Enforcing Hidden Access Policy for Supporting Write Access in Cloud Storage Systems

Somchart Fugkeaw and Hiroyuki Sato
Department of Electrical Engineering and Information Systems, University of Tokyo, Tokyo, Japan

Keywords: CP-ABE, Access Control, Cloud Computing, Access Policy, Privacy, Policy Hiding.

Abstract: Ciphertext Policy Attribute-based Encryption (CP-ABE) is recognized as one of the most effective approaches for data access control solution in cloud computing. This is because it provides efficient key management based on user attributes of multiple users in accessing shared data. However, one of the major drawbacks of CP-ABE is the privacy of policy content. Furthermore, the communication and computation cost at data owner would be very expensive if there are frequent updates of data as those updated data need to be re-encrypted and uploaded back to the cloud. For the policy privacy perspective in CP-ABE based access control, access policy is usually applied to encrypt the plain data and is carried with the ciphertext. In a real-world system, policies may contain sensitive information that must be hidden from untrusted parties or even the users of the system. This paper proposes a flexible and secure policy hiding scheme that is capable to support policy content privacy preserving and secure policy sharing in multi-authority cloud storage systems. To address the policy privacy issue, we introduce randomized hash-based public attribute key validation to cryptographically protect the content of access policy and dynamically enforce hidden policies to collaborative users. In addition, we propose a write access enforcement mechanism based the proxy re-encryption method to enable optimized and secure file re-encryption. Finally, we present the security analysis and compare the access control and policy hiding features of our scheme and related works. The analysis shows that our proposed scheme is secure and efficient in practice and it also provides less complexity of cryptographic formulation for policy hiding compared to the related works.

1 INTRODUCTION

Traditional cryptography methods such as symmetric or public key encryption is considered to be inappropriate to be directly applied as an access control solution for cloud computing. Public key encryption solution provides multiple copies of ciphertext that will be shared among multiple users, while symmetric key encryption introduces the key distribution problem for a large scale of users.

In 2007, the Ciphertext Policy Attribute Based Encryption (CP-ABE) was proposed in [Bethencourt et al. 2017]. It is regarded as an effective solution for formulating a flexible access control enforcement to outsourced data and multiple decrypting parties. In CP-ABE, a set of attributes is assigned to each user in the way they are embedded into the user's secret key. The ciphertext is associated with the access policy structure in which encryptors or data owners can define the access policy by their own control. Users are able to decrypt a ciphertext if their attributes satisfy the ciphertext access structure.

To date, several works adopting CP-ABE (e.g., Chase, 2009, Wang et al., 2010, Wan et al., 2012, Li et al., 2012, Zhao et al., 2012, Yang et. al., 2014) for access control solutions generally concentrate on minimizing key management cost, optimizing computing cost of interaction between data owner and outsourced data storage, improving scalability and efficient user or attribute revocation. However, there are two additional significant requirements regarding policy privacy and policy sharing for supporting data encryption to the users having write privilege. First, access control policy privacy becomes more crucial for the attribute-based encryption model. This is because access policies may contain sensitive information which needs to be protected from unauthorized users. For example, in hospital information systems, a policy used to encrypt patient treatment files usually reveals the attributes of diseases, symptoms, or specialized

Fugkeaw, S. and Sato, H.
Enforcing Hidden Access Policy for Supporting Write Access in Cloud Storage Systems.
DOI: 10.5220/0006349605580564
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 530-536
ISBN: 978-989-758-243-1

treatment. Therefore, a way to hide policies must be introduced to solve this problem. Second, policy sharing for supporting users with write permission is required in CP-ABE data sharing scheme. In the CP-ABE scheme, the policy is used by data owners for data encryption. Nevertheless, in some cases, users may have permission to update files. To complete the updating, the file needs to be encrypted and loaded back to the cloud. This incurs both communication and computation cost at data owner side.

The aforementioned problems become more serious when the shared data is considered as sensitive and shared to a large number of collaborative users. In addition, advanced data sharing services require efficient and real-time data encryption and policy management. Existing solutions have not yet addressed these two problems consecutively.

