Sharing in the Rain: Secure and Efficient Data Sharing for the Cloud

Michalas, A.

This is a copy of the author’s accepted version of a paper subsequently published in the proceedings of *the 11th International Conference for Internet Technology and Secured Transactions (ICITST-2016)*, Barcelona, 05 to 07 Dec 2016, IEEE.

It is available online at:

https://doi.org/10.1109/ICITST.2016.7856693

Sharing in the Rain: Secure and Efficient Data Sharing for the Cloud

Antonis Michalas
Cyber Security Group,
Department of Computer Science
University of Westminster, UK
a.michalas@westminster.ac.uk

***Abstract*—Cloud storage has rapidly become a cornerstone of many businesses and has moved from an early adopters stage to an early majority, where we typically see explosive deployments. As companies rush to join the cloud revolution, it has become vital to create the necessary tools that will effectively protect users' data from unauthorized access. Nevertheless, sharing data between multiple users' under the same domain in a secure and efficient way is not trivial. In this paper, we propose Sharing in the Rain – a protocol that allows cloud users' to securely share their data based on predefined policies. The proposed protocol is based on Attribute-Based Encryption (ABE) and allows users' to encrypt data based on certain policies and attributes. Moreover, we use a Key-Policy Attribute-Based technique through which access revocation is optimized. More precisely, we show how to securely and efficiently remove access to a file, for a certain user that is misbehaving or is no longer part of a user group, without having to decrypt and re-encrypt the original data with a new key or a new policy.**

***Index Terms*—Security; Cloud Computing; Storage Protection; Access Control; Policies; Attribute-Based Encryption;**

I. INTRODUCTION

During the last few years cloud computing has met a great development and it is consider as one of the latest major evolution in computing. This is evident by the fact that key industrial players, such as Google, IBM and Microsoft, have invested a lot into the cloud space with main aim to create innovative applications with tangible positive impact on the day-to-day users' experience. Nowadays, cloud-based services exist along a spectrum from open public to closed private. Furthermore, it has been observed that they have a great resonance and impact to the productivity of single users' and it tends to become an integral part of our everyday life.

In the fast paced business world of today, cloud computing undoubtedly offers a wide range of advantages to both large companies and single users'. Cloud-based services are ideal for businesses with growing or fluctuating bandwidth demands since it is easy to scale up or down cloud resources based the actual demand at each time. As a result, cloud offers a level of flexibility and agility that can give businesses a real advantage over competitors. In addition to that, by using a properly configured cloud environment that will meet the needs of an organization can have a grade, positive, influence in cutting out the high cost of both hardware and software. Moreover, one of the greatest advantages and most appealing feature of cloud computing is the increased collaboration that offers between users' of the same or even different organizations. More precisely, cloud users' are able to work in teams and access, edit or share documents in real time from almost anywhere. This can have impressive results since it dramatically increases the productivity and further promote the real time collaboration even between users' in different continents.

As companies rush to join the cloud revolution it has become vital to have an understanding of how it works. Being able to solve issues if a problem does arise is a major asset for any potential employee. Besides the benefits of cloud computing there are also certain risks that companies that move to the cloud need to take into consideration. Because of the nature of cloud computing where users' store their data in remote locations, security is of paramount importance and is considered as the most critical risk that needs to be tackled. Ensuring that stored data are protected at all times in order to avoid outages and protect data from breaches and threats is a non trivial problem and lot of research [1, 2, 3, 4] has been done towards this direction. Building the right security protocols that will effectively protect private information from unauthorized access is key to a successful cloud implementation. Even though, there are many security gaps that needs to be properly addressed, cloud computing has all the necessary features in order to offer strong security and privacy-preserving mechanisms to both end users' and large organizations.

