langle \mathcal{T}', \tilde{C}, C, C'', \{C_y, C'_y\}_{y \in Y} \rangle$

At the end of this phase, the data owner uploads the CT on the cloud storage.

3.4 Data Access

This phase facilitates the retrieval of encrypted data from CSP.

- **Data query.** In the initial phase, a retriever can first gets a pseudonym list of data owners either from the CSP or directly from the data owners. Once the retriever determines to retrieve the data with $C'' = P_O$ from the cloud storage, it can generate cryptographic index terms for the attributes included in his secret key SK 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 that, the retriever submits his data request query in the form of a subset of these scrambled index information $K_{O,\Lambda'_i} \subseteq K_{O,\Lambda_i}$ to the CSP.
- **Data Retrieval.** After receiving search query in form of scrambled index terms K_{O,Λ'_i}, the CSP searches in his database if the requested item is present in the storage and if it is present then whether it is satisfied by the requested index attributes. This is done by the algorithm $\mathcal{C}(\mathcal{T},\ K_{O,\Lambda'_i})$. The algorithm returns *true* or *false*.

  Let T_x be a subtree of T with root node x and $X' = \{x' \in Y_x$ and parent(x') $= x\}$. $\mathcal{C}(\mathcal{T}, K_{O,\Lambda'_i})$ 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. The CSP sends the ciphertexts to the retriever for which the algorithm $\mathcal{C}(\mathcal{T}^i, K_{O,\Lambda'_i})$ returns true.

3.5 Decryption

When a retriever receives the requested content from CSP in encrypted form, then he applies the decryption algorithm DecryptNode on that encrypted content to obtain the plaintext.

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 invokes 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

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

(Here, Δ is Lagrange coefficient).

The decryption 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 Weaknesses in Koo *et al.*'s scheme

In the scheme [1], the attributes in the access policy are scrambled with a pseudonym computed by the data owner. The pseudonym hides the data owner(encryptor)'s identity. To fetch the documents from CSP the receiver requires the pseudonym. The receiver gets the pseudonym in either of these two ways:

1. a pseudonym directly from data owner.
2. a list of pseudonyms from the CSP.

If the receiver gets a pseudonym directly from the data owner then the receiver is knowing the data owner. The receiver scrambles his attributes using the pseudonym and retrieves the documents from the cloud as described in the **Retrieve** procedure. This compromises the anonymity of the sender. If the receiver gets a list of pseudonyms from the CSP, then following two cases arise.

(i) The receiver does not know which pseudonym refers to which data owner. Therefore, sender and receiver anonymity is preserved. However, concealing the sender identity from the receiver leads an attack as described later in this section.

(ii) It creates an operational overhead for the receiver when he gets a list of pseudonyms from the CSP and the receiver does not know which pseudonym refers to which data owner. The receiver can scramble his attributes either with all pseudonyms one-by-one and send them to the CSP or the receiver can select a subset of pseudonyms, scramble his attributes with each of the pseudonym from subset one-by-one and send the queries to the CSP.

The scheme requires every user to get an anonymous encryption key A_O from trusted authority. Then the user is able to encrypt and upload the documents on CSP. However, we show that a user who knows the public parameters can generate a pseudonym, encrypt a message and upload the document on CSP. A user who has the knowledge of the public parameters $PK = (G, g, h = g^{\beta}, \omega = e(g, g)^{\alpha})$ chooses a random element $t \in Z_p$, generates his pseudonym g^t and publishes it. The user scrambles the attributes included in $\mathcal{T}$ as $e(h, H(\lambda_j)^t) = e(g, H(\lambda_j))^{t\beta}$ $\forall$ $\lambda_j \in \mathcal{T}$. Now, this ciphertext can be uploaded to the CSP. Next, we show that the CSP can break the receiver anonymity, if he has the knowledge of the public parameter and attributes in the system. The CSP performs following steps to identify the attributes of a receiver who has submitted a search query to CSP.