In this paper, we extend the capability of our access control model called Collaborative-Ciphertext Policy-Attribute Role-based Encryption (C-CP-ARBE) proposed in (Fugkeaw et al., 2015) to achieve access policy privacy preserving and optimized cost of file re-encryption. To this end, we apply hash-based conversion and random encryption to protect the content of the policy and employ proxy re-encryption to optimize the performance of file re-encryption due to the data update.
This paper encompasses two major contributions as follows:

- We develop a new policy hiding scheme based on randomized hash-based public attribute key validation. Our proposed scheme is a lightweight crypto mechanism for policy hiding and policy enforcement.
- We introduce a write access enforcement mechanism based on the proxy re-encryption to optimize the cost of file re-encryption.

2 RELATED WORK

The originally proposed CP-ABE (Bethencourt et al. 2007) formulates the access in monotone tree-based structure. To support a more complex environment of cloud computing where there is multi-owner and multi-authority, multi-authority attribute based encryption (MA-ABE) solution have been proposed by several works [M.Chase, 2007, M. Chase et al., 2009, Yang et al., 2014, Fugkeaw et al., 2015]. Nevertheless, none of these approaches have taken policy privacy and write access enforcement into consideration.

Zhao et al. (Zhao et al., 2011) proposed a secure data sharing scheme based on a combination of CP-ABE and attribute-based signature (ABS). Their approach supports read and write access control. A signature-based called Tsign contains the ABS' verification attributes specifying the write privilege attributes. Users who need to update the file and load it back to the cloud need to be verified with the cloud server.

Ruj et al. (Ruj et al., 2012) proposed a privacy preserving authenticated access control for data outsourced in clouds. They propose distributed architecture for authentication and access control which supports multiple reads and writes on shared data. ABE and ABS are used to encrypt and sign the data respectively. Signatures of both read and write are verified in order to validate privileges. Nevertheless, the write privilege is treated separately from access policy specification. In addition, the access policy taken by the write access user is not hidden.

The attempt in dealing with the attribute-based policy hiding was introduced by (Katz et al., 2008). They proposed an inner predicate encryption. Nevertheless, the proposed scheme is based on the bilinear groups that require a product of three large primes leading the overhead.

The notion of CP-ABE policy hiding is introduced by (Nishide et al., 2008). They proposed two constructions of ciphertext-policy hiding supporting restricted access structures that can be expressed as AND gates on multi-valued attributes with wildcards. However, the scheme is found to be limited in given CP-ABE construction and selectively secure.

In (S. Yu et al., 2008), the access control approach with hidden policy for content distribution networks (CDNs) was proposed. The proposed scheme is based on the symmetric external Diffie-Hellman (SXDH) assumption between paired elliptic curve groups. Nevertheless, the designed scheme is not designed to support the multi-authority cloud systems.

In (J. Lai et al., 2011), they proposed a ciphertext policy hiding CP-ABE which is proven to be fully secure. Their approach is based on the inner-product predicate encryption (PE), which makes use of composite order bilinear groups. With the PE scheme, there is a fixed anonymity set applied to all ciphertext while the predicate is also encoded in the user keys. However, the complexity comes both composite order groups and the size of predicates.

3 BACKGROUND

This section describes the concept of CP-ABE and our proposed access control policy (ACP).

3.1 Cp-Abe

Basically, the concept of cryptographic construction and key generation used in CP-ABE is based on the bilinear map.

Definition1: Bilinear Map

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p and *e* be a bilinear map, $e : G_1 \times G_1 \rightarrow G_2$. Let *g* be a generator of G_1 . Let H: $\{0,1\}^* \rightarrow G_1$ be a hash function that the security model is in random oracle.

The bilinear map *e* has the following properties:

1. Bilinearity: for all u, v $\in G_1$ and a, b $\in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: e(g, g) $\neq 1$.

