Private Key Delegation in Attribute-Based Encryption

Máté Horváth
Laboratory of Cryptography and System Security
(CrySyS Lab)
E-mail: mhorvath@crysys.hu

Abstract—**The focus of this work is to find ways to make attribute-based encryption (ABE) more suitable for access control to data stored in the cloud. However, ABE provides a flexible solution for "one to many" encryption, its adoption to the cloud environment require further refinement. One such issue is the assumption that secret key requests can be verified by one central key generator authority, which is often not realistic. The other is the problem of user revocation, for which a solution is essential in every real-word system, where unexpected events may occur. In contrast to the technically difficult multi-authority schemes (with user revocation feature), we investigate the feasibility of a much simpler approach: enabling secret key delegation in single authority schemes (with user revocation feature). We show key-delegation algorithm for one of the most important single authority ABE construction, and also for its extension that allow user revocation, thus achieving the desired features.**

I. Introduction

Cloud computing is an emerging paradigm of information technology, by which computer resources are provided dynamically via Internet. Besides cost savings, flexibility is the main driving force of outsourcing for instance data storage, although on the other hand it raises the issue of security, which leads us to the necessity of encryption. In order to fulfil the new requirements of the cloud environment, that traditional cryptographic protocols handle inflexibly, new schemes have appeared.

Attribute-Based Encryption (ABE) was proposed by Sahai and Waters [SW05] as the generalization of Identity-Based Encryption. Contrary to traditional public-key cryptography, ABE is intended for one-to-many encryption in which ciphertexts are not necessarily encrypted to one particular user, but for those who fulfil certain requirements. These requirements are related to attributes and access policies, namely decryption is possible if and only if the attributes satisfy the access policy. Goyal et al. [GPSW06] distinguished the two main variants of ABE, called key- and ciphertext-policy (KP/CP) ABE. The latter associates ciphertexts with access policies and attributes describe the user, accordingly these are embedded in the users' secret keys. A ciphertext can be decrypted by someone if and only if, his attributes - in his secret keys - satisfy the access structure given in the ciphertext[1]. In this concept the encryptor defines the access policy (usually expressed by a Boolean formula with AND, OR gates) and the key-issuer act as a certifier who gives out an attribute secret key to a user, only after it was made sure that he is eligible for that attribute. In a cloud computing scenario it is a natural expectation of a user to have the right of determining the access policy of the shared data, so in this work we are going to restrict our attention to CP-ABE. To meet the requirements of larger scale organizations that are intended to outsource data storage, CP-ABE still needs further refinement.

[1] Contrarily, in KP-ABE attributes are embedded in the ciphertext and the access policy in the secret key, with other words "intelligence" is assumed to be with the key issuer, and not the encryptor.

The assumption that a central key-generator authority is able to check the validity of all kinds of attributes in an institution is maybe unreasonable. For instance, in a university Alice may want to share some data with demonstrators, who are working on homework assessment. This target group can consist of undergraduates and other employees of different departments. To check whether an undergraduate student is truly a demonstrator of some department, before giving her the attribute "DEMONSTRATOR" is probably out of a university level authority's scope. Much more realistic to have the departments own authority for this task, however this requires a multi-authority setting, which is more complicated than standard single-authority schemes.

The other relevant issue is user revocation, as a tool for changing user's rights is essential in real life, when unexpected events may occur. Such occasion can be dismissal or the revealing of malicious activity, and we emphasise that user revocation is applied in *exceptional cases* like the above-mentioned, as all other cases can be handled with the proper use of attributes. E.g. a demonstrator's task usually lasts for a semester, and in the subsequent semester maybe different people will do the same tasks, although these *expected changes* can be handled simpler than revocation, by issuing more sophisticated attributes e.g.: "DEMONSTRATOR-2015/SPRING".

We note that the difficulties of the above problems have the same roots in ABE: different users may hold the same functional secret keys (corresponding to their attributes) which are bounded together (personalized) with randomization, using the same random values in each element. On the one hand, this method assumes that the randomization happens all at once, in a centralized manner, and on the other the revocation of a user's attributes affects all those secret keys, in which the revoked attributes were present.

