Detection of Access Control Violations in the Secure Sharing of Cloud Storage

Carlos André Batista de Carvalho[1,2], Rossana Maria de Castro Andrade[1], Nazim Agoulmine[3] and Miguel Franklin de Castro[1]

[1]*Computer Science Department, Group of Computer Networks, Software Engineering, and Systems (GREat), Federal University of Ceará (UFC), Fortaleza, Brazil*

[2]*Computer Science Department, Federal University of Piauí (UFPI), Teresina, Brazil*

[3]*IBISC Laboratory, University of Evry (UEVE), Evry, France*

Keywords: Secure Storage, Data Sharing, Access Control Violations.

Abstract: A cloud storage service implements security mechanisms to protect users data, including an access control mechanism to enable the data sharing. Thus, it is possible to define users permissions, granting the access only to authorized users. Existing solutions consider that the provider is honest but curious so that the designed mechanisms prevent the access to the files by the provider. However, the possibility of executing illegal transactions is not analyzed, and a malicious provider can perform transactions requested by unauthorized users, resulting in access control violations. In this paper, we propose monitoring and auditing mechanisms to detect these violations. As a result, new attacks are identified, especially those resulting from writing actions requested by users whose permissions were revoked. Colored Petri Nets (CPNs) are used to model and validate our proposal.

1 INTRODUCTION

Cloud computing is a distributed computing paradigm that enables the sharing of computational resources between many clients. It is possible to reduce the infrastructure costs by contracting a public cloud provider and paying only for the consumed resources. Besides, the services' scalability allows the dynamic allocation of resources, in accordance with customers' needs. However, this technology comes with the main drawback of losing control over the data stored in the cloud infrastructures. Therefore, there is some resistance in adopting public clouds, due to concerns about security and privacy (Ardagna et al., 2015) (Luna et al., 2015).

Among security concerns (*e.g.*, data loss and leakage, resource location, service disruption and multi-tenancy issues), (Rong et al., 2013) stress that the big concern is the assurance of the security in cloud storage. Cloud providers have developed security mechanisms based on frameworks and security guidelines elaborated by standardization bodies, such as ISO (International Organization for Standardization), NIST (National Institute of Standards and Technology) and CSA (Cloud Security Alliance) (Luna et al., 2015). In Amazon S3 (Simple Storage Service), for example, the SSL (Secure Socket Layer) protocol is used to protect data transmission, and the AWS KMS (Key Management Service) to protect data at rest (Amazon, 2017a). The KMS facilitates the creation, storage and distribution of keys. Besides, the permission management, based on Access Control Lists (ACLs), enables secure data sharing (Amazon, 2017b).

Security mechanisms are designed to prevent or detect attacks, and the success of an attack results in the violation of a security property (Stallings, 2016). An access control mechanism, for example, is used to allows the definition and updating of files' permissions. In addition, the providers can be "honest but curious", requiring cryptographic solutions so that the files cannot be accessed by the providers (Tiwari and Gangadharan, 2015b) (Jiang et al., 2014). In this context, the key management becomes essential, being responsible for updating the keys and their distribution only for authorized users (Popa et al., 2011).

However, it is not possible to avoid all kind of attacks, especially those performed by malicious providers. Therefore, the cloud customers request more transparency and security guarantees from providers

Batista de Carvalho, C., de Castro Andrade, R., Agoulmine, N. and Franklin de Castro, M.
Detection of Access Control Violations in the Secure Sharing of Cloud Storage.
DOI: 10.5220/0006698701240135
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 124-135
ISBN: 978-989-758-295-0
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

(Ardagna et al., 2015). In this context, scientific research has been realized to develop solutions that improve the trust in cloud providers (Jin et al., 2016) (Hwang et al., 2014a) (Tiwari and Gangadharan, 2015a). These solutions can detect malicious behaviors that result in violations of security properties.

A malicious provider can, for example, ignore the verification of access control, performing unauthorized transactions. This malicious behavior results in a violation of access control that must be detected. The existing mechanisms can detect, for example, an illegal writing when this writing uses invalid keys resulting in integrity violations. On the order hand, the cloud customer can revoke a user's write permission, but this user can use an older encryption key and try to write a new file (Kallahalla et al., 2003). In this case, if the malicious provider commits this writing, no integrity violation will be detected.

In this paper, we propose monitoring and auditing mechanisms to detect access control violations in a cloud storage service avoiding the provider to deceive or deny the violation detection. An honest broker that manages the cloud transactions (*e.g.*, reading and writing) and informs users the current state of the system, enabling the real-time detection. A private cloud or other Trust Third Party (TTP), for example, can acts as a honest broker. Otherwise, the broker can be dishonest, making the auditing necessary to identify the violations.