3.2 Access Control Policy (ACP)

Definition 2: Access Control Policy (ACP) ACP is an access tree-based structure. Let ACP *T* be a tree representing the access structure. Each non-leaf node of *T* represents the Role node where threshold gate is associated with. We denote by parent(*x*) the parent of the child node *x* in the tree. The function attr(*x*) is defined only for *x* in a leaf node of the tree as the set of attributes associated with *x*.

We also introduce a special attribute "privilege" as an extended leaf (EL) node of the ACP *T* in order to identify the read or write privilege.

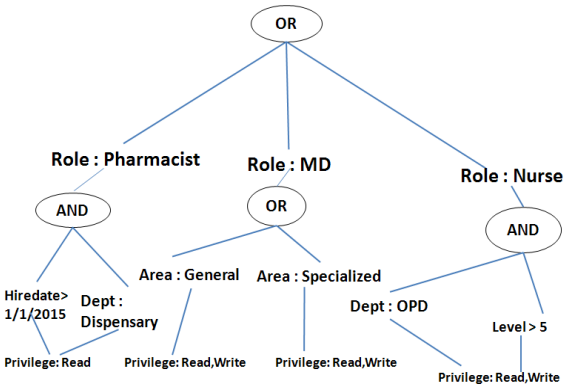

Figure 1: Access Policy Structure.

Figure 1 shows an example of the collaborative access control policy tree expressed in a patient treatment system. In the policy, there are three major roles pharmacist, MD, and nurse who are all allowed to access the prescription records with different privileges.

4 C-CP-ARBE CONSTRUCT

This section describes the major construction of our access control model [Fugkeaw et al., 2015].

Table 1 presents the notations and its description used in our proposed algorithms.

Table 1: Notations used in our access control.

Notation	Description
$S_{uid.aid}$	Set of all attributes issued to user *uid* and managed by authority *aid*.
SK_{aid}	a secret key which belongs to authority *aid*.
PK_{aid}	Public key which belongs to authority *aid.*
GSK_{uid}	A global secret key of a user *uid*. GSK is a private key issued by the certification authority CA.
$Cert_{uid}$	A public key certificate containing user's public key issued by a CA.
$PK_{x.aid}$	A public attribute key of attribute *x* issued by the authority *aid*
$UDK_{uid.aid}$	User Decryption key issued by authority *aid*
$EDK_{uid.aid}$	EDK is an encrypted form of a UDK which is encrypted by a user public key.
GRP	Group role parameter is a seed numbers computed from a set of user ids of the roles.
SS	Secret seal which is a symmetric key created from the AES algorithm together with the GRP.
ACP	An access control policy used to encrypt the data files.
SCT	A sealed ciphertext which is a ciphertext encrypted with the SS

Here, three major operational phases including System Setup, Key generation, Encryption, and Decryption.

Phase 1: System Setup

In the Setup phase, we selectively describe three algorithms as follows:

1. **Create Attribute Authority**(AA_{id})→PK_{aid}, SK_{aid}, $PK_{x.aid}$ The algorithm is based on the bilinear map. It takes the attribute authority ID (AA_{id}) as input. It outputs the authority public key (public parameter) PK_{aid}, Secret key SK_{aid}, and public attribute keys $PK_{x.aid}$ for all attributes issued by the AA_{aid}.
2. **Create GroupRole parameter GRP** (GSK_{uid}, set of RID → GRP) The Create GroupRole parameter algorithm takes input as a set of R_{ID}

and returns the GRP. Then, the GRP is signed (encrypted) by AA's private key GSK_{uid}.

3. **Create H-ACP(**ACP, $PK_{x.aid}$, H*(f)*), *R*, GSK_{uid}) → H-ACP. The algorithm consists of the following steps.

(1) The algorithm takes the public attribute key $PK_{x.aid}$ to be hashed in the hash function H*(f)*.
(2) These hash values are randomly applied with cryptographically pseudo random *R*, Then, set of randomized hash attribute value *R*(h($PK_{x.aid}$) is obtained.
(3) These hash attribute values *R*(h($PK_{x.aid}$) are used to construct the H-ACP as specified in the original ACP.
(4) H-ACP is signed by the data owner's private key, GSK_{uid}.

