International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967
Volume 2 , Issue 1(Jan-Feb 2015), PP51-57

Time-Based Proxy Re-encryption Review

Meghatai Maruti Bhoite[1], I Govardhan Rao [2]
CSE, University College of Engineering Osmania University, Hyderabad , India

Abstract :- Fundamentals approach for secure data sharing in a cloud environment is to let the data owner encrypt data before outsourcing. To achieve fine-grained access control on encrypted data and scalable user revocation, existing work combines attribute-based encryption (ABE) and proxy re-encryption (PRE) to delegate the cloud service provider (CSP) to execute re-encryption. However, the data owner should be online in order to send the PRE keys to the CSP in a timely fashion, to prevent the revoked user from accessing the future data. The delay of issuing the PRE keys may cause potential security risks. In this survey paper, we discuss a time-based proxy re-encryption (TimePRE) scheme to allow a user's access right to expire automatically after a predetermined period of time. In this case, the data owner can be offline in the process of user revocations. The basic idea is to incorporate the concept of time into the combination of ABE and PRE. Specifically, each data is associated with an attribute-based access structure and an access time, and each user is identified by a set of attributes and a set of eligible time periods which denote the period of validity of the user's access right. Then, the data owner and the CSP are required to share a root secret. Therefore, given the re-encrypted ciphertext, only the users whose attributes satisfy the access structure and whose access rights are effective in the access time can recover corresponding data.

Keywords: - Cloud computing; time; proxy re-encryption; attribute-based encryption

I. INTRODUCTION

The use of cloud computing is increasingly popular due to the potential cost savings from outsourcing data to the *cloud service provider* (CSP). One technique to protect the data from a possible untrusted CSP is for the data owner to encrypt the outsourced data . Flexible encryption schemes such as *attribute based encryption* (ABE) can be adopted to provide fine grained access control. ABE allows data to be encrypted using an *access structure* comprised of different *attributes*. Instead of specific decryption keys for specific files, users are issued attribute keys. Users must have the necessary attributes that satisfy the access structure in order to decrypt a file. For example, for data which is encrypted with the access structure *{*(*Student AND CIS*) *OR Staff}*, either users with attributes Student and CIS, or users with attribute Staff, can recover data. The key problem of storing encrypted data in the cloud lies in *revoking* access rights from users. A user whose permission is revoked will still retain the keys issued earlier, and thus can still decrypt data in the cloud. A na¨ıve solution is to let the data owner immediately re-encrypt the data, so that the revoked users cannot decrypt the data using their old keys, while distributing the new keys to the remaining authorized users. This solution will lead to a performance bottleneck, especially when there are frequent user revocations. An alternative solution is to apply the *proxy re-encryption* (PRE) technique .This approach takes advantage of the abundant resources in a cloud by delegating the cloud to re-encrypt data . This approach is also called command- driven re-encryption scheme, where cloud servers execute re-encryption while receiving commands from the data owner. A cloud is essentially a large scale distributed system where a data owner's data is replicated over multiple servers for high availability. As a distributed system, the cloud will experience failures common to such systems, such as server crashes and network outages. As a result, re-encryption commands sent by the data owner may not propagate to all of
the servers in a timely fashion, thus creating security risks. To illustrate, let us consider a cloud environment shown in Fig. 1, where the data owner's data is stored on cloud servers *CS*1*, CS*2*, CS*3*, CS*4. Assume that the data owner issues to *CS*4 a re-encryption command, which should be propagated to *CS*1*, CS*2*, CS*3. Due to a network outage, *CS*2 did not receive the command, and did not re-encrypt the data. At this time, if revoked users query *CS*2, they can obtain the old cipher text, and can decrypt it using their old keys. The access structure is specified by the data owner, but the access time is updated by the CSP with the time of receiving an access request. The data can be recovered by only the users whose attributes satisfies the access structure and whose

access rights are effective in the access time. To enables the CSP to update the access time automatically, first express *actual time* as *a time tree*. The height of the time tree can be changed as required. For ease of presentation, only consider a three layer time tree as shown in Fig 2. (a), where time is accurate to the day and the time tree is classified into three layers in order: year, month, and day. Here use (y, m, d), (y, m),

