A DEPENDABLE AND SECURE AUTHORISATION SERVICE IN THE CLOUD

Christian Neuhaus, Martin von Löwis and Andreas Polze
Operating Systems and Middleware Group, Hasso-Plattner-Institute, Potsdam, Germany
{*christian.neuhaus, martin.vonloewis, andreas.polze*}*@hpi.uni-potsdam.de*

Keywords: cloud storage; security; access control; threshold cryptography; information dispersal; authorization service;

Abstract: Cloud-based exchange of sensitive data demands the enforcement of fine-grained and flexible access rights, that can be time-bounded and revoked at any time. In a setting that does not rely on trusted computing bases on the client side, these access control features require a trusted authorization service that mediates access control decisions. Using threshold cryptography, we present an implementation scheme for a distributed authorization service which improves reliability over a single service instance and limits the power and responsibility of single authorization service nodes.

1 INTRODUCTION

Healthcare is one of the most costly and complex undertakings of a country. A high degree of division of labour and specialisation require a close collaboration of thousands of people and institutions. For data exchange in this scenario cloud storage services are an appealing option, as they require hardly any up-front investments into infrastructure, scale quickly to the current demand and provide high availability and performance. However, sensitive patient information may never be accessible to commercial storage providers and therefore has to be encrypted. Additionally, fine-grained, time-bounded and revokable access rights have to be enforced over the exchanged data. This is usually achieved by mediating access control decisions by a trusted authorization service instance (Neuhaus et al., 2011).

The **problem** with this scheme is that the authorization service represents a single-point of failure: If it is unavailable, no data access can be made. The challenge is to provide an implementation of the authorization service which is both secure and dependable. Simply replicating the service over several instances poses a risk to security, since an attacker has to take over only a single one of these instances to gain full control. The general approach to unify dependability and security is the idea of *Fragmentation-Redunancy-Scattering* (Fabre et al., 1994): Confidential information is broken up into insignificant pieces which can be distributed over several network nodes.

The **contribution** of this paper is a scheme for a distributed authorization service. We present a software architecture for fine-grained access control on data items in a distributed system. The achitecture satisfies the requirements of revokable access rights and cryptographic keys. To make the system secure and at the same time scalable and dependable, we make use of a threshold encryption scheme to limit the power of a single authorization service instance and provide redundancy.

The rest of this paper is structured as follows: We state the requirements for our solition in section 2. In section 3, we review related work and explain why the past approaches cannot fulfil our requirements. We explain theoretical foundations and algorithms in section 4. Our approach, the distributed license service, is explained in section 5. We discuss and summarize our approach in section 6.

2 REQUIREMENTS

Because of the strict regulations for the electronic processing of patient data and general security concerns, a solution for cloud-based data exchange in a healthcare scenario has to be highly secure while providing flexible access rights and high availability. Therefore, we assume the following requirements:

In terms of **security**, data may only be accessible to authorized end-users. The cloud provider is not regarded as trusted and may under no circumstances get access to data items stored on his infrastructure. Access rights for a data object have to be specified in a security policy assigned to that data object. It is required that these access rights can be specified for every security principal (i.e. user) of the system independently. It has to be possible to limit these rights to a certain timeframe and revoke any of these rights at any time. In terms of **availability**, it is required that the enforcement mechanism of these access rights does not have an unreasonably negative impact to the reliability of the system.

3 RELATED WORK

The advent of cloud computing (Armbrust et al., 2009) has the potential of great change: The amount of data generated and exchanged all over the world is growing rapidly, and a lot of it is going to be outsourced to cloud storage services (Gantz and Reinsel, 2011). The *Oceanstore*-Project (Kubiatowicz et al., 2000) explored questions of data locality and data movement in a worldwide-spanning storage network for good performance and availability. Several publications revolve around the question of availability and dependability (Laprie, 1985) of cloud storage services. Potential outages and failures of individual cloud storage services can be masked by employing redundancy, using RAID-like encoding techniques (Schnjakin et al., 2011; Bowers et al., 2009), information dispersal (Rabin, 1989) or secret sharing techniques (Shamir, 1979) in projects like *DepSky* (Bessani et al., 2011).

