Protecting medical data stored in public Clouds

Nikos Fotiou and George Xylomenos
Mobile Multimedia Laboratory, Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Athens 10434, Greece
fotiou@aueb.gr, xgeorge@aueb.gr

***Abstract*—Public Clouds offer a convenient way for storing and sharing large amounts of medical data. Nevertheless, using a shared infrastructure raises significant security and privacy concerns. Even if the data are encrypted, the data owner should share some information with the Cloud provider, in order to enable the latter to perform access control; given the high sensitivity of medical data, even such limited information may jeopardize end-user privacy. In this paper we employ an access control delegation scheme to enable the users themselves to perform access control on their data, which are stored in a public Cloud. To selectively provide access to these data without sacrificing their confidentiality we rely on encryption: our system encrypts data before storing them in the Cloud and applies proxy re-encryption so as to encrypt data separately for each (authorized) user.**

***Index Terms*—Access control, Identity-based encryption, Proxy re-encryption.**

I. Introduction

Nowadays, smart devices that collect users' vital signals have become a commodity. It is expected that soon the data collected by these devices will be used for preventing and/or diagnosing various health related problems, as well as for promoting a healthier way of living and well-being. Storing and sharing these data using a public Cloud infrastructure appears to be an appealing option, as public Clouds offer cost effective and reliable storage services. On the other hand, security and privacy concerns are raised, as medical data are highly sensitive and they should be protected, even against the Cloud service provider. Encryption and access control can be used as a countermeasure, but privacy threats remain. For example, an access control policy of the form "these (encrypted) data can only be accessed by psychiatrist A" reveals to the entity that performs access control that the data owner shares some data with a psychiatrist.

In this paper we propose a system that allows secure and private storage of medical records in the Cloud. Our system allows data owners to define access control policies and to enforce them by themselves. The Cloud provider is only responsible for storing data and for respecting the access control decisions of the data owner. Even if the Cloud provider misbehaves, the data remain protected since they are encrypted in a way that only authorized users can access them; unauthorized users – including the Cloud provider – learn nothing about the data. In order to achieve our goal we extend the system proposed by [1] by adding an additional layer of data confidentiality protection. Our proposal encrypts data before storing them in the Cloud and re-encrypts them as necessary before sharing; data are encrypted only once by an entity owned by the data owner, and then the Cloud is responsible for re-encrypting the data in such a way that only authorized clients can access them.

The remainder of the paper is organized as follows. Section II briefly presents access control delegation and proxy re-encryption. Section III presents our system design. In Section IV we evaluate our solution and in Section V we present related work in the area. Finally we conclude our paper in Section VI

II. Background

A. Access control delegation

The access control scheme proposed in [1] separates *data storage* and *access control* functions: the former is implemented in a public Cloud, whereas the latter is implemented by a trusted entity named *access control provider* (ACP). These entities interact with each other as follows (Figure 1)[1]: Initially, a data owner creates an *access control policy*, stores it in an ACP (step 1) and obtains a *URI* for that policy (step 2). Then, he stores some data in the Cloud, indicating at the same time the URI of the policy that protects these data (step 3). When a client tries to access these data (step 4), the Cloud responds with the URI of the access control policy and a unique token (step 5). Then, the client authenticates herself to the ACP and requests authorization (step 6). If the client "satisfies" the access control policy, the ACP generates a signed *authorization* and sends it back (step 7). Finally, the client repeats her request to the Cloud, this time including the authorization (step 8). The Cloud checks the validity of the authorization and if it is valid, it returns the desired data (step 9).

This scheme has many advantages. The Cloud provider learns nothing about the client since all her personal data (which are required to evaluate the access control policy) are stored in the ACP. Moreover, Cloud providers do not have to interpret any access control policies, therefore they do not need to understand content owner specific semantics. Access control policies are reusable i.e, in order to protect a new item using an existing access control policy the same URI can be simply re-used. Access control policies can be easily updated; updating

[1] The description has been modified to fit the purposes of the present paper.