II. Related Works and Our Results

Bethencourt et al. [BSW07] worked out the first CP-ABE scheme with a security proof in the generic bilinear and random oracle models. Waters [Wat11] improved this result both in terms of efficiency and security, giving a security reduction to a Diffie-Hellman-type hardness assumption. The problem of multiple authorities was first considered by Chase [Cha07] who had to rely on a central authority. Decentralized CP-ABE of Lewko and Waters [LW11] get rid of this restriction and avoided placing absolute trust in a single designated entity, however it is achieved in a practically inefficient way. Attribute revocation was first considered in [BSW07], while Wang et al. [WLWG11] and Yang et al. [YJRZ13] applied this in the multi-authority setting, considering the cloud storage scenario. A different approach is identity-based revocation, in which a special, unique "identity attribute" is given to each user that can be negated in the access policy (for all attributes it cannot be done efficiently). This approach was applied to enable revocation in the scheme of [Wat11] by [LZW+13] and in the decentralized setting by [Hor15].

In this work we investigate the viability of an alternative, much simpler approach instead of allowing multiple authorities in the system. We take advantage of the standard secret key generation algorithm in single authority CP-ABE constructions to enable the users to act as a restricted authority. More precisely, we supplement the CP-ABE with a key-delegation algorithm, that can be run by the user on any subset of his own secret keys resulting in new secret keys for the corresponding chosen attributes. Returning to our previous example, a professor with the attributes {"CRYPTOPROFESSOR", "CRYPTOHOMEWORKASSESSMENT"} could give "CRYPTOHOMEWORKASSESSMENT" attribute to a student, who helps him in this task. This approach significantly decreases the verification and key-generation burden of the authority and distribute these among already verified users.

Such key delegation algorithm for ABE was first introduced in [BSW07]. [IPN+09] considered the question of delegation together with attribute revocation in the presence of a "mediator", an additional, semi-trusted[2] entity. Without such an extra assumption we extend two CP-ABE constructions with delegation algorithms. First we consider the efficient and secure construction of [Wat11], than we investigate the possibility of delegation in [LZW+13] where ID-based user revocation is also possible. We conclude that in spite of some natural restrictions, this simple approach is a practical alternative of the multi-authority solution and can fulfil the requirements of secure cloud storage.

III. BACKGROUND

For our goal - to generate new secret keys for CP-ABE, using already existing keys and publicly available parameters - it is inevitable to know the structure of single authority ABE schemes. A Ciphertext-Policy Attribute-Based Encryption system is comprised of the following four algorithms:

Setup$(\lambda, U) \to (PK, MK)$ The algorithm takes security parameter λ and attribute universe description as input. It outputs the public parameters PK and a master secret key MK.

KeyGen$(MK, \gamma) \to SK_\gamma$ The key generation algorithm takes as input the master key MK and a set of attributes γ that describe the key. It outputs a private key SK_γ for the corresponding attributes.

Encrypt$(PK, \mathcal{M}, \mathbb{A}) \to CT$ The encryption algorithm takes as input the public parameters PK, a message $\mathcal{M}$, and an access structure $\mathbb{A}$ over the universe of attributes. CT ciphertext is produced such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt it and obtain $\mathcal{M}$. We assume that the ciphertext implicitly contains $\mathbb{A}$.

Decrypt$(PK, CT, SK_\gamma) \to \mathcal{M}$ The decryption algorithm takes in the public parameters PK, a ciphertext CT, which contains an access policy $\mathcal{A}$, and a private key SK_γ for a set γ of attributes. If these satisfy the access structure $\mathbb{A}$ then the algorithm will decrypt the ciphertext and return a message $\mathcal{M}$.

As our supplementing algorithm does not modify neither the functionality nor the operation of the above algorithms, we do not need the underlying tools of CP-ABE such as bilinear maps, access trees or linear secret sharing, so we omit the introduction of these here. For details on how the **Setup**, **Encrypt** and **Decrypt** algorithms work, we refer to the description of the used ABE constructions [Wat11], [LZW+13].

IV. KEY DELEGATION IN SINGLE-AUTHORITY SCHEMES

In case of CP-ABE where keys are issued by a central authority, the right of attribute delegation is a realistic expectation. This feature can subdue the tasks of the authority, in terms of both computation and verification of users' attributes. Of course our goal is *not sharing* of secret keys among users, but creating new secret keys without the master secret key, which have the same form and functionality as the ones that were created by the key issuing authority as it is depicted on Figure 1.