CSP generates a fake pseudonym say $P_O = g^t$ for $1 \leq i \leq n$, where t is chosen randomly from Z_p. Using this fake pseudonym, CSP computes and prepares a list of scrambled attributes for each of the attribute in the system as follows. $\{e(h, H(\lambda_j))^t\}_{\lambda_j \in L} = \{e(g, H(\lambda_j))^{\beta t}\}_{\lambda_j \in L}$ for $1 \leq i \leq n$.

This list of values he stores in a set T'. When a data retriever $\mathcal{U}$ wants a list of pseudonyms from the CSP, then the CSP submits this list of pseudonyms in which the fake pseudonym generated by the CSP is also included. The data retriever $\mathcal{U}$ will not be able to detect if there is any fake pseudonym, as all pseudonyms are random values. Let us denote the list as L. The $\mathcal{U}$ chooses a subset L' of L, where $L' \subseteq L$. Then $\mathcal{U}$ scrambles his attributes using each of the pseudonym present in the L' as $K_{O_l,\Lambda_i} = \{\mathrm{e}(P_{Ol},D'')\}_{j\in\Lambda_i} = \{\mathrm{e}(g^{t_l},H(\lambda_j)^{\beta})\}_{j\in\Lambda_i} = \{\mathrm{e}(g,H(\lambda_j))^{\beta t_l}\}_{j\in\Lambda_i}$ using each pseudonym $P_{Ol} = g^{t_l}$ present in the set L'. $\mathcal{U}$ then submits these different sets of scrambled attributes $\langle K_{O_l,\Lambda_i}$ for each $P_{Ol} \in L'\rangle$ to the CSP. The CSP needs to compare each set

K_{O_l,Λ_i} with the set of pre-computed values T' that he has. Whenever he finds $K_{O_l,\Lambda_i} \subseteq T'$, then CSP identifies which attributes the $\mathcal{U}$ possesses. Once the CSP identifies the attributes of $\mathcal{U}$, then by comparing the remaining sets of scrambled attributes K_{O_l,Λ_i} with the stored access policies of other encrypted documents, the CSP can either uncover the hidden access policies of other encrypted documents. Therefore, the receiver anonymity of a ciphertext is revealed.

5 Improved Scheme

The security flaws in scheme [1] occur because of the use of pseudonym. We propose an improvement without using pseudonym, which retains the security claims of the scheme without increasing any overhead. The improved scheme has the following phases.

5.1 System Setup

The System setup phase is same as described in Sect. 3.1.

5.2 Key Generation

The Key Generation phase remains same as explained in Sect. 3.2. In addition to the Key Generation algorithm, the trusted authority publicizes a list of IDs and the mapping of IDs with the data owners owing that ID. We note that the secret parameter β scrambles the attributes in access policy, so the mapping of IDs with the data owners do not reveal any information about the sender and receiver of the encrypted documents stored in CSP.

5.3 Encryption

The data owner encrypts data M as per the access policy $\mathcal{T}$ by running the **Encrypt** algorithm as mentioned in Sect. 3.3. After that, the attribute scrambling algorithm, **AttrScm**, is applied to the ciphertext CT_0 for generating the ciphertext CT to be located in the cloud storage. We propose a modification in the **AttrScm** algorithm by removing the use of random value t. The data owner can use his secret encryption key for attribute scrambling as described below. S is the set of attributes to be included in access tree. For each attribute from set $S = \{\lambda_i, \cdots, \lambda_k \mid 1 \le i \le k \le |L|\}$, the data owner calculates

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

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 stored 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 storage managed by the CSP.

5.4 Data Access

Data query (Query). In this phase there is no need for a retriever to acquire a pseudonym of any data owner. When the retriever determines to retrieve a data with identity ID_O from the CSP then the retriever generates cryptographic index terms for corresponding attributes as

$$\begin{aligned} K_{O,\Lambda_i} &= \{e(D''_j, H(ID_O))\}_{j \in \Lambda_i} \\ &= \{e(H(ID_O), H(\lambda_j)^{\beta})\}_{j \in \Lambda_i} \\ &= \{e(H(ID_O), H(\lambda_j))^{\beta}\}_{j \in \Lambda_i} \end{aligned}$$

After that, the retriever submits the data request query in form of $K_{O,\Lambda'_i} \subseteq K_{O,\Lambda_i}$ to the CSP.