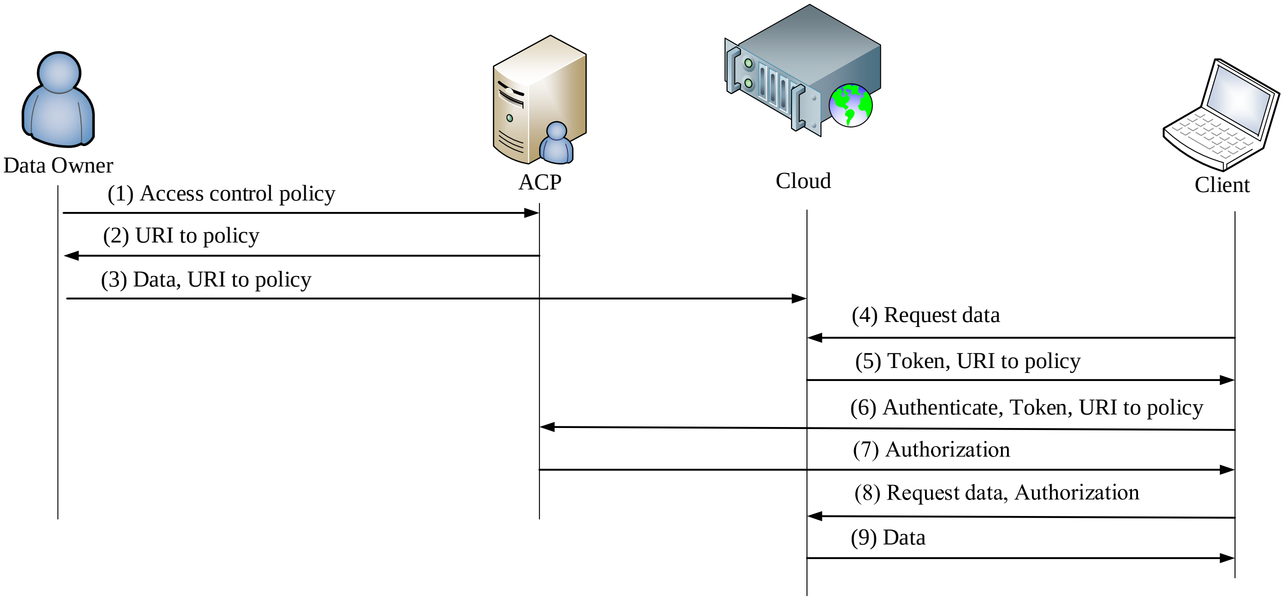

Fig. 1. Access control delegation

and access control policy does not involve any communication with the Cloud provider. Finally, providing that many Cloud providers support this scheme, it is trivial for a data owner to migrate from one Cloud provider to another.

B. *Proxy re-encryption*

A Proxy re-encryption (PRE) scheme is a scheme in which a third, semi-trusted party, the *proxy*, is allowed to alter a ciphertext encrypted with the public key of a user A (the delegator), in a way that another user B (the delegatee) can decrypt it with her own appropriate key (in most cases, her secret private key). During this process the proxy learns nothing about the private keys of A and B, and does not gain access to the encrypted data.

In this paper we employ the *identity-based proxy re-encryption* (IB-PRE) scheme by Green and Ateniese [2]. In particular we use a variant of that scheme in which the delegatee uses an RSA public key instead of identity-based encryption (section 5 of [2]). This scheme specifies the following algorithms (the description has been adapted to the RSA variant):