While lot of research has been done in securing cloud environments, storing sensitive data securely and launching hosts that are considered as trusted the problem of secure and efficiently share of data between multiple users' has been poorly addressed. More precisely, most of the existing solutions simply rely on encrypting data with a secret key and store them in the cloud in an encrypted form. However, this narrows down the vast capabilities of cloud computing since it makes data sharing rather inefficient. More precisely, the actual problem is identified when the owner of a file wants to revoke access to another, possibly malicious user'. Most of the existing approaches requires data owner to first share the secret key that has been used to encrypt the corresponding file with all users' that wishes to share the information. As a result, revoking access in the future requires from the data owner to decrypt the corresponding file and re-encrypt it with a fresh key that is not know and will not be shared with the malicious user'. This process is considered as inefficient and time consuming.

Having identified this, and taking into account that facilitating the sharing of information, projects, processes and resources with external parties, customers and employees, means that productivity and innovation get welcome boosts on a regular basis we propose a protocol that allows users' to share encrypted data in an efficient way based on an Attribute-Based Encryption technique [5, 6]. By doing this, we manage to fulfil one of the basic objectives for adopting cloud – that of increased collaboration between external partners.

A. Contribution

In this paper, we describe several contributions aimed at enhancing cloud infrastructure with additional security mechanisms. We hope these mechanisms will help address some of the issues that prevent security-sensitive users from benefiting from the functionality and economies of scale offered by cloud computing. More precisely, we describe a protocol for secure and efficient sharing of files between multiple users' of a cloud platform. Moreover, the proposed protocol shows how to construct a framework for secure file sharing by using the benefits of Revocable Attribute-Based Encryption. Furthermore, we define explicitly the entities that should be involved in such a scheme and we give valuable insights to not only to cloud providers who wish to incorporate our protocol in order to provide enhanced security to their clients but also to protocol designers who wish to build even more efficient cloud sharing schemes.

B. Organization

The remainder of this paper is organized as follows: In Section II we present the most important works that have been published and addressing the problem of secure data sharing in the cloud. In Section III we present the main entities that will participate in our system model and we proceed by defining the problem statement while in Section IV we introduce the basic notation that will be used throughout the paper as well as the threat model that we will consider. In Section V we describe a formal construction of the protocol while in Section VI we conclude the paper and we also present some future steps regarding the implementation and evaluation of the proposed architecture.

II. Related Work

In this section we present the related works that mainly focus on the problem of secure data sharing in a cloud environment.

In [7, 8] authors presented a framework for data and operation security in Infrastructure-as-a-Service (IaaS) clouds, consisting of protocols for a trusted launch of virtual machines and domain-based storage protection. Its security guarantees are supported by an extensive theoretical analysis with proofs about protocol resistance against attacks in the defined threat model. The protocols allow trust to be established by remotely attesting host platform configuration prior to launching guest virtual machines and ensure confidentiality of data in remote storage, with encryption keys maintained outside of the IaaS domain. In addition to that, authors provide functionality for sharing data between different domains. To this end, they present an XML-based language framework which enables clients of IaaS clouds to securely share data and clearly define access rights granted to peers. The paper also presents experimental results that demonstrate the validity and efficiency of the proposed protocols. The experimental results are based on an implementation of DBSP as an extension of OpenStack, a popular open-source cloud-computing platform. Even though the sharing functionality proposed in DBSP is based on standard cryptographic primitives, which makes it rather efficient, it is also considered as basic. In addition to that, the main drawback is the fact that DBSP is using a symmetric key to encrypt an entire disk. As a result, to give access to a user', data owner must reveal the secret key. As a result, revoking access to misbehaving users' opens up a new set of challenges since it will require to decrypt the entire disk and then re-encrypt the drive with a fresh key. This also implies that data owner will have to share again the fresh key with the legitimate users'.

Santos et al. [9] proposed Excalibur, a system using trusted computing mechanisms to allow decrypting client data exclusively on nodes that satisfy a tenant-specified policy. Excalibur introduces a new trusted computing abstraction, *policy-sealed data* to address the fact that TPM abstractions are designed to protect data and secrets on a standalone machine, at the same time over-exposing the cloud infrastructure by revealing the identity and software fingerprint of individual cloud hosts. The core of Excalibur is '*the monitor*', which is a part of the cloud provider, which organises computations across a series of hosts and provides guarantees to tenants. Tenants first decide a policy and receive evidence regarding the status of the monitor along with a public encryption key, and then encrypt their data and policy using ciphertext-policy attribute-based encryption [6]. To decrypt, the stored data hosts receive the decryption key from the monitor who ensures that the corresponding host has a valid status and satisfies the policy specified by the client at encryption time. The main drawback of this work is that the protocol does not offer revocation functionality. Sharing in the Rain is overcoming this limitation by using a revocable attribute-based encryption scheme.

