POSTER: Temporal Attribute-Based Encryption in Clouds

Yan Zhu[1,2], Hongxin Hu[3], Gail-Joon Ahn[3], Xiaorui Gong[1,2], Shimin Chen[1,2]
[1]Institute of Computer Science and Technology, Peking University, Beijing 100871, China
[2]Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing 100871, China
[3]Laboratory of Security Engineering for Future Computing (SEFCOM), Arizona State University, Tempe, AZ 85287, USA
{yan.zhu,smchen}@pku.edu.cn,{hxhu,gahn}@asu.edu

ABSTRACT

There has been little work that explores cryptographic temporal constraints, especially for data sharing in cloud computing. In this paper, we present a temporal attribute-based encryption (TABE) scheme to implement temporal constraints for data access control in clouds. This scheme has a constant size for ciphertext, private-key, and a nearly linear-time complexity. In addition, we implement a prototype system to evaluate our proposed approach. Our experimental results not only validate the effectiveness of our scheme and algorithms, but also show our scheme has better performance for integer comparison than BSW's bitwise comparison scheme.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption

General Terms

Security

Keywords

Access Control, Cryptography, Temporal, Integer Comparison, Attribute-Based Encryption

1. INTRODUCTION

Cloud computing provides an extensible and powerful environment for growing amounts of services and data by means of on-demand self-service. However, it is also facing many challenges for data security as the users outsource their sensitive data to clouds, which are generally beyond the same trusted domain as data owners. To address such a problem, access control is considered as one of critical security mechanisms for data protection in cloud applications. Unfortunately, traditional access control approaches usually assume that data is stored in trusted data servers for all users. This assumption however no longer holds in cloud computing since a data owner and cloud servers are very likely to be in different domains. Hence, attribute-based access control [1, 3] has been adopted for the cloud computing so that the outsourced sensitive data would be encrypted based on access policy on attributes for such data while only authorized users can decrypt and access the data. However, prior work could not provide a complete temporal control for cloud applications.

CCS'11, October 17–21, 2011, Chicago, Illinois, USA.
ACM 978-1-4503-0948-6/11/10.

Temporal dimension has generated a great amount of interest in security community as an important property of access control for security system management in recent years [2]. For example, we assume that a small IT company, which has many retailers distributed across US, constructs their data centers using attribute-based encryption (ABE) on a platform as a service (PaaS) environment, e.g., Google's App Engine or Amazon's Elastic. For the employee's working hours, Table 1 illustrates a simple schedule for some employees, which consists of five attributes: Period-of-Validity, which is a time attribute on month basis; Job, which is a string attribute to denote employee's position; Day and Hour, which are two period attributes to denote working days and hours, respectively.

Table 1: Lists for employee's working hours.

People	Period-of-Validity	Job	Day	Hour
Anderaon	2010/01-2012/06	Manager,	Mon.-Fri.	9:00-14:00
Grant	2011/04-2011/12	Accountant	Thu.-Fri.	10:00-16:00
Kidman	2011/04-2012/06	Engineer	Mon.-Fri.	9:00-16:00
Coolidge	2011/01-2011/12	Retailer	Mon.-Wed.	9:00-16:00
Jones	2010/08-2011/12	Retailer	Mon.-Sat.	10:00-17:00

In this example, a manager wishes to define access policies to protect each component of their business information: Technique Archive, Sales Record, Salary History, Service Log, and Contact Information. An access policy can be viewed as a description of attributes, which is used to match the attribute values in the employee's private key. As illustrated in Table 2, these attributes describe various functions and temporal constraints for business information.

Table 2: Schedule for data sharing in clouds.

Files	Period-of-Validity	Job	Day	Hour
Tech. Archive	2010/11-2011/03	Engineer	Mon.-Fri.	9:00-16:00
		Manager	(All)	16:00-9:00
Sales Record	2011/01-2011/05	Accountant	Thu.-Fri.	(All)
Salary History	2011/05-2011/11	Manager	Mon.-Fri.	9:00-16:00
Service Log	2010/06-2011/04	Retail	Mon.-Fri.	9:00-16:00
		Engineer	(All)	16:00-9:00
Contact Info.	2010/11-2011/05	(All)	Mon.-Sat.	9:00-16:00