Phase 2: Key Generation

This phase consists of two algorithms as follows:

4. **UserKeyGen**(h($S_{uid,aid}$),SK_{aid},$Cert_{uid}$)→$EDK_{uid,aid}$**.** This algorithm takes continuous two steps as follows:

(1) takes input as set of randomized hash values of attributes Suid,aid associated to each hash-based access control policy (H-ACP), attribute authority's secret key SK_{aid}, and public key certificate of users $Cert_{uid}$ issued by the CA listed in the trust list, then it returns the set of user decryption key $UDK_{uid.aid}$.
(2) a $UDK_{uid.aid}$ is encrypted with the global public key of the user and outputs the set of encrypted decryption key $EDK_{uid,aid}$. The EDKuid.aid is stored in the cloud and it will be retrieved by the user upon the access request.

Phase 3: Encryption

This phase runs our two encryption layer protocol which accommodates data encryption and ciphertext encryption.

5. ENC(PK_{aid} {SS, GRP} M, H-ACP) →CT. The encryption algorithm performs two consecutive steps as followings:

- Inner layer: the algorithm takes as inputs authority public key PK_{aid}, H-ACP, and data M. Then it returns a ciphertext CT.
- Outer Layer: the algorithm takes GRP and generates AES session key as a secret seal SS to encrypt the ciphertext CT. It returns sealed ciphertext SCT. Finally, a SS is encrypted with user's public key $Cert_{uid}$, and stored in a cloud server.

Phase 4: Decryption

6. DEC(PK_{aid}, SCT, GSK_{uid}, $EDK_{uid.aid}$,) → M. The decryption algorithm performs two steps as follows:

(1) Decrypt the secret seal SS. The algorithm takes user's global secret key GSK_{uid} and then obtains the session key to decrypt the SCT and gets the CT.
(2) Decrypt the encrypted decryption key ($EDK_{uid.aid}$). The algorithm takes GSK_{uid} to decrypt $EDK_{uid.aid}$. Then $UDK_{uid.aid}$ is obtained to decrypt message M.

5 OUR PROPOSED POLICY HIDING SCHEME

5.1 Overview

In this section, we propose a policy hiding scheme that achieves policy privacy preservation and secure policy sharing to support encryption service to users having write permission. With our scheme, the policy can be enforced in unreadable format; the policy elements attached to the ciphertext is thus anonymized. Also, hidden policies can be conveniently shared to users having write privilege for data encryption without the availability of data owners. Figure 2 presents the system overview for policy outsourcing model in Hospital Information Systems (HIS).

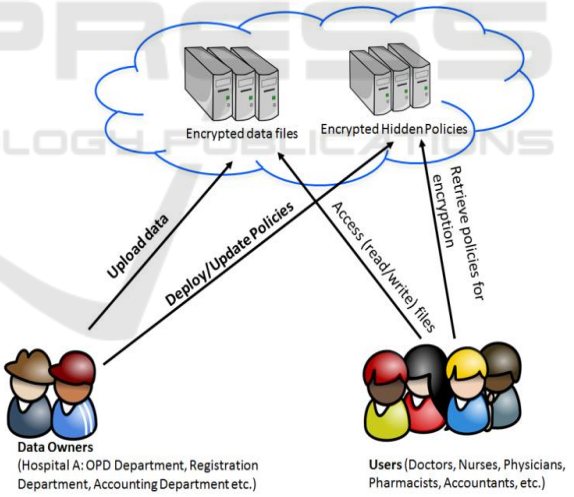

Figure 2: Policy outsourcing model in HIS.

As can be seen in Figure 2, data owners initially encrypt both data and access policies and then outsource to a cloud. Data files are encrypted based on the CP-ABE scheme while policy content is annoymized by the hash-based function and random encryption before it is encrypted with the CP-ABE. Both encrypted files and policies are then sent to be stored in the cloud servers. A server storing the encrypted policies can be only accessed by the data owners or users having write privilege. In our model,

users are grouped by their role, such as doctor, nurse, physicians, etc. These groups of user have different access privileges (read or write) for different files. Write privilege entails retrieval of encrypted policy for data encryption and re-uploading of files to the cloud server.