- `Setup`: it is executed by a *Private Key Generator* (PKG). It takes as input a security parameter k and returns a `master-secret key` (MSK) and some `system parameters` (SP). The MSK is kept secret by the PKG, whereas the SP are made publicly available.
- `Extract`: it is executed by a PKG. It takes as input the SP, the MSK, and an arbitrary string ID, and returns a `secret key` SK_{ID}.
- `Encrypt`: it can be executed by anyone. It takes as input an arbitrary string ID, a message M, and the SP, and returns a ciphertext C_{ID}.
- `RKGen`: it is executed by the owner of the identifier $ID1$. It takes as input the SP, the secret key SK_{ID1} and an RSA public key $RSA2$ and generates a (public) re-encryption key $RK_{ID1 \rightarrow RSA2}$.
- `Reencrypt`: it is executed by a *proxy*. It takes as input the SP, a re-encryption key $RK_{ID1 \rightarrow RSA2}$, and a ciphertext C_{ID1} and outputs a new ciphertext C_{RSA2}.
- `Decrypt`: is is executed by the owner of the key $RSA2$. It takes as input the SP, C_{RSA2}, the corresponding RSA private decryption key SK_{RSA2}, and returns the message M.

Figure 2 gives an example of a complete IBE-PRE transaction. In this figure, initially the PKG generates the MSK and the SP, and makes the SP publicly available (step 1). Then it extracts SK_{ID1} and distributes it to the corresponding user $ID1$ (step 2). Another user creates a ciphertext using as a public key the string $ID1$ and stores it in a proxy (step 3). This can only be decrypted by the user that owns $ID1$, (and therefore knows the corresponding SK_{ID1}). To allow a user $ID2$ to decrypt the content using an RSA private key RSA_2, the user that owns $ID1$ creates a re-encryption key $RK_{ID1 \rightarrow RSA2}$ and sends it to the proxy. The proxy re-encrypts C_{ID1} using the re-encryption key and generates C_{RSA2}. The user $ID2$ is now able to decrypt the re-encrypted ciphertext. The proxy learns nothing about the contents of the ciphertext or the secret keys of the users. Moreover, the scheme of Green and Ateniese assures the SK of the delegator (in this example, SK_{ID1}) is protected even if the proxy and the delegatee collude.

If the original version of the scheme is used (instead of the RSA variant) then all delegator-delegatee pairs have to agree on the the same PKG. This however, raises security concerns, since PKGs will know the private keys of both

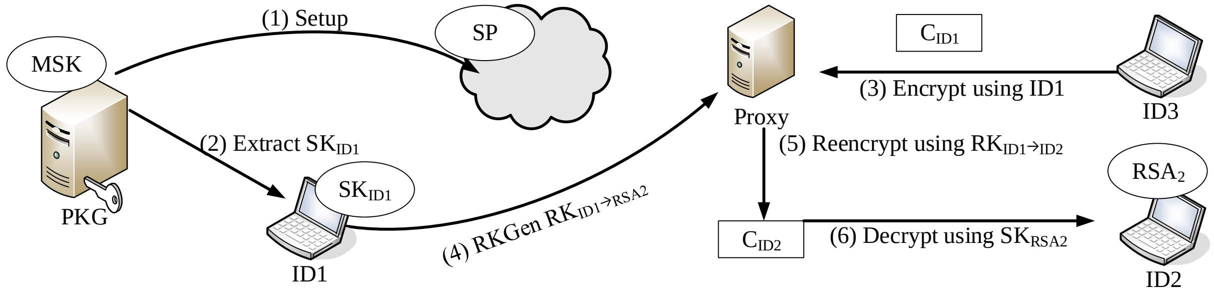
MSK
(1) Setup
SP
PKG
(2) Extract SK$_{ID1}$
SK$_{ID1}$
ID1
(4) RKGen RK$_{ID1\rightarrow RSA2}$
Proxy
C$_{ID1}$
(3) Encrypt using ID1
ID3
(5) Reencrypt using RK$_{ID1\rightarrow ID2}$
RSA$_2$
C$_{ID2}$
(6) Decrypt using SK$_{RSA2}$
ID2

Fig. 2. IBE-PRE example

parties of a transaction. Moreover, if a delegatee interacts with many delegators (as for example in the case of a hospital that interacts with its patients) then this results in a non-negligible key management overhead.