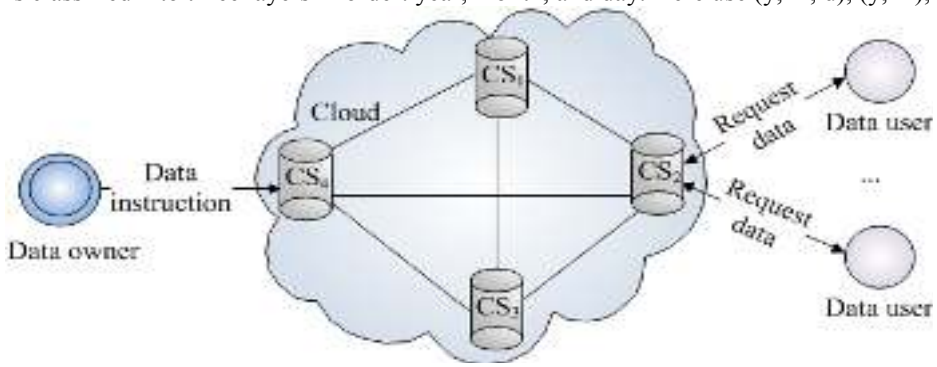

Fig 1. A typical cloud environment

A better solution is to allow each cloud server to *independently* re-encrypt data without receiving any command from the data owner. In this paper, a *Timebased Proxy re-encryption scheme (*TimePRE scheme for short) used. TimePRE is a time-based Proxy re-encryption scheme, which allows each cloud server to automatically re-encrypt data based on its internal clock. The basic idea of the TimePRE scheme is to associate the data with an *access control Structure* and an *access time*. Each user is issued keys associated with *attributes* and *attribute effective times*. The data can be decrypted by the users using the keys with attributes satisfying the access control structure, and attribute effective times satisfying the access time. Unlike the command-driven re-encryption scheme, the data owner and the CSP share a root secret key, with which each cloud server can re-encrypt data by updating the data access time according to its own internal clock.

II. RELATED WORK

In the cloud computing model, the data owners have to entrust sensitive data to a remote cloud, which is maintained by an external party, i.e., the cloud service provider (CSP). Rather than fully trusting the CSP, existing research proposed to only outsource encrypted data to the cloud. Our work is on scalable user revocation ,Proxy Re-encryption and Fine Grained Access Control. The related work as follows:

2.1 User Revocation

User revocation is a well studied, but non-trivial task. The key problem is that the revoked users still retain the keys issued earlier, and thus can still decrypt ciphertexts. Therefore, whenever a user is revoked, the re-keying and re-encryption operations need to be executed by the data owner to prevent the revoked user from accessing the future data. For example, when ABE is adopted to encrypt data, the work in reference [5] proposed to require the data owner to periodically re-encrypt the data, and re-distribute new keys to authorized users. This approach is very inefficient due to the heavy workload introduced on the data owner. A better solution is to let the data owner delegate a third party to execute some computational intensive tasks, e.g, re-encryption, while leaking the least information. Proxy re-encryption [2, 3] is a good choice, where a semi-trusted proxy is able to convert a ciphertext that can be decrypted by Alice into another ciphertext that can be decrypted by Bob, without knowing the underlying data and user secret keys. For example, the work in reference [6] is the first to combine KP-ABE and PRE to delegate most of the computation tasks involved in user revocation to the CSP. Our previous work [7] is the first to combine PRE and a CP-ABE system (HABE) to achieve a scalable revocation mechanism in cloud computing. The work in reference [8] that supports attribute revocation may be applicable to a cloud environment. This approach requires that once a user is revoked from a system, the data owner should send PRE keys to the CSP, with which the CSP can be delegated to execute re-encryption. The main problem of this approach is that the data owner should be online in order to send the PRE keys to the CSP in a timely fashion, to prevent the revoked user from accessing the data. The delay of issuing PRE keys may cause potential security risks.