A document, like a technique archive, reposits the core technology of latest products, which is being manufactured in 2010/11∼2011/03. Its policy therefore stipulates that only engineers can view this document during regular working hours (during 9:00∼16:00 from Monday to Friday), as well as managers can view it at other times. Another example is the sales record from 2011/01 to 2011/05, which can be accessed by accountants from Thursday to Friday. The following are the specifications of corresponding policies:

Technique Archive: *(2010/11 ≤ Period-of-Validity ≤ 2011/03)* ***AND*** *(((Engineer)* ***AND*** *(Monday≤ Day ≤ Friday)* ***AND*** *(9:00*

$\leq$ *Hour* $\leq$ *16:00))* **OR** *((Manager)* **AND** *((16:00* $\leq$ *Hour* $\leq$ *23:59)* **OR** *(0:00* $\leq$ *Hour* $\leq$ *9:00))))*

Sales Record: *(2011/01* $\leq$ *Period-of-Validity* $\leq$ *2011/05)* **AND** *(Accountant)* **AND** *(Thursday*$\leq$ *Day* $\leq$ *Friday)*

Unfortunately, existing ABE schemes cannot meet the requirements of temporal access control addressed in the above example due to the reason that these schemes cannot support integer range comparison, which is necessary to represent temporal constraints. In fact, Bethencourt *et al.* [1] has perceived this topic from a naive point of view on integer comparison and provided a bitwise-comparison method (called as BSW's scheme) to realize a pretty simple control, e.g. $a < 11$, but this method does not support range expressions in user's private key because both "*1*" and "*0*" may appear in the same bit position. This means that we are not able to specify a temporal licence in user's private key. This obstructs such a method to achieve fine-grained temporal access control. In addition, the complexity of bitwise comparison should influence the efficiency of temporal constraint in practical applications in clouds.

In this paper, we address temporal access control in clouds, along with an efficient cryptographic framework to support temporal constraints. We first define the range of integers to extend the power of attribute expression in access policy and user's private key, and then propose a temporal attribute-based encryption (TABE) scheme to implement various temporal constraints to regulate data sharing in clouds. This scheme has a constant size for ciphertext, private-key, as well as a nearly linear-time complexity. In addition, we implement a prototype of TABE system to evaluate our proposed approaches, and our experimental results show the effectiveness and efficiency of our approach.

This paper is organized as follows. In Section 2, we propose our TABE scheme with the definition of integer comparison. Then, we evaluate the performance of our scheme in Section 3. Finally, we conclude this paper in Section 4.

2. TABE FOR CLOUD COMPUTING

For the sake of clarity, we introduce the following notations:

- $\mathcal{A}$: the set of attributes $\mathcal{A} = \{A_1, \cdots, A_m\}$;
- $A_k(t_i, t_j)$: the range constraint of attribute A_k on $[t_i, t_j]$, i.e., $t_i \leq A_k \leq t_j$;
- $\mathcal{P}$: the access control policy expressed as a Boolean function on AND/OR logical operations, generated by the grammar: $\mathcal{P} ::= A_k(t_i, t_j) | \mathcal{P}$ AND $\mathcal{P} | \mathcal{P}$ OR $\mathcal{P}$;
- $\mathcal{L}$: the access privilege assigned to the user's licence, generated by $\mathcal{L} ::= \{A_k(t_a, t_b)\}_{A_k \in \mathcal{A}}$;
- $PK_{\mathcal{A}}$: the public key over $\mathcal{A}$;
- $SK_{\mathcal{L}}$: the private key with $\mathcal{L}$;
- MK: the master key presided by system managers;
- $\mathcal{H}_{\mathcal{P}}$: the ciphertext header over $\mathcal{P}$;
- ek: the session key used to encrypt the data by symmetrical encryption scheme.

The definitions of $\mathcal{P}$ and $\mathcal{L}$ can meet the basic requirements of dual temporal expressions.