5.2 Policy Hiding

We introduce a randomized hash-based public attribute key validation (RH-PAKV) technique to conceal access control policy (ACP) content and validate policy rules. Our technique combines cryptographically pseudorandom number R (NIST, 2010) and cryptographic hash function to enable a digest to be more secure and resistant for attacks compared to normal hash. Generally, each attribute authority (AA) maintains the attributes issued together with their associated public attribute keys, and randomized hash value. Attributes with constant values (such as role name) will have their hashes generated from the hash-value-pair. Computable values (such as numeric, date/time) will only have their attribute names hashed. Table 2 shows the profile of hashed-based value of public attribute key. This profile table is locally maintained by the data owner.

Table 2: Public Attribute Keys and their hash values.

AttrID	Attribute	$PK_{x.aid}$	$h(PK_{x.aid})$
0001	Medical Doctor	b2xdx4512s	hash value1
0002	Dept: General	y245xdss03	hash value2
0003	Dept:Specialized	K45xds7ws	hash value3
0004	Level	S24512sdf	hash value4
…	…	….	…

When the hash value of each public attribute key ($PK_{x.aid}$) is obtained, a random is generated by the data owner and it will be applied to the individual hash result of $PK_{x.aid}$. Then, the randomized hash value $R(h(PK_{x.aid}))$ of each $PK_{x.aid}$ is obtained. In essence, we combine hash-based scheme and random number encryption for preserving the policy privacy because of two reasons as follows.

1. Compared to other encryption algorithms, hashing produces a lightweight output with fixed and small size of digest.
2. Since there is no key required, it is computationally feasible to verify the integrity of the policy content based on the hashed data.

Figure 3 shows the hash-based policy tree equivalent to the policy shown in Figure 1.

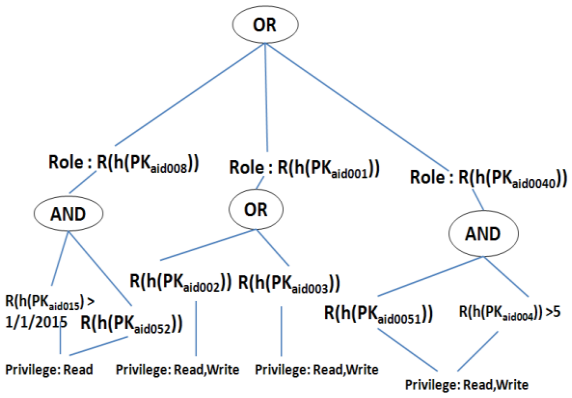

Figure 3: Hash-based ACP (H-ACP).

To enable a hash valued of attributes to be more secure and highly resistant for the attack. We apply secureRandom Java class (Java Platform Specification:SecureRandom Number Generator, 2016) as a cryptographically secure random number (R) to encrypt the hash of attribute value.

5.3 Write Access Enforcement

Our solution allows the users having write privilege can update the data file and retrieve the policy to re-encrypt the updated file. This avoids expensive communication cost for returning the file to be re-encrypted by the data owners and the encrypted files will be loaded back to the cloud server. With our solution, data owners encrypt a set of hashed policies H-ACPs with a simple CP-ABE policy where data owner's ID and a set of user IDs having write permission are included. Then the H-ACPs are sent to be stored in the cloud. Therefore, the permitted users with write privilege can retrieve the policy by using their current $UDK_{uid.aid}$ as it is used to re-encrypt the data. In our scheme, we delegate a proxy to perform file re-encryption initiated by the clients (users having write privilege) as a part of our write privilege enforcement mechanism. A proxy is a semi-trusted server located in the cloud environment. Even though the proxy is delegated to perform file re-encryption, it cannot access to the plaintext as the decryption key is not given. In our model, X.509 certificate and PKI key pair are issued to the proxy and users for the authentication purpose. The procedure of write privilege enforcement is shown below.