2.2 Proxy Re-Encryption

Proxy Re-Encryption (PRE) is a cryptographic primitive in which a semi-trusted proxy is able to convert a cipher text encrypted under 'Alice' public key into another cipher text that can be opened by 'Bob' private key without seeing the underlying plaintext. More formally, a PRE scheme allows the proxy, given the proxy re-encryption key rka, to translate cipher texts under public key pk_a into cipher texts under public key pk_b and vice versa. Let us illustrate the motivation of the PRE scheme by the following example: 'Alice' receives emails encrypted under her public key PK_A via a semi trusted mail server. When she leaves for vacation, she wants to share her secret key SKA with him. The PRE scheme allows 'Alice' to provide a PRE key $RK_{A->B}$ to the mail server, with which the mail server can convert a cipher text that is encrypted under Alice's public key PKA into another cipher text that can be decrypted by Bob's secrete key SKB, without seeing the underlying plaintext, SK_A, and SK_B.

2.3 Fine-Grained Access Control on Encrypted Data

To protect data security from an untrusted server, the work in reference [9] adopts traditional symmetric key cryptographic system to encrypt data. Before outsourcing, the data owner will first classify data with similar access control lists (ACLs) into a file-group, and then encrypt each file-group with a symmetric key. The symmetric key will be distributed to the users in the ACL, so that only the users in the ACL can access this group of files. The main drawback of this approach is that the key size managed by the data owner grows linearly with the number of file-groups. Another approach is proposed by [14], which is based on the combination of traditional symmetric key and public key cryptographic systems. The data owner first specifies an ACL for a data, and then encrypts the data with a symmetric key, which is encrypted with the public keys of users in the ACL. Therefore, only the users in the ACL can use their secret keys to recover the symmetric key, and then use the symmetric key to recover the data. The main drawback of this approach is that the costs for encrypting a data will grow linearly with the number of users in the ACL. Therefore, an ideal approach is to encrypt each data once, and distribute appropriate keys to users once, so that each user can only decrypt his authorized data. Attribute-based encryption (ABE) [10], which has developed to two branches, key-policy ABE (KP-ABE) [11] and ciphertext-policy ABE (CP-ABE) [12, 13], is a promising cryptographic technique having such a property. This flexibility makes ABE an attractive choice when selecting an encryption scheme for cloud computing.

III. PRELIMINARY

3.1 Outline of TimePRE Scheme

3.1.1 Main Idea

The main idea of the TimePRE scheme is to incorporate the concept of time into the combination of HABE and PRE. Intuitively, each user is identified by a set of *attributes* and a set of *effective time periods* that denotes how long the user is eligible for these attributes, i.e., the period of validity of the user's access right. The data accessed by the users is associated with an *attribute-based access structure* and *an access time*. The access structure is specified by the data owner, but the access time is updated by the CSP with the time of receiving an access request. The data can be recovered by only the users whose attributes satisfies the access structure and whose access rights are effective in the access time. and (y) to denote a particular day, month, and year, respectively. For example (2014, 4, 5) denotes April 5,2014. The access time associated with a data corresponds to a leaf node in the time tree, and the effective time periods associated with a user correspond to a set of nodes in the time tree. If there is a node corresponding to a

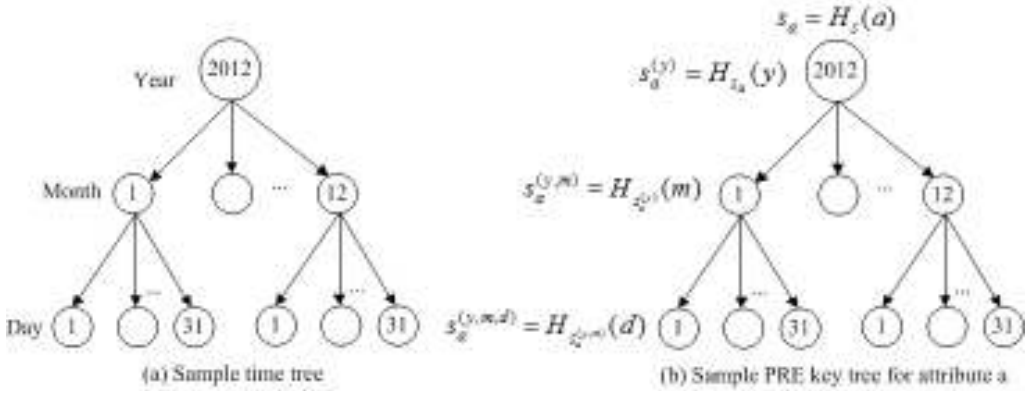

Fig 2: three-level time tree and attribute *a*'s PRE key tree.

effective time period that is ancestor of the node corresponding to the access time, then the user's access right is effective in the access time.