III. Design

Our system assumes smart devices that collect user related data (such as smart watches that measure cardio activity) and store them in a public Cloud. All these devices interact with the Cloud through a user controlled *gateway*. This gateway has the roles of the PKG and ACP described in the previous section. All communications (between the smart devices and the gateway and between the gateway and the Cloud) are secured using TLS. Data storage is implemented using the following steps:

- Initially the gateway executes the IBE-PRE *setup* algorithms and generates the user's master secret key (MSK) and the corresponding (public) system parameters. The MSK is then securely stored in the gateway. Moreover, the gateway generates a secret key (SK) that corresponds to the user's *identity*. The form and the semantics of a user identity are application specific.
- The user defines access control policies specifying the public keys that can access the data generated by each device. These policies are also stored in the gateway. For each policy the gateway generates a URI of the form <gateway FQDN/access control policy name>. We will refer to this URI as URI_{policy}.
- For each data item that arrives in the gateway, the gateway generates a symmetric encryption key K, encrypts the item using K (we refer to the output as $Enc(key)$), and encrypts K using the IB-PRE *encrypt* algorithm, with the user's identity as input (we refer to the output as $C_{ID}(K)$).
- The gateway stores $Enc(key)$, $C_{ID}(K)$, and URI_{policy} in the Cloud.

All potential *clients* that want to access some data stored in the Cloud must have generated a public/private key pair. The public part of this pair (which we refer to as PK_{client}) is the key used by data owners when defining access control policies. Data access is implemented using the following steps:

- The client sends a data request to the Cloud. The Cloud responds with URI_{policy} and a *token*.
- The client communicates with the gateway of the user (located in gateway FQDN), and authenticates himself. The authentication procedure is application specific. For example, it can be implemented by having the client digitally sign a gateway-generated nonce using his private key. When the authentication procedure is completed, and providing that the client is authorized to access data items protected by URI_{policy}, the gateway generates the appropriate *authorization* and uses the IB-PRE *RKGen* algorithm to generate the (public) re-encryption key $RK_{ID\rightarrow PK_{Client}}$.
- The client sends a new data request to the Cloud, including this time the authorization and the re-encryption key. The Cloud provider validates the authorization, and if it is valid, it re-encrypts $C_{ID}(K)$ and sends the new ciphertext along with $Enc(key)$ to the client.
- Provided that the client is indeed the owner of the PK_{client} used during the authentication process, he is able to decrypt the re-encrypted version of K and then use K to decrypt the file.

Figure 3 gives an overview of our system. It should be noted that access control policies and re-encryption keys are re-usable. Therefore, if a client, authorized for a particular URI_{policy}, requests another item protected by the same policy, then communication with the gateway is not required.

A. Group policies

It is often desirable to organize potential clients in a group and define access control polices based on these groups. For example, "doctors of hospital A" could be considered a group. Contemporary cryptographic techniques such as attributed-based encryption [3], or hierarchical identity-based encryption [4] could be used to achieve this goal. However, we do not consider this option, because, for security reasons, we want each client to be able to generate her keys by herself, which is not possibly with these cryptographic techniques. Instead, we follow a more conservative approach. We assume that each group is identified by a public key. This key is known to data owners and it is used during access control definition. Moreover, each group member has generated a public/private key pair. The public part of this pair is included in a X.509 certificate which is digitally signed using the private key of the group. For instance, in our example the public keys of the doctors should be signed by the private key of the group "doctors of hospital A". If a client belongs to multiple groups, he should have multiple X.509 certificates.