3.1 Encryption

The method of choice to achieve security and implement access control by technical means is usually the protection of data by encryption. Encryption of data on the client-side hides it from the cloud storage provider and makes it accessible only to clients in possession of the appropriate cryptographic keys. In the previously mentioned projects (*DepSky*, *Oceanstore*, etc.) client-side encryption is proposed in different levels of detail but with one limitation: Security by client-side encryption is only considered for a single-user scenario, where a single user holds the cryptographic keys for his data. Sharing of data items and access rights between a group of users is not intended. Shared access and sharing of access rights on particular data items is however required in our scenario. In multi-user access scenarios, the access rights on data items are usually specified in an explicit data structure, such as an *access control list* or a *security policy* (see section 4.1). Several approaches exist on how security policies can be technically enforced: One approach is to implement them entirely by means of cryptography: Several works were published that extend the original idea of public-key cryptography (Rivest et al., 1978; Diffie and Hellman, 1976) by a multi-user aspect. Key generation and management schemes can be used to generate cryptographic user keys and encryption keys for data items. The generated keys enable decryption of the content for the right combination of user and data keys as specified in an access control matrix (e.g. (Zych et al., 2006)). A recent development in cryptography is the approach of *attribute-based encryption* (ABE) (Sahai and Waters, 2005; Bethencourt et al., 2007): Cryptographic user keys are generated in such a way that they reflect which *attribute* the user possesses (e.g. *"nurse"*, *"radiology staff"*). Protected data items can only be decrypted by someone in possession of a user key which matches a certain criterion of attributes. An important property of ABE-schemes is *collusion resistance*: User keys cannot be combined to join their set of assigned attributes. These schemes can be used to implement security policies through cryptography (e.g. (Akinyele et al., 2010)).

However, as these approaches do not mediate access control decisons they do not provide the possibility to revoke access rights: Once a cryptographic key is transferred to a user it cannot be taken away anymore.

3.2 Trusted authorization services

The revocation of access privileges can be achieved by introducing an indirection in the authorization process: Instead of granting the access token (i.e. a cryptographic key) directly, this token has to be requested from a centralized trusted authorization service. When a user requests access to a protected resource, the authorization service checks the requesters privileges against a security policy or access control matrix. If the user has sufficient privileges, the access token is granted. The revocation of rights can be achieved by blacklisting privileges or users on the authorization service. This concept has been implemented as *mandatory access control* using a *reference monitor* (Anderson, 1972) in *Kerberos* (Kohl and Neuman, 1993). A similar concept is shown for a distributed document repository (Katt et al., 2009), largely following the XACML-nomenclature. The concept of a trusted authorization service can also be used with cryptographic access tokens. An approach that relies on cryptography is the development and research around Microsofts Cloud-Storage *CS2* (Kamara and Lauter, 2010; Kamara et al., 2011), which uses trusted services to help with access control decisions. The idea of a semi-trusted mediator service is also considered to extend the flexibility of attribute-based encryption schemes (Ibraimi et al., 2009).

However, a single instance of an authorization services puts the dependability of the system at risk, since it is a single point of failure. Therefore, distributed approaches have been proposed in the literature: A combination of a distributed authorization service and trusted computing is presented by (Abghour et al., 1999) to guarantee tolerance against byzantine failures of single nodes.

The FADE system (File Assured Deletion) (Tang et al., 2010) is similar to our approach in its objectives. A *key manager service* provides enforcement of access policies using cryptographic techniques. Deletion can be assured by having the key manager erase private keys associated with access policies; the corresponding file data then might remain available in the cloud in an encrypted form, but the plain text becomes unrecoverable. The specific cryptographic techniques in FADE differ significantly from our approach, though: while FADE employs blinded RSA encryption to hide keys for data items from the authorization service, this becomes obsolete by distributing the authorization service over several instances. We believe that the overhead of the blinded RSA approach does not provide additional security. In particular, the threats to FADE and to our approach are the same.