Then, allow the data owner and the CSP to share a root secret key s in advances, with which the CSP can calculate required PRE keys based on its own time and re-encrypt corresponding cipher text, automatically. Specifically, at any time, each attribute a is associated with one initial public key PK_a, and three time-based public keys: day-based public key $PK_a^{(y,m,d)}$, month-based public key $PK_a^{(y,m),}$ and year-based public key $PK_a^{(y)}$,each of which denotes a's public key in a particular day (y, m,d), month (y, m), and year (y)respectively. For example, given current time (2014, 4,5), attribute a's public keys include PK_a, $PK_a^{(2014,4,5)}$, $PK_a^{(2014,4)}$, and $PK_a^{(2014)}$. In the TimePRE scheme, the original cipher text is encrypted by the data owner using initial public keys of attributes in the data access structure. On receiving a request, the CSP first uses the root secret key s to calculate PRE keys on all attributes in the access structure based on its own time, and then uses these PRE keys to re-encrypt the original cipher text by updating the initial public keys of all attributes in the access structure to time-based public keys.

Here, use sa(y), sa(y, m), and sa(y, m, d) to denote the PRE keys on attribute a in time (y), (y, m),and (y, m, d), which can be used to update attribute a's initial public key PK_a to time-based public keys $PK_a^{(y)}$, $PK_a^{(y,\ m)}$, and $PK_a^{(y,\ m,\ d)}$, respectively. Since the PRE key used in this scheme is derived from a root secret key and the current access time, use different notations as those shown in Fig.2 (b), for each attribute a, the CSP can use the root secret key s and the time tree to hierarchically calculate the time-based PRE keys with Equation 1:

$$S_a^{(y)} = H_{sa}(y) \quad (1a)$$

$$S_a^{(y,m)} = H_{sa}^{(y)}(m) \quad (1b)$$

$$S_a^{(y,\ m,\ d)} = H_{sa}^{(y,m)}(d) \quad (1c)$$

Where $s_a = H_s(a)$,a, y, m, d$\in$ { 0, 1}* is a string corresponding to a specific attribute, year, month, and day; and H_s, H_{sa}, $H_{sa}^{(y)}$, $H_{sa}^{(y,m)}$: {0, 1}$*$ -> Z^*_q are hash functions with indexes s, s_a, $s_a^{(y)}$, and $s_a^{(y,m)}$, respectively.

Furthermore, to incorporate the concept of time to HABE, each user is granted with a set of *time-based user attribute secret keys* (UAK). Each time-based UAK is associated with a user, an attribute, and an effective time period. If user *u* is eligible for attribute *a* in day (y, m, d), the data owner first uses the root secret key *s* to obtain day-based attribute public key $PK_a^{(y,\ m,d)}$ from initial attribute public key PK_a, and then uses $PK_a^{(y,\ m,\ d)}$ to generate a day-based UAK $SK_{u,a}^{(y,\ m,\ d)}$ for user *u*. The same situation holds for the case that user *u* is eligible for attribute *a* in a month (*y,m*) or a year (*y*).

3.2. Definition of the TimePRE scheme

3.2.1 Setup(K,UA) -> (PK,MK, s) : The data owner takes a sufficiently large security parameter *K* as input to generate the system public key PK, the system master key MK, and the root secret key *s*. The system public key will be published, the system master key will be kept secret, and the root secret key will be sent to the CSP.

3.2.2 GenKey(PK,MK, s, PK_u, a, T_u) -> (SK_u, $SKT_{u,a}^{Tu}$) : Suppose that user u with public key PK*u* is eligible for attribute *a* and his access right is effective in time Tu. The data owner uses the system

public key PK, the system master key MK, the root secret key s, user public key PKu, attribute *a*, and effective time period Tu to generates user identity secret key (UIK) SKu and time-based user attribute secret key(UAK) $SKT_{u,a}^{Tu}$ for u.