The rest of this paper is organized as follows. In Section 2, we detail the concepts related to secure sharing in cloud storage. The proposed mechanisms and the security evaluation are described in Section 3. Next, the related work is discussed. Lastly, Section 5 presents the final considerations and future work suggestions.

2 SECURE SHARING IN CLOUD STORAGE

In this section, we present the mechanisms, for access control and violation detection, used as the basis for this research. In addition, we enumerate attacks that result in access control violations.

2.1 Access Control

The security of the stored data and file sharing are typical requirements of a cloud storage service. Then, existing solutions include an access control mechanism, allowing the file sharing while preventing data leakage. In these solutions, a cloud customer (or client administrator) can purchase the storage service, and define permissions for cloud users that can, at least, read and write files. The cloud transactions are managed by a broker and performed by a cloud provider. Figure 1 presents an overview of a cloud storage service, and Table 1 describes the roles of each actor of this service. Due to the possibility of the provider and the broker act maliciously, the access control violations can be detected by the users and the Third-Party Auditor (TPA).

Security mechanisms must be designed to protect a user against existing threats, which are, in the scope of this research, data leakage and corruption. In this context, these mechanisms provide data confidentiality and integrity, ensuring the data access or modification only by authorized users. An access control mechanism allows the cloud customer to define who can read and who can write in each file. The proxy re-encryption (PRE), Attribute-Based Encryption (ABE) and Access Control Lists (ACLs) are the approaches used to design access control mechanisms (Thilakanathan et al., 2014).

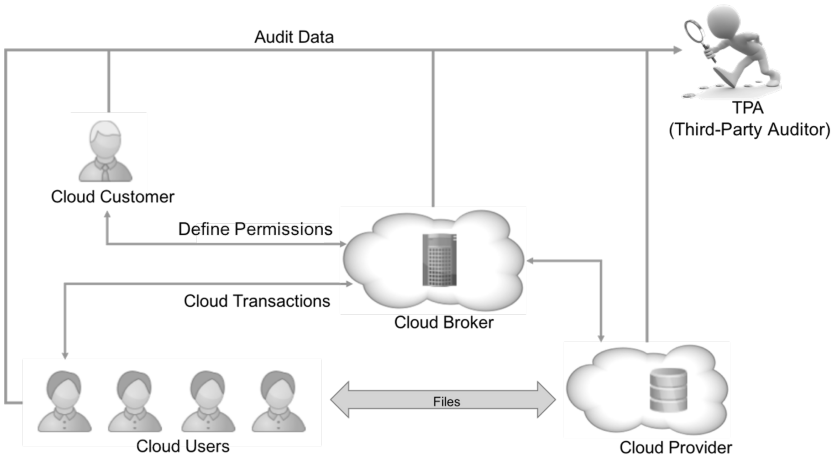

Figure 1: Storage service overview.

Table 1: Cloud actors definition.

Actor	Definition
Costumer	The person of an organization that administer the storage service. He/she is responsible for define/modify the permissions and upload new files.
User	An employee that can read or write files in accordance with his/her permissions. A user also checks the security properties based on the information received by the broker
Broker	The entity responsible for managing the transactions. It check permissions and sends the keys and the current status of a file.
Provider	The entity responsible for storing the files.
Third-Party Auditor (TPA)	The entity responsible for analyzing the transactions' logs to demonstrate the security or prove the occurrence of a violation.

The Access Control Lists (ACLs) has been used in solutions that include mechanisms for violation detection (Popa et al., 2011) (Jin et al., 2016). Due to its suitability to be used in conjunction with other security mechanisms, ACLs are used, in this research, to specify users' permissions. Each file has an ACL that contains a list of users with their permissions. The users have read or read/write permissions, and only the cloud customer can change the permissions, granting or revoking access. For efficiency purposes, it is possible to use one single ACL for authorizing the access to a group of files (Kallahalla et al., 2003).

In addition to setting permissions, it is essential to use cryptography algorithms to make effective the access control so that providers do not have access to the plaintext (Tiwari and Gangadharan, 2015b). A symmetric cipher is used for data confidentiality so that only authorized users can encrypt or decrypt the file, using the secret key (R). The data integrity is ensured based on digital signature in which the pair of keys (W and W_v) is used, respectively, to sign the file and verify its signature. These keys are generated by the process defined by the asymmetric cipher used for digital signature. The security of the cryptographic primitives used by cloud storage solutions is attested by the scientific community and outside the scope of this paper.