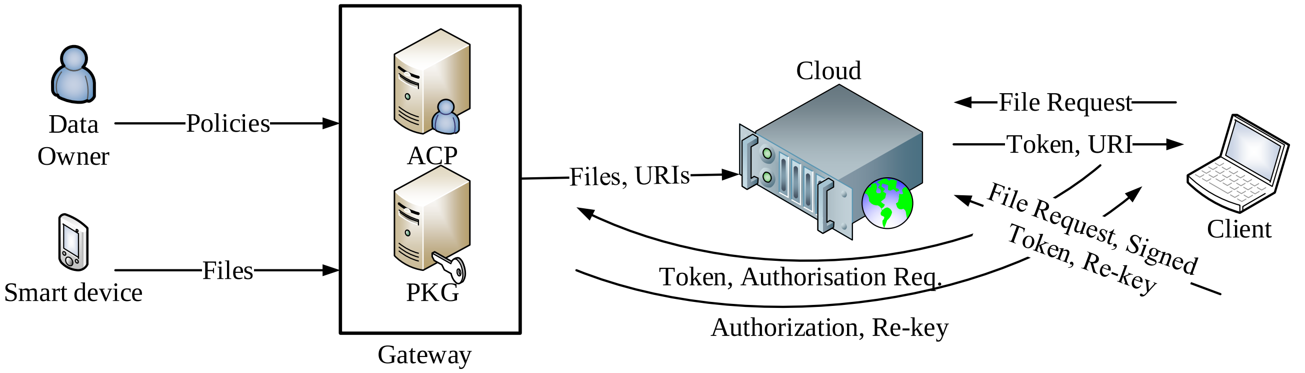

Fig. 3. Design overview

When a client requests authorization from the gateway, she includes in her request her digital certificate. The digital signature included in the certificate is used by the gateway in order to evaluate whether or not the client belongs to an authorized group. If this is the case, then the gateway can use the public key of the client (included in the certificate) to perform the proxy re-encryption procedure described above, and therefore to allow the user access to the data. Note that the gateway does not need to know or store any details about the members of the group; it only needs to know the public key of the group.

IV. Evaluation

We have implemented the IB-PRE part of our system by modifying the Green-Ateniese IBE-PRE implementation included in the Charm Crypto library [5] to support RSA public keys for the delegatee. In order to achieve a security level equivalent to RSA with a key size of 1024 bits for the encryption of the symmetric key, the size of $C_{ID}(key)$ is 3232 bits, and the size of a re-encryption key is 1536 bits. In an Ubuntu 12.04 Desktop machine, running in a single core of an Intel i5-4440 3.1 GHz processor with 2GB of RAM, the creation of $C_{ID}(key)$ required 40 ms, the creation of a re-encryption key required 20 ms, the re-encryption of a ciphertext required 31 ms, and the decryption of a ciphertext required 28 ms.

The IB-PRE cryptographic algorithm used by our system has been proven to be secure in [2]. Moreover, the access control delegation solution used by our system exhibits many advantages: it is generic enough, it can be easily implemented by a Cloud provider, data can be easily transferred between Cloud providers that implement this solution, it protects client privacy against third parties (including the Cloud providers), and it allows easy modification of access control policies [1].

Each data item is encrypted using a different symmetric encryption key, therefore, the compromise of a symmetric encryption key would require the re-encryption of that specific item only with another fresh key. This is an inevitable overhead of all similar systems and it is due to the fact that public key encryption cannot be applied directly to the file contents, due to its computation complexity. Nevertheless, for small data items, such as readings from wearable devices, it may be possible to negate the need for symmetric encryption.

V. Related work

Löhr et al. [6] have proposed a solution for securing e-health clouds based on *Trusted Virtual Domains* (TVDs). TVD is a virtualization technique that creates secure “sandboxes” where user data can reside. This solution is orthogonal to our system: the solution by Löhr et al. concerns the design of secure clouds specific to e-health services, whereas our solution assumes a generic cloud service and builds a secure data sharing system on top of it.