2.1 TABE Framework

We focus on the temporal access control and encryption process in cloud computing. TABE consists of four algorithms as follows:

- Setup($1^{\kappa}, \mathcal{A}$): Takes a security parameter κ as input, outputs the master key MK and the public-key $PK_{\mathcal{A}}$;
- GenKey($MK, u_k, \mathcal{L}$): Takes the user's ID number u_k as input, the access privilege $\mathcal{L}$ and MK, outputs the user's private key $SK_{\mathcal{L}}$;
- Encrypt($PK_{\mathcal{A}}, \mathcal{P}$): Takes a temporal access policy $\mathcal{P}$ and PK as input, outputs the ciphertext header $\mathcal{H}_{\mathcal{P}}$ and a random session key ek; and
- Decrypt($SK_{\mathcal{L}}, \mathcal{H}_P$): Takes a user's private key $SK_{\mathcal{L}}$, and a ciphertext header $\mathcal{H}_{\mathcal{P}}$ as input, outputs a session key ek;

For the sake of brevity, the encryption on data is not shown in this framework since data owners could easily employ traditional symmetric key cryptography to encrypt and then outsource data with the help of a random session key. This framework is based on BSW's scheme [1], in which both AND/OR operations and basic fine-grained access control are not within the scope of this paper.

2.2 Security Models

In attribute-based access control, we define policies with temporal constraints to specify the exact intervals during which an event can be enabled or disabled by matching the user's licence. Generally, we denote the time as a set of integers U, and the permitted time of access policy $U_{\mathcal{P}}$ is also denoted as a subset of U. Assume that the access privilege in the licence is $U_{\mathcal{L}} \subseteq U$. We have the following definition:

DEFINITION 1. *Given a set of time U, a user is authorized if and only if $U_{\mathcal{P}} \bigcap U_{\mathcal{L}} \neq \emptyset$, where $U_{\mathcal{P}} \subseteq U$ is the permitted time for an access policy and $U_{\mathcal{L}} \subseteq U$ is the granted time in the user's licence.*

Note that, this definition is not strict enough for some access control models due to the reason that there may exist a time t in which a user is authorized but ($t \notin U_{\mathcal{P}}$ and $t \in U_{\mathcal{L}}$) or ($t \in U_{\mathcal{P}}$ and $t \notin U_{\mathcal{L}}$) where $U_{\mathcal{P}} \bigcap U_{\mathcal{L}} \neq \emptyset$. However, it is acceptable for real-world applications since a user may store her/his previously accessed data.

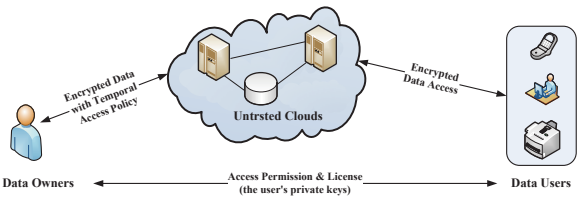

Figure 1: Temporal attribute-based encryption for cloud computing.

Based on our TABE framework, we construct a cloud-based data sharing service involving three different entities as depicted in Fig. 1: data owner, cloud server, and some data users (e.g., computers, mobile devices, or general equipments). In order to implement temporal access control, a data owner has a collection of data to be outsourced to cloud servers in an encrypted form of Encrypt($PK_{\mathcal{A}}, \mathcal{P}$). At all times the data owner can assign a private key $SK_{\mathcal{L}}$ to data users by using GenKey($MK, u_k, \mathcal{L}$). To access these data, the data users download data from cloud servers and then decrypt them by utilizing Decrypt($SK_{\mathcal{L}}, \mathcal{H}_P$).

3. PERFORMANCE EVALUATION

We have implemented our scheme in Qt/C++ and our experiments were run on an Intel Core 2 processor with 2.16 GHz and 500M of RAM on Windows Server 2003. All disk operations were performed on a 1.82TB RAID 5 disk array. Using GMP and PBC libraries, we have implemented a cryptographic library (called as PKUSMC) upon which temporal attribute systems can be constructed. This C library contains approximately 5,200 lines of code and has been tested on both Windows and Linux platforms.

In order to demonstrate the advantages of our scheme, we compared the performance of BSW's scheme and our scheme over integer ranges. For the comparable comparison, we also implemented a version of BSW's scheme in the same development environment. In the red part of Figure 2, we show the computational overheads for BSW's bitwise comparison scheme under the different sizes of U. Given a comparison range U and a random comparison operation $t_1 \leq t \leq t_2$ in U, the user's secret key includes $2\log_2|U|$ bit-attributes to describe the attributes t_1 and t_2, each bit of which is represented with a '***1***'-type attribute or a '***0***'-type attribute. For a certain integer t, the ciphertext describes the relation $t_1 \leq t$ and $t \leq t_2$ by using two policy trees, each of which is a tree with $\log_2|U|$ attribute leaves and at most $\log_2|U|$ depth. Correspondingly, the decryption can be realized by $2\log_2|U|$ bilinear map operations. Hence, the computational overheads of three algorithms (KeyGen, Encrypt, and Decrypt) increase proportionally to the size of U.