Write Privilege Enforcement

Input: UDK, ACP, Random *R*
Output: Updated file *M'*

Step 1: Data Access

1. A user requests to access the file.
2. A user is authenticated with her public key certificate. If authentication is successful, the capability list consisting data files she is able to access is presented.
3. A user downloads the chosen file and uses her $UDK_{uid.aid}$ to decrypt and update the file.
4. After the file is updated, user signs the updated content with her private key.

Step 2: Proxy Re-Encryption Initialization

5. User retrieves the encrypted ACP^c and decrypts it by using her $UDK_{uid.aid}$ to get the ACP.
6. The algorithm generates Random *R* which is used to encrypt the updated file, and ACP. Then, it returns M^R and ACP^R which are the random encryption result of file and ACP respectively.
7. User sets a passphrase to protect (encrypt) *R* and then the protected *R'* is generated.
8. User submits updated file *M'* and ACP^R, *R'* to the proxy.

Step 3: Re-Encryption

9. The proxy takes *M'*, *R'*, and ACP^R for file re-encryption.
10. In the PRE process, the algorithm asks the user to enter the passphrase to decrypt *R'* and performs file re-encryption.
11. The proxy sends the updated file *M'* to be stored in the cloud.

With our write enforcement mechanism, the users having write privilege can update the data and request for file re-encryption to the proxy without the intervention of data owners.

6 ANALYSIS OF OUR PROPOSED SCHEME

6.1 Security Analysis

For the security proof of our cryptographic construct, we refer to the proof of our inner encryption layer based on the CP-ABE encryption [Bethencourt et al, 2007]. Also, all access policies located in the cloud server are also encrypted based on the CP-ABE model.

Regarding the security protection in the re-encryption process done by a proxy, the plain data cannot be learned by the proxy or any parties as the re-encryption key is encrypted with the random number additionally protected by the passphrase defined by the user. The statistical details of cryptographic secure random number are given in (NIST, 2010).

In our model, the PRE process is executed instantly when the proxy gets the request for file re-encryption. Therefore, our core access control model, policy hiding and outsourcing scheme, and proxy re-encryption method are secure.

6.2 Comparative Analysis of Access Control Features of Existing Works

In this section, we give a comparative analysis of policy hiding features provided by J. Lai et al. scheme, Asghar et al. scheme, and our scheme. Both Lai and Asghar access control scheme are based on CP-ABE and RBAC model respectively. Thus, they are suitable to be compared with our model based on the integration of CP-ABE and RBAC. Table 3 presents the comparison of existing CP-ABE access control schemes supporting policy hiding.

Table 3: Comparison of Policy Hiding Schemes.

Scheme	Write Access	Cost of encryption	Policy Hiding
Lai et al.	No	Composite Bilinear map	CP-ABE Inner Predicate
Asghar et al.	Yes	2 round of encryptions + re-encryption	RBAC encryption
Our Scheme	Yes	AES, and bilinear maps	CP-ABE attribute key hashing

Importantly, our scheme generally supports multi-authority access policy enforcement and provides write access control, while Lai et al.'s scheme does not support these features. Regarding the computation analysis, our scheme significantly provides less cost for encryption than both compared schemes since our scheme uses symmetric encryption and general bilinear map based on CP-ABE. In addition, our hidden policy is pre-computed based on the hashing technique and possesses less complexity compared to the inner predicate computation.

In Lai scheme (Lai et al., 2011), the composite bilinear groups with the inner predicate encryption could introduce the performance problem for encryption, decryption, and policy update when a policy consists of a high number of attributes and various logical operations.

In Asghar method (Asghar et al, 2013), encrypting the RBAC policy element could make the communication and computation overhead high if the policy size is big and there is frequent update on the policy. The client-side workload is thus an issue for this scheme.

In our scheme, an equality-based hashing scheme yields less computation cost which is similar to the original CP-ABE. With our scheme, data owners also do not need to stay online to support encryption service. Therefore, in addition to achieving policy privacy, our solution offers secure policy retrieval for

data encryption. Users with write permission can retrieve the policy but they cannot learn the policy content.