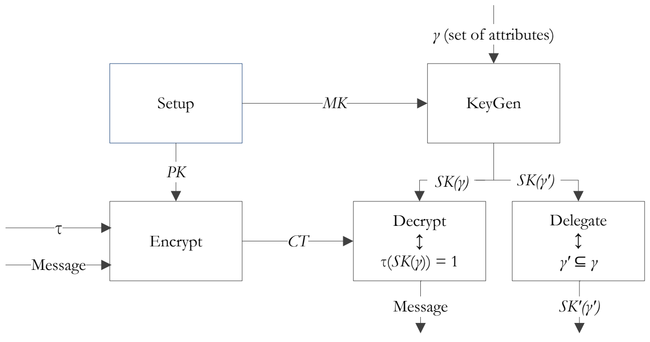

Figure 1. The structure of CP-ABE schemes, supplemented with key delegation (where τ denotes the access policy).

A. *Key Delegation in Waters' Scheme*

In the scheme of [Wat11], the publicly available parameters (generated by the **Setup** algorithm) include a generator g of the prime order group $\mathbb{G}$, g^a with random (secret) exponent $a \in \mathbb{Z}_p$ and U random group elements $h_1, \ldots, h_U \in \mathbb{G}$ associated with the attributes of the system[3]. The master secret key (MK) is

[2]Meaning that although it has no information about the plaintexts, it owns parts of the secret keys and has to be honest and available during the life-time of the system.

[3] An additional value, $e(g,g)^\alpha \in \mathbb{G}_T$ is also part of the public parameters, but it is used during encryption, so we will not need it.

g^α with random exponent $\alpha \in \mathbb{Z}$. The attribute secret key generation algorithm of the scheme (run by the authority) is the following:

KeyGen$(PK, MK, \gamma) \longrightarrow SK_\gamma$
The key generation algorithm will take as input the master secret key MK and a set of attributes γ and output a key that associated with that set. It first chooses a random $t \in \mathbb{Z}_p$, then it computes the key as

$$SK_\gamma = \{K = g^\alpha g^{at}, L = g^t, \forall x \in \gamma : K_x = h_x^t\}.$$

Note that the only information, used during key generation and not available for a user is the value of g^α, which only appears in K. Although, the delegation of secret keys (without sharing them) can be realized without this, as it is implicitly given to the user in his own private key. Delegation can be done in the manner of [BSW07]: after removing attributes $x \notin \gamma' \subseteq \gamma$ from the original key, by deleting the component K_x and re-randomize the remaining values. In order to re-randomize the components we would like to change the random exponent t to some $(t + t')$, where $t' \in \mathbb{Z}_p$ is chosen randomly by the user who delegates the key, so $(t+t')$ is a random number. To get this value in the exponent we have to multiply the original component with some power with index t'. Re-randomization is possible if in all components the base of powers with index t is public. In this case we can raise them to the power t' and multiply the result with the original component, gaining the re-randomized value. As the g^a term was already available in the reduction of [Wat11], the security proof is essentially unaffected. Based on these observations the algorithm is the following:

Delegate$(PK, \gamma', SK_\gamma) \rightarrow SK'_{\gamma'}$
The algorithm takes in a secret key SK, which is for a set γ of attributes, and another set $\gamma' \subseteq \gamma$. The secret key is of the form $SK = (K, L, \forall x \in \gamma : K_x)$. It chooses random $t' \in \mathbb{Z}_p$ then creates a new secret key SK' as

$$\begin{aligned} K' &= K \cdot g^{at'} = g^\alpha g^{a(t+t')} \\ L' &= L \cdot g^{t'} = g^{t+t'} \\ K'_x &= K_x \cdot h_x^{t'} = h_x^{t+t'} \quad \forall x \in \gamma' \subseteq \gamma. \end{aligned}$$

This simple algorithm reveals that the [Wat11] construction inherently contain the opportunity of key delegation, which can be utilised to decrease the authority's burden of verifying user's key requests, as these can be handled by other users. This possibility can resolve the need for multiple authorities, especially in case of a *hierarchical* setting, where the attributes of a user form the subset of the attributes of someone, who is in a higher level in the hierarchy. This is a natural expectation e.g. in a corporate environment where the central authority might issue the keys for users belonging to the top ℓ levels, who can issue keys for employees under themselves in the hierarchy.

