Analysis of a Secure and Verifiable Policy Update Outsourcing Scheme for Big Data Access Control in the Cloud

Wei Yuan

State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. Email: yuanwei@iie.ac.cn

Abstract. How to flexibly change the access policy after the initial data access policy has been set is a critical problem to promote attribute-based encryption (ABE) from a theoretical tool to a practical tool. Since the first ABE scheme emerges, many schemes have been proposed to solve the problem but the problem remains unsolved yet. The reason is that the overheads of changing an old access policy to a new one are larger than that of generating a ciphertext with the new access policy directly. Recently, in IEEE Transactions on Parallel and Distributed Systems (DOI:10.1109/TPDS.2014.2380373), Yang et al. proposed a multi-authority ciphertext-policy (CP) ABE scheme with ciphertext updating function. The authors declared that the access policy of the ciphertext can be dynamically modified with the old ciphertext and the scheme is correct, complete, secure, and efficient. However, after revisiting this paper, we found that the scheme is not correct under the system model defined by the authors. Some necessary algorithms are missing such that users cannot decrypt the updated ciphertexts. Moreover, if new algorithms are added into the system model to ensure that the scheme is correct, complete, and secure, the scheme will be not as efficient as the authors declared. Consequently, the scheme fails to achieve the claimed results.

Keywords: Access Control, Attribute-based Encryption, Policy Updating

1 Introduction

Attribute-based encryption is a promising technique to control access policy for the data stored on remote cloud servers. However, when the data access policy needs to be updated, the efficiency of current policy update methods [1, 2] cannot satisfy practical demands. Data owners have to retrieve their data from the cloud servers and re-encrypt them under the new access policy, and then upload them back to the cloud server. Many computation and communication resources are wasted in this procedure. Recently, Yang et al. proposed a new ciphertext update scheme [3] based on a multi-authority CP-ABE scheme [4]. The authors declared that their scheme enable efficient access control with dynamic policy updating. The ciphertext corresponding to the new access policy can be directly transformed from the old ciphertext with the old and the new access policies and the scheme is correct, complete, secure, and efficient. However, they do not provide the strict proof to show the claimed properties. Only some informal explanations were given. In this paper, we show that the scheme proposed in [3] is not correct under its system model. It lacks necessary algorithms to achieve the claimed properties. If the necessary algorithms are added, the scheme will be not as efficient as the authors declared.

2 Brief Review of the Scheme

There are four kinds of entities in the scheme [3]: Authorities, the cloud server, data owners and users.

An authority generates public/secret key pairs for the attributes in its domain and provides secret keys for users. The cloud server stores data and updates ciphertexts for changing the data access policy. A data owner generates ciphertexts for his/her data and provides ciphertext update keys for the cloud server. A user is able to freely get ciphertexts from the cloud server and tries to obtain corresponding data with his/her own secret key.

The scheme includes 5 phases: System Initialization, Key Generation, Data Encryption, Data Decryption and Policy Updating.

During the system initialization phase, the global setup algorithm and the authority setup algorithm are executed. The global setup algorithm outputs the global parameters for all the other algorithms. Then the authority setup algorithm generates the public/secret key pairs of the authority. During the key generation phase, the key generation algorithm is executed to output a secret key for each user. During the data encryption phase, the encryption algorithm is executed to generate ciphertext from the data under the given access policy. During the data decryption phase, the ciphertext is decrypted to recover the data. Above 4 phases are the same as in [4].

During the policy updating phase, the update key generation algorithm and the ciphertext updating algorithm are executed. The update key generation algorithm produces the update key with the old and the new access policies as well as some random numbers kept in the encryption phase and then the update key is used as the input of the ciphertext updating algorithm to update the ciphertext. Due to the limited space, we do not list the details of [3] in this paper.

3 Analysis of the Scheme

First, the scheme is not correct under the current system model defined in section 2. In the data encryption phase described in section 3.4, a group of constants $(c_i) \in \mathbb{Z}_p$ should be chosen to ensure $\sum_i c_i M_i = (1, 0, \cdots, 0)$ such that the decryption result is correct. If the ciphertext is generated by the encryption algorithm, the condition is easy to be satisfied. For the ciphertext generated by the ciphertext updating algorithm, the authors declared that the constant c_{n+1}, which corresponds to the newly added attribute x_{n+1}, can be obtained by $c_{n+1} = c_j/a_m$ for the Attr2OR operation and can be obtained by $c_{n+1} = -c_j/a_m$ for the Attr2AND operation in section 6.1. However, a_m is a random number chosen by the data owner as a part of the update key UK_m for the data m (described in section 4.1). In section 2, the system model clearly indicates that UK_m is generated by the data owner with the update key generation algorithm and transmitted to the cloud server as the input of the ciphertext updating algorithm. That is to say, users do not know a_m. As a result, users cannot compute c_{n+1} and decrypt the updated ciphertexts with the decryption algorithm defined in section 3.4. The correctness proof in section 6.1 fails under the system model defined in section 2.