In [10] authors presented a forward-looking design for secure storage and file sharing in cloud environments. The scheme was based on a Symmetric Searchable Encryption (SSE) scheme [11] that allows patients of an electronic healthcare system to securely store encrypted versions of their medical data and search directly on them without having to decrypt them first. Even though the scheme offers some kind of secure sharing it is not that flexible and efficient since it does not rely on policies. Furthermore, even though authors provide a discussion regarding access revocation they do not provide a concrete and efficient solution. Hence, the protocol is considered as inefficient for sharing large amount of data between multiple users'.

Authors in [12] proposed an efficient access control scheme that allows users' to dynamic update a policy. More precisely, the policy update is outsourced to the cloud server while at the same time the server does not learn any private information regarding the processed data. In addition to that, the scheme is based on ABE and assumes a semi-trusted and not fully trusted cloud service provider. Furthermore, the proposed scheme supports different types of access policies (e.g., Boolean Formulas, LSSS Structure and Access Tree).

III. System Model and Problem Statement

In this section, we introduce the system model that we consider by explicitly describing the main entities that participate in our protocol. Furthermore, we strictly define the problem that we try to tackle.

Cloud Service Provider (CSP): One of the common models of a cloud computing platform is Infrastructure-as-a-Service (IaaS). In its simplest form, such a platform consists of cloud hosts which operate virtual machine guests and communicate through a network. Often a cloud middleware manages the cloud hosts, virtual machine guests, network communication, storage resources, a public key infrastructure and other resources. Cloud middleware creates the *cloud infrastructure* abstraction by weaving the available resources into a single platform. In our system model we consider a cloud computing environment based on a trusted IaaS provider like the one described in [8]. The IaaS platform consists of cloud hosts which operate virtual machine guests and communicate through a network. In addition to that, we assume a Platform-as-a-Service (PaaS) provider, like the one described in [1], that is built on top of the IaaS platform and can host multiple outsourced databases. Furthermore, the cloud service provider is responsible for storing the data of users' and also providing data access.

Registration Authority (RA): RA is responsible for the registration of users. Additionally, RA has a public/private key pair denoted as $\mathsf{pk_{RA}}/\mathsf{sk_{RA}}$. Apart from that, RA is responsible for generating parameters that will be used for the proper function of the application (e.g. reveal the identity of a misbehaving user). RA can run as a separate third party but can be also implemented as part of the cloud platform we described earlier.

User (u): In our scenario a user interacts with the CSP in order to mange certain files that has access to. The operations that a user can perform are the following: *a)* register to the service, *b)* generate encryption keys to safely protect her data, *c)* store data in the cloud, *d)* share data with other users' by creating certain policies using a Key Policy Attribute-Based Encryption scheme. Furthermore, each user' has a unique identifier u_i. A user u_i might be also referred as data owner when she is the one who generated a certain file. Each user' u_i has a private/public key pair ($\mathsf{pk_i}/\mathsf{sk_i}$). The private key is kept secret, while the public key is shared with the rest of the community. These keys will be used to secure message exchanges in the community, hence the communication lines between parties are assumed to be secure. It is also assumed that users knows the public keys of RA and the hosts operated by the CSP. In addition to that, each u_i that serves as data owner has a master key $\mathsf{MSK_i}$ that will be used to generate secret keys based on certain policies and attributes.