7 CONCLUSION AND FUTURE WORK

We have presented a privacy-preserving access control model in collaborative cloud data storage systems. We introduce the policy hiding scheme as an integrative solution for enhancing the capability of our access control scheme C-CP-ARBE. The proposed hash-based policy enforcement compliments limitation of the traditional CP-ABE in terms of policy privacy. Significantly, our scheme does not require the process of de-anonymization of policy and the encryption is done as the same as plain policy encryption. Finally, we analyze the access control features of related works and present the comparative analysis of our method and two related works.

For future works, we will conduct a larger scale of experiments and evaluate the performance of the proposed system in the real cloud environment such as CloudStack. We will also investigate the cloud forensics and auditing techniques to guarantee the accountability of user access and integrity of the data and policy outsourced.

REFERENCES

Bethencourt, J., Sahai, A., and Waters B., 2007. Ciphertext-policy Attribute-based Encryption, In IEEE Symposium of Security and privacy, SP'07, IEEE, pages 321.-334.

Chase, M., 2007. Multi-authority attribute based encryption, In Proceedings of the 4th Theory of Cryptography Conference on Theory of Cryptography (TCC'07), Springer, pages 525-534.

Nishide, T., Yoneyama, K., and Ohta, K., 2008. Attribute-based encryption with partially hidden encryptor-specified access structures. In Proceedings of Applied Cryptography and Network Security, ACNS'08. LCNS, Vol.5037, pages 111-129. Springer.

Katz, J., Sahai, A., and Waters, B., 2008. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, Eurocrypt 2008. LNCS, Vol 4965. pages 146-162, Springer.

Yu, S., Ren, K., and Lou., W., 2008. Attribute-based content distribution with hidden policy. In Proceedings of 4th Workshop on Secure Network Protocols, NPSEC 2008. IEEE.

Chase, M. and Chow, M. 2009. Improving privacy and security in multi-authority attribute-based encryption, In Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, pages 121-130, ACM.

NIST. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". NIST, Special Publication April 2010.

Wang, G., Liu, Q., and Wu, J., 2010. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In Proceedings of the 17th ACM Conference on Computer and Communications Security,CCS '10, pages 735-737, ACM.

Zhao, F., Nishide, T., and Sakurai, K., 2011. Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems, In Proceedings of 7th International Conference of Information Security Practice and Experience, ISPEC'11, papges 83-97, Springer.

Lai, J., Deng, R. H., and Li, Y., 2011. Fully Secure Cipertext-Policy Hiding CP-ABE, In Proceedings of the 7th International Conference on Information Security Practice and Experience, ISPEC'11, pages 24-39, Springer.

Wan, Z., Liu, J., and Deng, R. H., 2012. HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing. In IEEE Transactions on Information Forensics and Security 7(2): pages 743-754, IEEE.

Li, M., Yu, S., Zheng, Y., Ren, K, and Lou, W., 2012. Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption, In IEEE Transactions on Parallel and Distributed Systems. Volume 24, Issue: 1, pages 131-143, IEEE.

Ruj, S., Stojmenovic, M., and Nayak, A., 2012. Privacy Preserving Access Control with Authentication for Securing Data in Clouds, In Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, pages 556-563, IEEE.

Asghar, M. R., Ion, M., Russello, G., and Crispo, B., 2013. ESPOONERBAC: Enforcing Security Policies in Outsourced Environments, In Elsevier Jounral of Computers & Security (COSE), Volume 35, pages 2-24. Elsevier Advanced Technology Publications.

Yang, K., Jia, X., Ren, K., Zhang, B., Xie, R., 2014. Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage. IEEE Transastions on Parallel Distributed Systems, Vol. 25(7), pages 1735-1744, IEEE.

Fugkeaw, S. and H. Sato, 2015. An extended CP-ABE based Access control model for data outsourced in the cloud, In Proceedings of IEEE International Workshop on Middleware for Cyber Security, Cloud Computing and Internetworking, MidCCI 2015, IEEE.

Java™ Platform, Standard Edition 7 API Specification, 2016. : Secure random number generator Java library, https://docs.oracle.com/javase/7/docs/api/java/security/ SecureRandom.html