Data Retrieval (Retrieve). It is same as described in Sect. 3.4.

5.5 Decrypt

The decrypt operation is same as explained in Sect. 3.5.

6 Analysis

Theorem 1. The improved scheme provides sender and receiver anonymity.

Proof. We prove that the CSP or any other unintended receiver can not learn the sender or receiver identity. To break the sender and receiver anonymity the adversary needs to find out the value of sender's ID ID_O and λ_j from the scrambled attribute value $\{e(H(ID_O), H(\lambda_j)^{\beta})\}_{j \in \Lambda_i}$. For each of the attribute λ_j in the system and senders' identities $ID_{O,i}$, the following computed results are stored in CSP.

$$\{\{e(H(ID_{O,i}), H(\lambda_j))\}_{j \in \Lambda_i}\}_{1 \leq i \leq n}.$$

Here, n is the number of users in the system and it is assumed that every user possesses a unique identity and a set of attributes. To compare the scrambled attributes stored along with the ciphertext the adversary needs to get the value of β, where β is the master key of the system which the adversary can not get. The use of β prevents any unintended retriever to generate the scrambled attributes index terms for which he has not got the private key. The complexity of getting the value of β from the public parameter $h = g^{\beta}$ is equivalent to that of solving the discrete logarithm problem, which is an intractable problem. Therefore, the adversary can not learn the sender or receiver identity from the hidden access policy or from the search query because of scrambled attributes.□

In addition to the security strengths of the improved scheme, the scheme reduces the communication and computation overheads, as the receiver neither requires a pseudonym from data owner or from CSP nor uses it in attributes scrambling procedure.

7 Conclusion

Anonymous attributes based schemes provide interesting features such as sender and/or receiver anonymity, privacy-preserved data access and unlinkability. We discussed a recently proposed anonymous CP-ABE scheme, which claims secure and efficient data retrieval with sender and receiver anonymity. We showed that the scheme suffers from security weaknesses, lacks sender and receiver anonymity. We proposed an improved scheme by removing the use of pseudonym that mitigates the weaknesses of the scheme and retains the claimed security features intact without adding any communication and computation overhead.

References

1. Koo, D., Hur, J., Yoon, H.: Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage. Comput. Electr. Eng. **39**, 34–46 (2013)
2. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE Symposium on Security and Privacy (2007)
4. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
5. Zhang, Y., Chen, X., Li, J., Wong, D.S., Li, H.: Anonymous attribute-based encryption supporting efficient decryption test. In: Proceedings of the ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 511–516 (2013)
6. Kapadia, A., Tsang, P.P., Smith, S.W.: Attribute-based publishing with hidden credentials and hidden policies. In: Proceedings of Network and Distributed System Security Symposium, pp. 179–192 (2007)
7. Yu, S., Ren, K., Lou, W.: Attribute-based content distribution with hidden policy. In: Proceedings of the IEEE Workshop on Secure Network Protocols, pp. 39–44 (2008)
8. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden encryptor-specified access structures. In: Proceedings of Applied Cryptography and Network Security, pp. 111–129 (2008)
9. Li, J., Ren, K., Zhu, B., Wan, Z.: Privacy-aware attribute-based encryption with user accountability. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 347–362. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04474-8_28
10. Chaudhari, P., Das, M.L.: Cryptanalysis of searchable anonymous attribute based encryption. IACR Cryptology ePrint Archive 2016: 347 (2016)

http://www.springer.com/978-981-10-4641-4

Mathematics and Computing
Third International Conference, ICMC 2017, Haldia, India, January 17-21, 2017, Proceedings
Giri, D.; Mohapatra, R.N.; Begehr, H.; Obaidat, M.S. (Eds.)

2017, XX, 424 p. 51 illus., Softcover
ISBN: 978-981-10-4641-4