Problem Statement: *Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be the set of all users that are registered through a registration authority (RA) and CSP the cloud service provider that stores users' data. Lets assume that a user u_i stores a set of m different data/files to the CSP. We denote this set of files as $\mathcal{D}_i = \left\{d_1^i, \ldots d_m^i\right\}$. The problem here is to find a way to achieve the following:*

1) *Keep the content of each $d_j^i \in \mathcal{D}_i$ private against both internal and external attacks;*
2) *User u_i should be able to securely share a file d_j^i with another user based on a certain policy;*
3) *A data owner u_i should be able to efficiently revoke access to a user u_c for a file that has shared with her. This should not require the data owner to decrypt and re-encrypt the file with a fresh key that will no longer be available to u_c;*

IV. Notation and Threat Model

In this section, we introduce the notations that we use throughout the rest of the paper as well as the threat model that we consider.

A. Notation

In order to provide an efficient and realistic solution to the problem described in Section III, we need to build a protocol through which newly encrypted data will not be decryptable by a user's key if that user's access has been revoked. To this end, we will be using a key-policy attribute-based encryption (KP-ABE) scheme. In a KP-ABE scheme every secret key is generated with a policy P and every ciphertext is binded to a set of attributes U. Then, decryption is only possible if $P(U) = \mathsf{True}$. From now on we will refer to the set of all available attributes as $\Omega = \{a_1, \ldots, a_n\}$, while the set of all available policies will be denoted as $\mathcal{P} = \{P_1, \ldots, P_m\}$.

We now proceed with the definition of a revocable KP-ABE scheme as described in [13].

Definition 1 (Revocable Key-Policy ABE): A revocable KP-ABE scheme is a tuple of the following five algorithms:

1. Setup is a probabilistic algorithm that takes as input a security parameter λ and outputs a public key pk and a master key MSK. We denote this by $(\mathsf{pk}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. Gen is a probabilistic algorithm that takes as input a master key, a policy $P \in \mathcal{P}$ and the unique identifier of a user and outputs a secret key which is bind both to the corresponding policy and user'. We denote this by $(\mathsf{sk_{P,ID}}) \leftarrow \mathsf{Gen}(\mathsf{MSK}, P, ID)$.
3. Enc is a probabilistic algorithm that takes as input a public key, a message m, a set of attributes $\mathcal{S} \in \Omega$ and a timestamp t. After a proper run, the algorithm outputs a ciphertext $c_{\mathcal{S},t}$ which is bind both to the set of attributes and the time. We denote this by $(\mathsf{c}_{\mathcal{S},\mathsf{t}}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m, \mathcal{S}, t)$.
4. $\mathsf{KeyUpdate}$ is a probabilistic algorithm that takes as input a master key, a revocation list rl and a timestamp t and

TABLE I
NOTATION INDEX

Symbol	Description
CSP	Cloud Service Provider
RA	Registration Authority
u_i	A user with unique identifier i
m	An arbitrary message
Enc	Encryption algorithm
Dec	Decryption algorithm
K	Symmetric key
$\mathsf{pk_i}/\mathsf{sk_i}$	Public/private key pair of user i
$\mathsf{MSK_i}$	A master secret key of data owner i
a	Attribute
P	Policy
rl	Revocation List

outputs a key update information for time t. We denote this by $(\mathsf{K_t}) \leftarrow \mathsf{KeyUpdate}(\mathsf{MSK}, rl, t)$.

5. Dec is a deterministic algorithm that takes as input a secret key, a key update $\mathsf{K_{t'}}$ and a ciphertext and outputs the original message m *iff* the set of attributes $\mathcal{S}$ that are bind to the ciphertext satisfies the policy P, $t' \geq t$ and the ID of the corresponding user was not revoked at time t. We denote this by $\mathsf{Dec}(\mathsf{sk_{P,ID}}, \mathsf{K_{t'}}, c_{\mathcal{S},t}) \rightarrow m$.

A summary of the notation introduced so far is presented in Table I.

B. Threat Model

Our threat model is similar with the one described in [7], which is based on the Dolev-Yao adversarial model [14] and further assumes that privileged access rights can used by a remote adversary $\mathcal{ADV}$ to leak confidential information. $\mathcal{ADV}$, e.g. a corrupted system administrator, can obtain remote access to any host maintained by the IaaS provider, but cannot access the volatile memory of guest VMs residing on the compute hosts of the IaaS provider.