4 FOUNDATIONS

4.1 Security Policies

A security policy (McLean et al., 1994) is a document that specifies the property of *security* of a system. In a very formal view, a system can be seen a finite-state machine. A security policy can then separate the set of states into two disjoint subsets. The states of one subset are considered as *authorized*, the others as *unauthorized*. We use a more practical approach, where security policies serve as confidentiality policies: For sharing data between users, policies specify access rights to data items and can be considered as security metadata: They provide additional, machine-interpretable information that specifies which subject (e.g. a natural or legal person) has which rights (e.g. read access) to which resource (e.g. a data item).

4.2 Threshold Cryptography

A *(m,n) threshold cryptography scheme* (see (Shamir, 1979)) is a cryptographic scheme that can be used to divide a piece of data D into n shares of data $D^1 \dots D^n$, with the following properties:

- Knowing a subset of m or more shares of the n shares generated, D can easily be computed.
- With less than m pieces of the n shares available, the original value of D cannot be reconstructed.

The value of the parameter m can be chosen freely in the range of $1 \leq m < n$. The length of each generated share $D^1 \dots D^n$ is the same as the length of D.
The implementation of the scheme proposed by (Shamir, 1979) is based the interpolation of polynomials: The shares $D^1 \dots D^n$ are distinct data points of a randomly chosen polynomial of degree $m-1$, the constant term of which is D. With at least m shares available, the polynomial can be reconstructed and the value of D can be obtained.

4.3 Hash-based Message Authentication Codes

A *Hash-based authentication code* (HMAC) is a message authentication code (MAC) based on cryptographic hash function. The purpose of a MAC is to enable the receiver of a message to verify both the integrity and the authenticity of a message. Using the message, a cryptographic hash function H (such as SHA-1), a secret key K and two fixed padding values (*ipad, opad*), a HMAC is computed in the following way, as suggested by (Krawczyk et al., 1997):

$$H(K \oplus opad, H(K \oplus ipad, text))$$

The HMAC is computed by the sender of the message and sent along with it. It is also computed by the receiver of the message and compared to the received HMAC. If both values match, the message is unchanged and thus authenticated. The integrity and authenticity of the message are protected by the fact that the secret key is necessary to compute a valid HMAC.

5 DISTRIBUTED AUTHORIZATION SERVICE

This section illustrates our solution for a distributed authorization service that can cryptographically enforce revokable and time-constrained access rights of security policies on *data objects*, fulfilling the requirements stated in section 2. A data object holds arbitrary information in a byte array and is assigned a *security policy*, which states the access rights that users have on this object. For more details on these security policies, see (Neuhaus et al., 2011).

5.1 Distribution Mechanism

The service is distributed over n nodes by using a threshold cryptography scheme (see section 4.2). The underlying protection mechanism encrypts every data object with a randomly generated key. This key is then broken up into n shares, m of which suffice to reconstruct the key. m can be freely chosen within $1 \leq m < n$ to balance security and dependability. Each share is made available to exactly one particular authorization service instance. A quorum of m of these instances has to be queried by a client to reconstruct the data object key and get access to a data object. Depending on parameter m, a certain number of these nodes can fail or fall under hostile control without affecting availability of the service or security of the protected data.

5.2 System overview

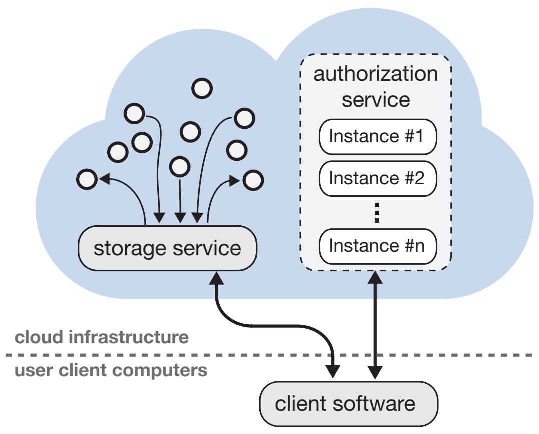

Figure 1: Overview of the system and its building blocks.

The users of this system interact with it using a **client software** library, which runs on the device they are using. It it assumed that every user authenticates himself to this client software using public key cryptography. The users' private key is available to the client software, and the users public key is assumed to be known throughout the system.