3.2.3 Encrypt(PK, A, F) -> (CA) : The data owner takes a DNF access structure A, a data F, and system public key *PK*, e.g., initial public keys of all attributes in the access structure $\{PK_a\}_{a\in A}$ as inputs to output a cipher text C_A.

3.2.4 Re-encrypt(CA, PK, s, t) -> (C^t_A) : Given a cipher text *C*A with structure A, the CSP first uses the system public key *PK* and the root secret key *s* to generate PRE keys on all attributes in the access structure A based on the access time *t*, and then uses these PRE keys to re-encrypt the original cipher text CA to C^t_A.

3.2.5 Decrypt(PK,C^t_A, SK_u, { $SKT_{u,a}^{Tu}$ }$a\subseteq A;T_u\subseteq t$) -> (F) : User u, whose attributes satisfy the access structure A, and whose effective time period Tu satisfy the access time t, can use SK_u and { $SKT_{u,a}^{Tu}$ }$a\subseteq A$ to recover F from C^t_A.

3.3 System Modules

3.3.1 Data owner initialization*:* The data owner runs the *Setup* function to initiate the system. When the data owner wants to upload file *F* to the cloud server, it first defines an access control A for *F*, and then determines the current time slice TS_i. Finally, it runs the *Encrypt* function with A and TS_i to output the ciphertext. When the data owner wants to grant a set of attributes in a period of time to data user Alice, it runs the *GenKey* function with *attributes* and *effective times* to generate keys for Alice.

3.3.2 Data user read data*:* When data user Alice wants to access file *F* at TS_i, she sends a read command$R(F)$ to the cloud server, where *F* is the file name. On receiving the read command $R(F)$, the cloud server runs the *REncrypt* function to re-encrypt the file with TS_i. On receiving the ciphertext, Alice runs the *Decrypt* function using keys satisfying A and TS_i to recover *F*.

3.3.3 Data owner write data*:* When the data owner wants towrite file *F* at TS_i, it will send a write command to the cloud server in the form of: $W(F, seqnum)$, where $seqnum$ is the order of the write command. This $seqnum$ is necessary for ordering when the data owner issues multiple write commands that have to take place in one time slice. On receiving the write command, the cloud server will commit it at the end of TS_i.

3.4 Algorithm 1 (asynchronized clock with delays)
while Receive a write command $W(F, t_{i+1}, seqnum)$ **do**
if Current time is earlier than $t_{i+1} + \alpha$ **then**
Build Window *i* for file *F*
Commit the write command in Window *i* at $t_{i+1} + \alpha$
else
Reject the write command
Inform the data owner to send write command earlier
while Receive a read request $R(F, TS_i)$ **do**
if Current time is later than $t_{i+1} + \alpha$ **then**
Re-encrypt the file in Window *i* with TS_i
else
Hold on the read command until $t_{i+1} + \alpha$

3.4.1 Protocol Description We let the data owner and the cloud server agree on a maximal waiting time α. Thus, the cloud server will wait until $ti + \alpha$ to commit the write commands that should be committed at *ti*, and to respond to the read commands for reading data at *TSi*. The data owner and data user will include additional information in their write or read commands. When the data owner wants to update the file *F* at *TSi*, he will issue a command *W*(*F, ti*+1*, seqnum*), where *F* is the file name,*ti*+1 is when the updates have to take place and *seqnu* is the order of the write command. When the data user wants to read the file *F* at *TSi*, he will use a command *R*(*F, TSi*).Then, we need to determine the maximal time difference between the data owner and the cloud server. We denote this time difference as _, where _ is no larger than the duration of one time slice. In

other words, when the data owner is at *TSi*, the cloud server's time may be *TSi*−1, *TSi*, or *TSi*+1. We let the data owner issue his write command before *ti*+1 when he wants this update to be reflected in *TSi*+1. Algorithm 1 shows the actions of the cloud server.