Hardware Integrity: We assume that the cloud provider has taken all the necessary technical and non-technical measures in order to protect the underling hardware from tampering.

Physical Security: We assume physical security of the data centres where the IaaS is deployed. This assumption holds both when the IaaS provider owns and manages the data center (as in the case of Amazon Web Services, Google Compute Engine, Microsoft Azure, etc.) and when the provider utilizes third party capacity, since physical security can be observed, enforced and verified through known best practices by audit organizations. This assumption is important to build higher-level hardware and software security guarantees for the components of the IaaS. We assume the record is kept on protected storage with read-only access and the adversary cannot tamper with it.

Network Infrastructure: The IaaS provider has physical and administrative control of the network. $\mathcal{ADV}$ is in full control of the network configuration, can overhear, create, replay and destroy all the exchanged messages between the CSP and their resources (virtual machines, database components etc) as well as with other entities that participate in our system model (i.e. the registration authority).

Cryptographic Security: We assume encryption schemes are semantically secure and the $\mathcal{ADV}$ cannot obtain the plain text of encrypted messages. In addition to that, we explicitly assume that the $\mathcal{ADV}$ cannot forge the revocation list and cannot decrypt a ciphertext without knowing the corresponding secret key. Furthermore, we assume that the probability of $\mathcal{ADV}$ guessing a generated random number is negligible. Finally, we explicitly exclude denial-of-service attacks [15, 16, 17, 18] from our adversarial model and we focus on $\mathcal{ADV}$ that aims to compromise the confidentiality of data by forging existing access policies generated by the corresponding data owners.

V. SHARING IN THE RAIN

In this section, we present Sharing in the Rain (RitS) that constitutes the core of this paper's contribution. Before we proceed with the formal construction of the protocol, we provide a high level description that gives the reader a good overview of the functionality that is offered by our protocol as well as a typical use-case scenario.

A user u_i registers to the cloud service by contacting the registration authority. By doing this, u_i receives a credential through which she can prove that is a legitimate/registered user later on when she interacts with the CSP, other users' or any other entity that might be part of the system model (e.g. a third party that is collaborating with the CSP). Now that u_i has registered, she can start uploading files to the CSP. However, to do that in a secure way the files needs to be transmitted and stored in an encrypted form in order to avoid both internal and external attacks. To this end, u_i encrypts each file that wishes to store in the cloud by using a set of attributes (apart from the set of attributes also a public key and other parameters are used as we will see in the next paragraph). The generated ciphertext is sent to the CSP who cannot decrypt it since it does not have knowledge of a valid private key – hence the content of the file remains private even if the CSP acts maliciously. Now that u_i has stored a file in the cloud storage, she wishes to share it with a different user u_j. To do this, u_i, who is the data owner, generates a unique private key for u_j and send it to her. This key is binded to a certain policy that is defined by the data owner and allows u_j to decrypt the corresponding file only if the attribute set that the ciphertext is bind with satisfies the policy. Apart from successfully sharing files, one of our goals is to efficiently revoke access for a user. Having this in mind, we assume that u_i wishes to revoke access for the users u_j. To do so, u_i will only have to run an algorithm that will actually revoke access for the unique key that was generated for user u_j. Apart from that, u_i will not have to decrypt and then re-encrypt the file with a fresh key since the key that is hold by u_j will not be a valid decryption key.

A. Protocol Construction

Now we can proceed with the formal description of our protocol RitS in which we will show a formal construction of all the necessary algorithms with four main participating entities: *Cloud Service Provider*, *Users'* and *Registration Authority*. RitS mainly comprises a public-key encryption scheme and a key-police attribute-based encryption scheme.