Data is stored in the cloud by the **storage service**. This service provides a simple interface to store and retrieve data, using the classic four *CRUD*-operations: Create, Read, Update and Delete. The *storage service* is assumed to run on possibly untrusted infrastructure. All data sent to and stored by the service has previously been encrypted and is therefore inaccessible to the service and its operator.

Access control decisions are made by the **authorization service** by checking the security policy of a data item. The service is distributed over n instances. To get access to a data item, the *client software* has to query a number of $m < n$ instances of this service. Every instance of this service holds its own public-private-keypair, the public key of which is assumed to be known throughout the system. No single instance of this service holds the cryptographic key material to make this access decision alone. Instead, a client software needs positive answers from at least m instances of this service.

5.3 Placement of Services & Trust

It is important to note that this approach can only keep its promises of dependability and security if the instances of the *authorization service* each run in completely independent locations on a cloud infrastructure. For achieving dependability, the different instance locations have to be structurally independent and thus are unlikely to fail simultaneously. For achieving a gain in security, these service instances may not be accessible to a common operator. The service operators of the authorization service instances have to be trusted not to influence the computations of the services, not cooperate using the cryptographic keys of the service instances to escalate their rights and to always provide accurate time information to the services.

5.4 Cryptographic protection of data

Every *data object* that enters the system through the client software has to carry a security policy and the payload data. When a data object is uploaded to the storage service, it is prepared by the client-software the following way:
For the data item, a random symmetric key K_{RND} is generated. This key is then broken up into n shares by using a threshold cryptography scheme (see section 4.2). The x-th share is denoted K^{x}_{RND}. All n instances of the authorization service and their corresponding public keys are known to the client software. The public key of a particular instance m of the authorization service is denoted as K^{+}_{m}. The data package sent to the storage service is then assembled by filling its data fields as shown in Table 1.

Table 1: Data fields in a data item and cryptographic keys they are encrypted with

Data Field	**Encryption Key**
Payload Data	K_{RND}
Policy	- not encrypted -
HMAC(K^1_{RND}, Policy), K^1_{RND}	K_1^+
HMAC(K^2_{RND}, Policy), K^2_{RND}	K_2^+
...	...
HMAC(K^n_{RND}, Policy), K^n_{RND}	K_n^+

The payload data is symmetrically encrypted under key K_{RND}. The security policy is added to the data object unencrypted. For every instance x of the authorization service, the following data is encoded under its public key K_x^+:

- The x-th share of the dispersed key K_{RND}, denoted as K^x_{RND}.
- A message authentication code (HMAC, see section 4.3) over the policy, using K^x_{RND} as a signing key.

By asymmetrically encrypting it with K_x^+, the key share K^x_{RND} is is a shared secret between the creator of the data and the n-th instance of the authorization service. It is sent to a client software requesting access to a data item if the security policy allows it. K^x_{RND} is also used to create a message authentication code (see section 4.3) over the policy, enabling the authorization service instance to check the integrity of the transmitted policy.

5.5 Data access

In this section we describe the sequence of steps of a user client software accessing a data object in the system. We assume that the user of the client software has made his personal public-private key pair available to it.

1. The client software downloads a data object from the storage service. This data object contains the data fields shown in table 1.
2. The client software sends requests for data access to a quorum of at least m instances of the authorization service. The following information is sent to the x-th instance: The users public key, the policy and the cryptographic information prepared for this instance: The message authentication code HMAC(K^x_{RND}, Policy), and the key share K^x_{RND}.
3. Each instance of the authorization service decrypts HMAC and key share. HMAC and the key share are used to check the integrity of the transmitted policy. If the policy is valid, the instance checks whether the requested access right is granted to the requesting user by the policy. If the right is granted, the service instance sends back the key share K^x_{RND} to the requesting client software, encrypted under the users public key. If not, the request is discarded.
4. The answers from the authorization service instances are decrypted using the users private key. After receiving at least m positive answers with shares of the data key from authorization service instances, the client software reconstructs the original data key K_{RND}.
5. Using K_{RND} the data payload of the package is decrypted and is available to the user.

6 CONCLUSION

We have presented a software architecture for using cloud-based storage in the context of medical applications. We discussed an approach for achieving privacy and access control, including the ability enforce revokable and time-bounded access rights. This work builds on earlier work where we demonstrated how the medical records need to be split into fragments so that cryptographic methods can guarantee the required selective access rights.