IV. ANALYSIS

4 .1 Performance Analysis

The efficiency of the *Setup* algorithms is rather straightforward. Therefore,we only analyze the costs introduced by algorithms *GenKey*, *Encrypt*, *ReEncrypt*, and *Decrypt*. If a user is identified by *n* attributes and his effective time periods correspond to *m* nodes in the time tree, the *GenKey* algorithm requires the data owner to execute *O*(*mn*) point multiplications to generate secret keys of *O*(*mn*) length. However, if we deliberately design the time tree, *m* can be limited to a relatively small value. To recover *F*,a user, whose attributes satisfy the access structure and whose access right is effective in the access time, needs to execute *O*(1) bilinear map operations.
Then, we briefly compare our scheme with the work in references [6,7], which also allow the CSP to be delegated to execute re-encryption. We first consider the workload on the data owner in the user revocation, as shown in Table 1, where *n* is the number of attributes associated with a user and *w* is the number of remaining authorized users. In the TimePRE scheme,the data owner has nothing to do when a user is revoked. The *ReEncrypt* algorithm run by the CSP is without any involvement of the data owner.
Then, we compare the re-encryption costs incurred at the CSP, as shown in Table 2, where *n* is the number of attributes associated with a revoked user and *N* is the number of conjunctive clauses in an access structure.To re-encrypt a ciphertext *C*A based on the access time *t* = (*y; m; d*), our scheme requires the CSP to execute *O*(6*N*) exponentiation operations and one bilinear map operation to re-encrypt a ciphertext. Since the CSP can batch the re-encryption operations and re-encrypt data only when receiving a data access request, the re-encryption cost is relatively low.

TABLE 1: COMPARISONS OF RE-ENCRYPTION COST ON DATA OWNER

Properties	Reference [6]	Reference [7]	Our Scheme
Number of PRE keys	*O*(*n*)	*O*(*n*)	0
Number of update keys	*O*(*nw*)	*O*(*nw*)	0

TABLE 2: COMPARISONS OF RE-ENCRYPTION COST ON CSP

Properties	Reference [6]	Reference [7]	Our Scheme
Exp	*O*(*n*)	*O*(*n*)	*O*(6*N*)
Map	*O*(0)	*O*(0)	*O*(1)

4.2 Security Analysis

The *Encrypt* algorithm in the TimePRE scheme is the same as the *Encryption* algorithm in HABE, which has been proven to be semantically secure in [7]. Therefore, we consider that the TimePRE scheme is secure if the following propositions hold:

4.2.1 Proposition 1. The keys produced by the *GenKey* algorithm are secure.

4.2.2 Proposition 2. The ciphertext produced by the *ReEncrypt* algorithm is semantically secure.

4.2.3 Proposition 3. Given the root secret key and the original ciphertext, the CSP cannot know neither the underlying data, nor UAKs while executing re-encryption.

V CONCLUSION

In this paper, the Time based Proxy Re-encryption scheme to achieve fine grained access control and

scalable user revocation in a cloud environment. Our scheme enables each user's access right to be effective in a pre-determined period of time, and enable the CSP to re-encrypt cipher texts automatically, based on its own time. Thus, the data owner can be offline in the process of user revocations. The main problem with this scheme is that it requires the effective time periods to be the same for all attributes associated with a user. Our future work is to allow different attributes associated with a user, without increasing the number of UAKs associated with each user.

REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attributebased encryption. In Proceedings of IEEE Symposium on Security and Privacy (SP), pages 321–334, 2007..

[2] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT),pages 127–144, 1998.

[3] M. Green and G. Ateniese. Identity-based proxy re-encryption. In Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS), pages 288–306, 2007.

[4] Q. Liu, C.C. Tan, J. Wu, and G. Wang. Reliable re-encryption in unreliable clouds. In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), 2011.

[5] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attributebased systems. Journal of Computer Security, 18(5):799–837, 2010.

[6] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 534–542, 2010.

[7] G. Wang, Q. Liu, J. Wu, and M. Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers.Computers & Security, 30(5):320–331, 2011.

[8] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), pages 261–270, 2010.

[9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In Proceedings of the USENIX Conference on File and Storage Technologies (FAST), pages 29–42, 2003.

[10] A. Sahai and B.Waters. Fuzzy identity-based encryption. In Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), pages 557–557, 2005.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the ACM Conference on Computer and Communications Security(CCS), pages 89–98, 2006.

[12] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attributebased encryption. In Proceedings of IEEE Symposium on Security and Privacy (SP), pages 321–334, 2007.

[13] S. M¨uller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In Proceedings of Annual International Conference on Information Security and Cryptology (ICISC), pages 20–36, 2009.