B. Key Delegation and ID-based User Revocation

In this part, we consider the possibility of key delegation, when ID-based revocation is supported by the system. We are going to use the construction of [LZW$^+$13], which builds on [Wat11] and provides its extension. As negation of attributes in the access policy is an expensive operation (see [OSW07]) it is allowed only for unique "identity attributes". The ciphertext additionally embeds a list of revoked users and the decryptor is forced to compare his own ID with the list elements and decryption is possible if and only if no matching was found. To achieve this, the used keys and parameters are somewhat modified, i.e. the input of the **KeyGen** and **Decrypt** algorithms include the ID of the particular user and **Encrypt** takes in the IDs of already revoked users. The public parameters in [LZW$^+$13] are extended with three elements, compared with [Wat11]: an other generator h of group $\mathbb{G}$, h^a and g^{a^2} ($a \in \mathbb{Z}_p$ is still random secret), while MSK remains unchanged. In this case, the authority runs the following algorithm in order to generate secret key for a user with ID and attribute set γ:

KeyGen$(PK, MSK, \gamma, ID) \rightarrow SK_{ID,\gamma}$ The algorithm takes a unique identifier ID, an attribute set γ, the master key MSK and the public parameters PK as input. First, it checks the unique identifier ID to see whether it has been queried before. If yes, γ must be the same as in the previous query and the algorithm outputs the same secret key; if not, then the algorithm chose a random $t \in \mathbb{Z}_p$ and creates the private key $SK_{ID,\gamma}$ as:

$$K = g^\alpha g^{at} g^{a^2 t},\ L = g^{-t},\ D_{ID} = (g^{aID}h)^t,$$
$$\forall x \in \gamma : K_x = h_x^t.$$

The opportunity to revoke users raises several questions about key delegation. Should a system trust a user who delegates keys as much as it trusts the key issuing authority? What happens with those keys, which have been delegated by a user who was revoked later? Is it secure to allow users to create new IDs as well?

To answer them, we must consider the possible consequences. Suppose that a user is allowed to create new IDs and delegate attributes to them (from among his own). If this chain of created IDs is not recorded, then someone can simply duplicate his attribute secret keys, attaching them to a new ID and after the original ID was revoked, just use this new one. To record and follow the ID and key delegation chain is overcomplicated and also against the spirit of delegation, that supposed to simplify the tasks of the authority and the system. However, without this, it is obviously not secure to let users to totally substitute the trusted authority when revocation can occur.

In our proposal we make the key delegator responsible for the propagated keys in a simple way. As we have full trust only in the authority, which can verify whether an ID-attribute pair is valid or not, users are allowed to delegate their keys, only together with their ID. Practically it means, that from now on an ID does not represent a single user, but a user set, which contains one user with attributes from the authority and all of those parties who received their keys from that previous user or from someone who was already a member of that ID set. In this way, when an ID is revoked, it affects not only its original owner, but also everyone else who had delegated keys from any member of the user set with the revoked ID. The intuition behind this approach is that revocation is most often the result of some malicious activity, so we need to extend our countermeasures to all fields of the user's activity. If this policy has unwanted victims from the revoked set, these users can ask for new keys from the trusted authority, after they proved their right to own the given attributes, or from the party from where they received their previous key (if his ID was renewed). However, this method seems squandering, it is exactly the analogy of the more traditional attribute-based user revocation (applied by e.g. [WLWG11], [YJRZ13]), where the revoked attributes affects more users (those who owned any of the revoked attributes) than intended and these user's keys must be updated. This observation roughly implicates, that until the cardinalities of the biggest sets of users with the same IDs are under the occurrences of the most often used attributes, our solution does not require more key updates after revocation, than attribute-based revocation either in the single- or in the multi-authority case.