Our approach supports the enforcement of revokable and and time-bounded access rights by using mediated access control: Time-limits and revocations of rights are enforced by the authorization service instances. A system with n nodes of the authorization service with a m,n (where $m < n$) encryption scheme remains functional with at least m nodes available. Stored data remains secure with less than m nodes under hostile control.

For future work, we still need to evaluate our approach from a performance point of view. We expect that the specific outcome of the performance measurements will depend on the specific cloud provider to be selected, and that such evaluation therefore will have limited value. The key point of this evaluation will be the comparison of the cryptographic storage compared to unsecured storage in the cloud. It remains an open question whether such an approach will actually be adopted by users in the medical domain. This will depend on many factors, including political decisions; in our work, we can, however, only deal with scientific and engineering aspects.

REFERENCES

Abghour, N., Deswarte, Y., Nicomette, V., and Powell, D. (1999). Specification of authorisation services. *Maftia project ist*, 11583.

Akinyele, J., Lehmann, C., Green, M., Pagano, M., Peterson, Z., and Rubin, A. (2010). Self-protecting electronic medical records using attribute-based encryption. Technical report, Cryptology ePrint Archive, Report 2010/565, 2010. http://eprint. iacr. org/2010/565.

Anderson, J. (1972). Computer security technology planning study. volume 2. Technical report, DTIC Document.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2009). Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*.

Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2011). Depsky: Dependable and secure storage in a cloud-of-clouds. In *Proceedings of the sixth conference on Computer systems*, pages 31–46. ACM.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE.

Bowers, K., Juels, A., and Oprea, A. (2009). HAIL: A high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 187–198. ACM.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654.

Fabre, J., Deswarte, Y., and Randell, B. (1994). Designing secure and reliable applications using fragmentation-redundancy-scattering: an object-oriented approach. *Dependable Computing—EDCC-1*, pages 21–38.

Gantz, J. and Reinsel, D. (2011). Extracting value from chaos. *IDC research report IDC research report, Framingham, MA, June. Retrieved September*, 19:2011.

Ibraimi, L., Petkovic, M., Nikova, S., Hartel, P., and Jonker, W. (2009). Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application. *Information Security Applications*, pages 309–323.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. *Financial Cryptography and Data Security*, pages 136–149.

Kamara, S., Papamanthou, C., and Roeder, T. (2011). Cs2: A semantic cryptographic cloud storage system. Technical report, Technical Report MSR-TR-2011-58, Microsoft Research, 2011. http://research. microsoft. com/apps/pubs.

Katt, B., Breu, R., Hafner, M., Schabetsberger, T., Mair, R., and Wozak, F. (2009). Privacy and access control for ihe-based systems. *Electronic Healthcare*, pages 145–153.

Kohl, J. and Neuman, C. (1993). The kerberos network authentication service (v5).

Krawczyk, H., Bellare, M., and Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational). Updated by RFC 6151.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., et al. (2000). Oceanstore: An architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201.

Laprie, J. (1985). Dependable computing and fault-tolerance. *Digest of Papers FTCS-15*, pages 2–11.

McLean, J., Schell, R., and Brinkley, D. (1994). Security models. *Encyclopedia of Software Engineering*.

Neuhaus, C., Wierschke, R., von Löwis, M., and Polze, A. (2011). Secure cloud-based medical data exchange. *Lecture Notes in Informatics (LNI) - Proceedings*, P-192.

Rabin, M. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473.

Schnjakin, M., Alnemr, R., and Meinel, C. (2011). A security and high-availability layer for cloud storage. In *Web Information Systems Engineering–WISE 2010 Workshops*, pages 449–462. Springer.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

Tang, Y., Lee, P., Lui, J., and Perlman, R. (2010). Fade: Secure overlay cloud storage with file assured deletion. *Security and Privacy in Communication Networks*, pages 380–397.

Zych, A., Petkovic, M., and Jonker, W. (2006). Key management method for cryptographically enforced access control. In *Proc. of the 1st Benelux Workshop on Information and System Security, Antwerpen, Belgium*.