RitS.Setup : Each entity obtains a public/private key pair and publishes its public key while it keeps the private key secret. Below we provide the list of key pairs used in the following protocol:

- $(\mathsf{pk_{CSP}}, \mathsf{sk_{CSP}})$ – public/private key pair for the cloud service provider;
- $(\mathsf{pk_{RA}}, \mathsf{sk_{RA}})$ – public/private key pair for the Registration Authority;
- $(\mathsf{pk_i}, \mathsf{sk_i})$ – public/private key pair for a user with ID i;

RitS.Registration : The registration phase takes part between a user u_i that wishes to register and the registration authority who is responsible for evaluating registration requests for users'. During this phase a user that wishes to register contacts RA by sending a registration request and the following protocol takes place between RA and the user:

*Definition 2 (*RitS.Registration*):* A registration scheme, denoted by RitS.Registration, is defined by two algorithms (RegRequest, RegOutput) such that:

1. RegRequest is a probabilistic algorithm that takes as input $\mathsf{pk_{RA}}$ and $\mathsf{pk_i}$ and outputs a registration request. $(\mathsf{Enc_{pk_{RA}}}(\mathsf{pk_i}), H(\mathsf{pk_i})) \leftarrow \mathsf{RegRequest}(\mathsf{pk_{RA}}, \mathsf{pk_i})$.
2. RegOutput is a deterministic algorithm that takes as input a RegRequest and outputs a credential if the request is valid and $\perp$ otherwise. If RegRequest is successfully verified then RegOutput outputs a credential for user u_i encrypted with her public key $(\mathsf{Enc_{pk_i}}(\mathsf{cred_i}))$. Otherwise, RegOutput returns $\perp$ which means that user sent an invalid RegRequest.

RitS.Store : After the successful registration, u_i is now able to store data to the cloud storage. During this phase the communication takes place between the user' and the CSP.

*Definition 3 (*RitS.Store*):* A file storage scheme, denoted by RitS.Store, is defined by a tuple of three algorithms (StoreRequest, StoreToken, StoreFile) such that:

1. StoreRequest takes as input $\mathsf{pk_{CSP}}$ and $cred_i$ and outputs a store request. This is executed by a registered user u_i who wishes to store a file in the cloud. We denoted this $(\mathsf{Enc_{pk_{CSP}}}(cred_i)) \leftarrow \mathsf{RegRequest}(\mathsf{pk_{CSP}}, cred_i)$. The output is sent by u_i to the CSP.
2. StoreToken is a probabilistic algorithm that takes as input a StoreRequest and outputs a token if the input is valid or $\perp$ otherwise. If StoreRequest is successfully verified then StoreToken outputs a token for u_i encrypted with her public key $(\mathsf{Enc_{pk_i}}(\tau))$. Otherwise, StoreToken returns $\perp$.
3. StoreFile is a probabilistic algorithm that takes as input a StoreToken, a public key, a file to be encrypted, a set of attributes and a timestamp and outputs an encrypted version of the file which is binded with the set of attributes. We denote this by: $c_{\mathcal{S},t}^{d_j^i} \leftarrow \mathsf{StoreFile}(\mathsf{StoreToken}, \mathsf{pk_i}, d_j^i, \mathcal{S}, t)$. The generated ciphertext is binded with the set of attributes from $\mathcal{S}$ and is sent by u_i to the CSP. More precisely, u_i sends the following message to the CSP: $\left\langle c_{\mathcal{S},t}^{d_j^i}, H(c_{\mathcal{S},t}^{d_j^i}), \mathsf{Enc_{CSP}}(\tau) \right\rangle$ If, StoreToken is equal to $\perp$, StoreFile also returns $\perp$.

RitS.Share : Now that u_i has stored an encrypted file in the cloud is ready to share its content with other registered users'. To do so, u_i who is the data owner of file d_j^i will have to create a unique key for each user u_j that wishes to share the file with.

*Definition 4 (*RitS.Share*):* A file storage scheme, denoted by RitS.Store, is defined by a two algorithms (ShareKeyGen, ShareDec) such that:

1. ShareKeyGen is a probabilistic algorithm that takes as input a master key $\mathsf{MSK_i}$, a policy P_j, a user ID u_j and the public key of u_j and outputs a secret key $\mathsf{sk_{P,j}}$. We denote this by: $(\mathsf{Enc_{pk_j}}(\mathsf{sk_{P,j}}), H(\mathsf{sk_{P_j}})) \leftarrow \mathsf{ShareKeyGen}(\mathsf{MSK_i}, P, u_j, \mathsf{pk_j})$.
2. ShareDec is a deterministic algorithm that takes as input a private key $\mathsf{sk_{P,j}}$, a key update $\mathsf{K_{t'}}$ and an encrypted file $c_{\mathcal{S},t}$ and outputs a message m. We denote this by: $\mathsf{ShareDec}(\mathsf{sk_{P,j}}, \mathsf{K_{t'}}, c_{\mathcal{S},t}) \to m$.