In this spirit, we give the following key delegation algorithm that supplements [LZW$^+$13]:

Delegate$(PK, ID, \gamma', SK_{ID,\gamma}) \to SK'_{ID,\gamma'}$
The algorithm takes in a secret key $SK_{ID,\gamma}$, which is for a set γ of attributes, and another set $\gamma' \subseteq \gamma$. The secret key is of the form $\{K, L, D_{ID}, \forall x \in \gamma : K_x\}$. It chooses random $t' \in \mathbb{Z}_p$ then creates a new $SK'_{ID,\gamma'}$ secret key by computing the following components:

$$\begin{aligned} K' &= K \cdot (g^a)^{t'} \cdot \left(g^{a^2}\right)^{t'} = g^\alpha g^{a(t+t')} g^{a^2(t+t')} \\ L' &= L \cdot g^{-t'} = g^{-(t+t')} \\ D'_{ID} &= D_{ID} \cdot \left((g^a)^{ID} h\right)^{t'} = \left(g^{aID} h\right)^{t+t'} \\ K'_x &= K_x \cdot h_x^{t'} = h_x^{t+t'} \quad \forall x \in \gamma' \subset \gamma. \end{aligned}$$

Note that we used the same re-randomization technique as before. In the algorithm the new keys are computed using only those values which were available in the original scheme as well, so the security proof of [LZW$^+$13] remains unaffected.

V. Evaluation

Our latter result provides an alternative solution for CP-ABE, in which secret keys can be issued by multiple parties and user revocation is also possible. Essentially it is a very similar functionality as achieved in [IPN$^+$09], [WLWG11], [Hor15] with different methods. From the perspective of security, [LZW$^+$13] with our delegation inherits the original security argument, that is based on a Diffie-Hellman-type assumption (just like [WLWG11]), while [IPN$^+$09] and [Hor15] are proven to be secure only in the generic bilinear group and random oracle models. In the other hand, a distributed scheme like [Hor15] can be more stable, as its security does not depend on the integrity of a single designated entity. Although from this point of view, our approach is equivalent to [WLWG11], where the authorities are not independent; and more beneficial than [IPN$^+$09], where the proper operation requires two entities to remain intact, one of which (the mediator) also needs to be available all the time.

In terms of usability, our proposal is quite natural in case of the above mentioned hierarchical organizations (which were also considered in [WLWG11]), as the verification of attribute requests can be done by the most competent parties i.e. by someone from an upper layer of the hierarchy. At the same time, we note that our stronger security guarantees result in a less flexible revocation than seen in [Hor15] or [IPN$^+$09]. Because of the identification of user sets instead of separate users, our revocation handling is more comparable with the attribute-based revocation of [WLWG11], where key updates might be necessary, just like in our case when not all members of a user set are intended to be revoked.

VI. Conclusion

In this work, we considered an alternative approach to deal with two obstacles of using ABE for the purpose of access control for data stored in the cloud. Evading the technically more challenging case of multiple authorities, we investigated the viability of key delegation by the users and found that this opportunity also can lead to a practical solution even when user revocation is possible, especially in case of a hierarchical structure of users.

Acknowledgment

This work was started as a master thesis at Eötvös Loránd University, in the Security&Privacy program of EIT ICT Labs Masterschool. I am thankful to my supervisor, Levente Buttyán from CrySyS Lab for his help and valuable comments and to the unknown reviewers for the useful remarks.

References

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[Cha07] Melissa Chase. Multi-authority Attribute Based Encryption. In *Theory of Cryptography*, volume 4392 of *LNCS*, pages 515–534. Springer Berlin Heidelberg, 2007.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. 2006.

[Hor15] Máté Horváth. Attribute-Based Encryption Optimized for Cloud Computing. In G.F. Italiano et al., editor, *SOFSEM 2015: Theory and Practice of Computer Science*, number 8939 in LNCS, pages 566–577. Springer, 2015.

[IPN$^+$09] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes (extended version), April 2009.

[LW11] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 568–588. Springer, 2011.

[LZW$^+$13] Yang Li, Jianming Zhu, Xiuli Wang, Yanmei Chai, and Shuai Shao. Optimized Ciphertext-Policy Attribute-Based Encryption with Efficient Revocation. *International Journal of Security & Its Applications*, 7(6), 2013.

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.

[WLWG11] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30(5):320–331, 2011.

[YJRZ13] Kan Yang, Xiaohua Jia, Kui Ren, and Bo Zhang. DAC-MACS: Effective data access control for multi-authority cloud storage systems. In *INFOCOM, 2013 Proceedings IEEE*, pages 2895–2903, 2013.