Figure 2 shows the computational overheads of BSW's scheme and our scheme for different sizes of U. It is obviously noticeable that our scheme is more efficient than BSW's bitwise comparison scheme. The reason is that the computation costs of algebraic operations and simple modular arithmetic operations can be neglected, because they run fast enough in contrast with bilinear map operations. Without loss of generality, the performance of our scheme is better than that of BSW's scheme in $[1; 10,000,000]$ [1]. Especially, the performance is little affected by the change of comparison ranges [1;100,000].

Table 3: Comparison of BSW's scheme and our scheme for integer comparison.

	BSW's Scheme		Our Scheme	
	$t_1 \leq t$	$t \leq t_2$	$t_1 \leq t$	$t \leq t_2$
Ciphertext size	$\log_2\|U\|$	$\log_2\|U\|$	1	1
Private-key size	$\log_2\|U\|$	$\log_2\|U\|$	1	1
Depth of policy tree	$\log_2\|U\|$	$\log_2\|U\|$	1	1
Computation overhead	$\log_2\|U\|$	$\log_2\|U\|$	1	1

Next, we analyze the storage and communication overheads of our TABE scheme. Because of forward (or backward) derivation function for total ordering, TABE scheme has $O(1)$ size of private-key and ciphertext for a certain integer attribute. Assuming that BSW's scheme and our scheme are constructed on bilinear group systems with the same length, we show the comparison of BSW's scheme and our scheme for integer comparison in Figure 3. Although the extra space is used for TABE scheme to store the table $\{w_{t_i}, \bar{w}_{t_i}\}_{t_i \in U}$, it is easy to find that TABE scheme has a

[1]Note that, the semicolon in the internal is used to distinguish from the thousands separator.

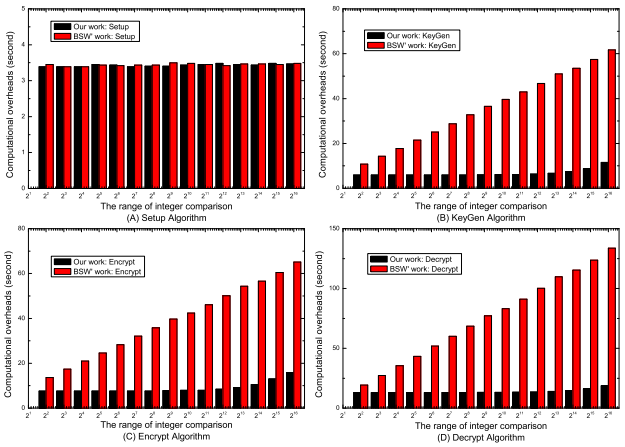

Figure 2: Computational overheads of BSW's scheme (Red) and our scheme (Black) for integer comparison operations: (a) Setup algorithm, (b) KegGen algorithm, (c) Encrypt algorithm, and (d) Decrypt algorithm.

constant size for ciphertext, private-key, and depth of policy-tree, as well as a nearly linear-time complexity. For a comparison range $[1, Z]$, the storage, communication and computation costs of BSW's scheme are nearly $O(\log_2 Z)$ times higher than those of our scheme. Hence, in comparison with BSW's scheme, TABE scheme provides a lower bound on various aspects including storage, communication and computation overheads.

4. CONCLUSIONS

In this paper, we addressed the construction of temporal access control for data sharing in clouds. Based on the integer range expression of attributes, we proposed a novel temporal attribute-based encryption to support integer comparisons and temporal constraints. We also evaluated the performance of our scheme by comparing with BSW's bitwise scheme.

Acknowledgments

The work of Y. Zhu, X. Gong and S. Chen was partially granted by the Chinese NDRC InfoSec Foundation under Project "Mobile Internet Mal-behavior Detection Platform based on Cloud Computing(2010)". This work of G.-J. Ahn and H. Hu was partially supported by the grants from US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360) and Department of Energy (DE-SC0004308).

5. REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[2] J. B. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17:4–23, 2005.

[3] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, pages 534–542, 2010.