RitS.Revoke : The last phase of our protocol allows a data owner to revoke access to a user that has shared a file with.

*Definition 5 (*RitS.Revoke*):* A file storage scheme, denoted by RitS.Revoke, is defined by the following algorithm Revoke:

1. Revoke takes as input a master key $\mathsf{MSK_i}$, a revocation list rl and a timestamp t and outputs a key update information. We denote this by: $(\mathsf{K_t}) \leftarrow \mathsf{Revoke}(\mathsf{MSK_i}, rl, t)$.

RitS.Revoke is run by the data owner and by generating this key she adds the secret keys of user u_j to be revoked in the revocation list. As a result, the the secret key $\mathsf{sk_{P,j}}$ that was shared earlier with u_j will no longer be valid since the attribute set $\mathcal{S}$ that is binded with the encrypted file will no longer satisfy the policy P that is binded with $\mathsf{sk_{P,j}}$. Hence, u_j can no longer access the file d_j^i and u_i will not have to decrypt and re-encrypt the file with a fresh key.

VI. Conclusion

In this paper, we proposed a protocol for secure and efficient data sharing in a cloud environment. The proposed protocol was based on Key-Policy Attribute-Encryption and allowed cloud users' to encrypt files based on certain policies and attributes. Furthermore, the protocol allows to securely and efficiently remove access to a file, for a certain user that is misbehaving or is no longer part of a user group, without having to decrypt and re-encrypt the original data.

As future steps, we plan to implement our protocol in order to measure its performance and prove its effectiveness in a real cloud environment. Furthermore, we plan to explore the incorporation of our protocol with mobile sensing applications and more precisely with privacy preserving reputation systems for cloud-based participatory sensing applications. The envisioned system will be based on [19, 20, 21, 22] and will effectively maintain the privacy and anonymity of users' [23, 24, 25].

References

[1] Yiannis Verginadis et al. "PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services". In: *Proceedings of the 5th International Conference on Cloud Computing and Services Science*. 2015, pp. 206–213. ISBN: 978-989-758-104-5. DOI: 10.5220/0005489302060213.

[2] N. Paladi and A. Michalas. ""One of our hosts in another country": Challenges of data geolocation in cloud storage". In: *Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014 4th International Conference on*. 2014, pp. 1–6.

[3] Antonis Michalas, Nicolae Paladi, and Christian Gehrmann. "Security aspects of e-Health systems migration to the cloud". In: *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*. IEEE. 2014, pp. 212–218.

[4] A. Michalas and M. Bakopoulos. "SecGOD Google Docs: Now i feel safer!" In: *Internet Technology And Secured Transactions, 2012 International Conference for*. 2012, pp. 589–595.

[5] Amit Sahai and Brent Waters. "Fuzzy Identity-based Encryption". In: *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT'05. Aarhus, Denmark: Springer-Verlag, 2005, pp. 457–473. ISBN: 3-540-25910-4, 978-3-540-25910-7. DOI: 10.1007/11426639_27. URL: http://dx.doi.org/10.1007/11426639_27.

[6] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-Policy Attribute-Based Encryption". In: *Proceedings of the 2007 IEEE Symposium on Security and Privacy*. SP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 321–334. ISBN: 0-7695-2848-1. DOI: 10.1109/SP.2007.11. URL: http://dx.doi.org/10.1109/SP.2007.11.

[7] N. Paladi, C. Gehrmann, and A. Michalas. "Providing User Security Guarantees in Public Infrastructure Clouds". In: *IEEE Transactions on Cloud Computing* PP.99 (2016), pp. 1–1. ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2525991.