Wu et al. [7] propose an access control mechanism for sharing electronic health records in the Cloud. The main component of their mechanism is an *access broker* that is responsible for enforcing access control policies. The access broker is an entity shared among many stakeholders, therefore, privacy concerns are raised. In our work, access control policies are enforced by data owners in a way that reveals no information about data owners or clients to third parties (including the Cloud provider). Son et al. [8] propose a mechanism that supports “dynamic” access control, i.e., access control that takes into consideration the user’s context. In their solution, access control is also implemented in the Cloud, therefore the same privacy concerns are raised.

Fabian et al. [9] use attribute-based encryption (ABE) to protect medical data stored in multi-Cloud environments and shared among different cooperative organizations. ABE produces encrypted data in a way that only users with specific “attributes” can decrypt. In essence, ABE incorporates access control policies into ciphertexts. The disadvantage of using ABE for this purpose is that the loss of a private key that corresponds to an attribute requires the generation of a new key, the distribution of this key to all users that have this attribute, and the appropriate encryption of all files protected by this attribute. In contrast, in our system the loss of the data owner’s secret key only requires a new encryption of all symmetric keys. Similarly, [10], [11] use attribute-based encryption to protect personal health records stored in public cloud environments; these solution also suffer from the same problems.

Thilakanathan et al. [12] use ElGamal public key encryption and a proxy re-encryption like protocol to protect generic health data stored in the cloud. Their solution relies on a centralized trusted third party that generates private keys on behalf of users. In our system users generate their private keys by themselves, therefore our approach offers increased security.

VI. Conclusion and future work

In this paper we presented a solution that allows secure and privacy preserving storage of medical data in public Clouds, by leveraging access control delegation and proxy re-encryption. Our solution is based on a gateway-based design, where a user controlled gateway is responsible for encrypting user generated data, as well as for enforcing access control policies.

Future work involves the transfer of the encryption process to the devices that generate the data. In this manner, the device could store the data directly to the Cloud, avoiding the gateway, therefore reducing communication overhead. Currently, our work assumes that devices can be securely authenticated to the gateway and, similarly, the gateway can be securely authenticated to the Cloud. Future enhancements of our system would also consider and implement these functions.

References

[1] N. Fotiou, A. Machas, G. C. Polyzos, and G. Xylomenos, "Access control as a service for the cloud," *Journal of Internet Services and Applications*, vol. 6, no. 1, pp. 1–15, 2015.

[2] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds. Springer Berlin Heidelberg, 2007, vol. 4521, pp. 288–306.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 89–98.

[4] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology EUROCRYPT 2005*, ser. Lecture Notes in Computer Science, R. Cramer, Ed. Springer Berlin Heidelberg, 2005, vol. 3494, pp. 440–456.

[5] J. Akinyele, C. Garman, I. Miers, M. Pagano, M. Rushanan, M. Green, and A. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[6] H. Löhr, A.-R. Sadeghi, and M. Winandy, "Securing the e-health cloud," in *Proceedings of the 1st ACM International Health Informatics Symposium*, ser. IHI '10. New York, NY, USA: ACM, 2010, pp. 220–229.

[7] R. Wu, G.-J. Ahn, and H. Hu, "Secure sharing of electronic health records in clouds," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, Oct 2012, pp. 711–718.

[8] J. Son, J.-D. Kim, H.-S. Na, and D.-K. Baik, "Dynamic access control model for privacy preserving personalized healthcare in cloud environment," *Technology and Health Care*, vol. 24, no. s1, pp. S123–S129, 2015.

[9] B. Fabian, T. Ermakova, and P. Junghanns, "Collaborative and secure sharing of healthcare data in multi-clouds," *Information Systems*, vol. 48, pp. 132 – 150, 2015.

[10] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 131–143, Jan 2013.

[11] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Generation Computer Systems*, vol. 52, pp. 67 – 76, 2015, special Section: Cloud Computing: Security, Privacy and Practice. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14002076

[12] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, and L. Alem, "A platform for secure monitoring and sharing of generic health data in the cloud," *Future Generation Computer Systems*, vol. 35, pp. 102 – 113, 2014, special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.