Second, the system model is not complete. In the encryption algorithm of [4], $\sum_i c_i M_i = (1, 0, \cdots, 0)$ is to ensure $\sum_i c_i \lambda_i = s$ and $\sum_i c_i w_i = 0$. The encryption algorithm in [4] is directly re-used in [3] and the policy updating algorithm is constructed on this ciphertext structure. The operation Attr2OR adds a new ciphertext component C_{n+1} corresponding to the new attribute x_{n+1}. The random number a_m and two new shares $\lambda_{n+1} = a_m \cdot \lambda_j$ and $w_{n+1} = a_m \cdot w_j$ are introduced to generate C_{n+1}. To let $c_j \lambda_j$ equal $c_{n+1} \lambda_{n+1}$ and $c_j w_j$ equal $c_{n+1} w_{n+1}$ such that the decryption succeeds, the authors define $c_{n+1} = c_j / a_m$. Similarly, the authors define $c_{n+1} = -c_j / a_m$ to let the decryption succeed for the operation Attr2AND. However, users do not know the random number and new shares. Consequently, a user-key update key transmission algorithm is missing in the system model.

Suppose a ciphertext is generated with an access policy ($\mathbb{A}$ *and* x_j) by the encryption algorithm and a user Alice is exactly able to decrypt the ciphertext. If the data owner adds an attribute x_{n+1} to the access policy with Attr2AND or adds an attribute x_{n+1} to the access policy with Attr2OR and then removes the attribute x_j from the policy with AttrRmOR, Alice should lose the ability to decrypt the new ciphertext. However, if Alice receives UK_m as the cloud server, she is able to transform the new ciphertext component C_{n+1} back to C_j with UK_m and thus recovers her decryption ability. Then, the scheme is not secure. That is to say, the user-key update key should be different from the ciphertext update key UK_m. Therefore, a user-key update key generation algorithm and a secure ciphertext update key transmission algorithm are missing in the system model.

Actually, the original constants (c_i) are decided by the access policy and the attribute set in LSS schemes [5]. These constants are chosen independently. However, the added constants are dependent on some existing constants in [3]. Thus, the updated ciphertext should be given a clear indication that which attributes are updated. In such a way that users are able to know which constants should be obtained from the access policy and the attribute set and which constants should be obtained from existing constants. As a result, another decryption algorithm to decrypt the ciphertexts generated by the ciphertext updating algorithm is missing in the system model.

Third, the scheme is not as efficient as the authors declared. If the authors modify the system model and add the algorithms mentioned above such that the correctness condition is satisfied, the efficiency of the scheme should be in the same level as the attribute-based proxy re-encryption (ABPRE) schemes [1, 2]. The difference is that ABPRE mainly updates ciphertext while the scheme in [3] mainly updates secret key. ABPRE produces ciphertext update keys by means of user secret keys and generates new ciphertexts according to the ciphertext update keys. Then the user secret keys do not change. The scheme in [3] only generates a ciphertext update key and updates the ciphertext employing it. But each league user needs a new user-key update key to decrypt the new ciphertext. That is to say, each time the ciphertext is updated in [3], a new ciphertext update key should be given to the cloud server and new user-key update keys should be given to all the league users in the system. To avoid the overheads of transmitting and re-encrypting the ciphertext, many update keys (in the same level of the number of the users in the system) need generating and transmitting. Since transmitting update keys needs private channels between the data

owner and the users, the efficiency of the scheme is not better than that of the previous methods. We believe that the scheme cannot be efficient as the authors declared even if it can be modified to satisfy the claimed properties of correctness, completeness, and security.

4 Conclusions

We analyzed the CP-ABE scheme proposed by Yang et al. and showed that their scheme is not correct. Even if the scheme can be modified to satisfy the claimed properties of correctness, completeness, and security, it cannot be efficient as the authors declared. Thus, the problem of flexibly changing access policy after the initial data access policy has been set is still open.

References

1. K.T. Liang, L.M. Fang, W. Susilo, D. S. Wong, "A Ciphertext-Policy Attribute-Based Proxy Re-encryption with Chosen-Ciphertext Security," INCoS 2013: 552-559.
2. S. Yu, C. Wang, K. Ren, "Attribute Based Data Sharing with Attribute Revocation," ACM ASIACCS 2010,pp.261-270, 2010.
3. K. Yang, X.H. Jia, K. Ren, "Secure and Verifiable Policy Update Outsourcing for Big Data Access Control in the Cloud," IEEE Trans. Parallel Distrib. Syst., vol.26, no.12, pp.3461-3470, 2015.
4. A. Lewko and B. Waters, "Decentralizing Attribute-Based Encryption," EUROCRYPT 2011, LNCS 6632, pp.568-588, 2011.
5. A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution," PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.