[8] Nicolae Paladi, Antonis Michalas, and Christian Gehrmann. "Domain Based Storage Protection with Secure Access Control for the Cloud". In: *Proceedings of the 2014 International Workshop on Security in Cloud Computing*. ASIACCS '14. Kyoto, Japan: ACM, 2014. ISBN: 978-1-4503-2805-0.

[9] Nuno Santos et al. "Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services". In: *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. Bellevue, WA: USENIX, 2012, pp. 175–188. ISBN: 978-931971-95-9. URL: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/santos.

[10] A. Michalas and R. Dowsley. "Towards Trusted eHealth Services in the Cloud". In: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. 2015, pp. 618–623. DOI: 10.1109/UCC.2015.108.

[11] Rafael Dowsley, Antonis Michalas, and Matthias Nagel. *A report on design and implementation of protected searchable data in IaaS*. Tech. rep. Swedish Institute of Computer Science (SICS), 2016.

[12] K. Yang et al. "Enabling efficient access control with dynamic policy updating for big data in the cloud". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 2013–2021. DOI: 10.1109/INFOCOM.2014.6848142.

[13] Amit Sahai and Hakan Seyalioglu. "Dynamic credentials and ciphertext delegation for attribute-based encryption". In: *in Proceedings of the 32nd Annual International Cryptology Conference: Advances in Cryptology - CRYPTO2012*. Springer, 2012, pp. 199–217.

[14] Danny Dolev and Andrew C Yao. "On the security of public key protocols". In: *Information Theory, IEEE Transactions on* 29.2 (1983).

[15] Antonis Michalas et al. "New client puzzle approach for dos resistance in ad hoc networks". In: *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*. IEEE. 2010, pp. 568–573.

[16] Antonis Michalas, Nikos Komninos, and Neeli R Prasad. "Mitigate DoS and DDoS attack in mobile ad hoc networks". In: *International Journal of Digital Crime and Forensics (IJDCF)* 3.1 (2011), pp. 14–36.

[17] A. Michalas, N. Komninos, and N.R. Prasad. "Multiplayer game for DDoS attacks resilience in ad hoc networks". In: *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*. 2011, pp. 1–5. DOI: 10.1109/WIRELESSVITAE.2011.5940931.

[18] Antonis Michalas, Nikos Komninos, and Neeli R Prasad. "Cryptographic Puzzles and Game Theory against DoS and DDoS attacks in Networks". In: *International Journal of Computer Research* 19.1 (2012), p. 79.

[19] T. Dimitriou and A. Michalas. "Multi-Party Trust Computation in Decentralized Environments". In: *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*. 2012, pp. 1–5.

[20] T. Dimitriou and A. Michalas. "Multi-party trust computation in decentralized environments in the presence of malicious adversaries". In: *Ad Hoc Networks* 15 (2014), pp. 53 –66.

[21] Antonis Michalas et al. "Vulnerabilities of Decentralized Additive Reputation Systems Regarding the Privacy of Individual Votes". English. In: *Wireless Personal Communications* 66.3 (2012), pp. 559–575. ISSN: 0929-6212.

[22] A. Michalas et al. "Privacy-preserving scheme for mobile ad hoc networks". In: *Computers and Communications (ISCC), 2011 IEEE Symposium on*. 2011, pp. 752–757. DOI: 10.1109/ISCC.2011.5983930.

[23] Antonis Michalas and Thanassis Giannetsos. "The Data of Things: Strategies, Patterns and Practice of Cloud-based Participatory Sensing". In: *Proceedings of the 1st International Conference on Innovations in InfoBusineess and Technology*. Colombo, Sri Lanka, 2016.

[24] Antonis Michalas and Nikos Komninos. "The lord of the sense: A privacy preserving reputation system for participatory sensing applications". In: *Computers and Communication (ISCC), 2014 IEEE Symposium*. IEEE. 2014, pp. 1–6.

[25] A. Michalas et al. "Secure and trusted communication in emergency situations". In: *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*. 2012, pp. 1–5. DOI: 10.1109/SARNOF.2012.6222751.