These keys that can be accessed in accordance with the users' permissions, enabling a granular access control. Then, the key management is essential to distribute the keys only to authorized users and update the keys whenever necessary. It is recommended to use different keys for each file and change the encryption key when the permissions are updated, following the guideline to limit the amount of data encrypted by a single key (Barker, 2016) and avoiding that revoked users can access a file (Kallahalla et al., 2003).

In a simple and costly solution, while W_v is kept public, R and W can be encrypted with the public key of each authorized user so that they are accessed only by them. Due to the inefficiency of this approach, Broadcast Encryption has been used (Popa et al., 2011) (Jin et al., 2016). The key and the group of authorized users (G) are the inputs of a broadcast encryption scheme (E_B), generating a ciphertext that can be deciphered only by users in G. (Boneh et al., 2005) detail a scheme that enables the broadcast encryption.

Figure 2 illustrates the layout of each stored file and its metadata, detailing how each key is used. The users with read or read/write permissions belongs to the set G_r and can decipher the $E_B(R_i, G_r)$ and extract the key used to encrypt/decrypt the file. W_i can be obtain by users with read/write permissions (*i.e.*, members of the set G_w). Thus, authorized users can modify a file, encrypt it and generate a valid signature.

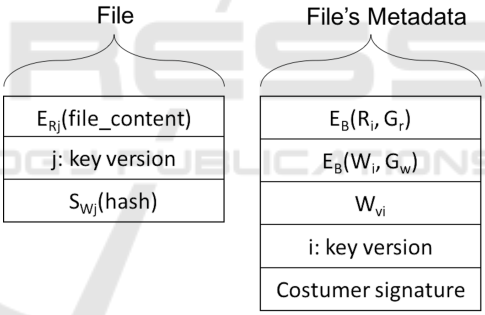

Figure 2: File and file's metadata layout.

In addition, there are different versions of the reading and writing keys, indicated by the indexes i and j. The key version used to encrypt the file can be different from the key indicated in the metadata due to the lazy revocation approach. This approach is frequently adopted to reduce the overhead when permissions are revoked (Jiang et al., 2014) (Tassanaviboon and Gong, 2011). With lazy revocation, the keys are updated, and the file is re-encrypted only in the next writing transaction. When the permissions are updated, the key version (index i) in metadata is also updated. However, the file is not re-encrypted so that its key version (index j) remains unchanged.

The Key Rotation scheme is used to simplify the keys updating when the permissions are changed and enable the lazy revocation (Kallahalla et al., 2003). In this scheme, only the customer can generate a se-

quence of keys from an initial key and a secret key so that when the customer modifies the permissions, he/she creates the next key version and signs the new metadata. After deciphering the file's metadata, authorized users can extract any previous version from the current version, obtaining the correct key used to cipher or sign a file. It is interesting to mention this scheme can create keys to be used by symmetric and asymmetric ciphers.

Besides the high cost of the immediate revocation, the possibility of a malicious user making copies of the files before losing his privileges encourages the use of this approach (Popa et al., 2011). However, it is necessary to mention the loss of security resulting of the lazy revocation because it enables the occurrence of access control violations. When a user is revoked, he/she leaves the set of users that can decipher the new metadata. Despite, the revoked user can keep the reading or writing key extracted from the previous metadata, making he/she able to access or modify the file if these keys are not updated. Therefore, it is essential to evaluate the impact of this approach in the access control.

In accordance with (Thilakanathan et al., 2014), the evaluation of the access control must analyze if revoked users really lost their privileges and if the provider cannot collude with an unauthorized user, granting access to this user without the customer's permissions. The collusion between entities must also be evaluated when a third party is included to mediate the communication between the users and the provider (Tiwari and Gangadharan, 2015b).

On the other hand, the assumption that the provider is "honest but curious" is common in this environment (Jiang et al., 2014) (Tiwari and Gangadharan, 2015b). In this context, access control mechanisms usually are evaluated to show the protection against data leakage, demonstrating that the provider and unauthorized users have not access to the keys and, consequently, cannot read files or write uncorrupted files. However, the possibility of the provider acting maliciously is not analyzed, and the execution of an illegal transaction (*e.g.*, unauthorized writing) results in a violation that must be detected.

2.2 Violation Detection

The access control is a prevention mechanism and must be combined with other security mechanisms to detect malicious behaviors from malicious internal entities, especially providers. Attacks from these entities not always can be avoided and result in security violations that must be identified. There are monitoring and auditing solutions for verification of security properties (Jin et al., 2016) (Tiwari and Gangadharan, 2015a).

In our preliminary paper (Carvalho et al., 2017a), we propose mechanisms to improve the violation detection, addressing attacks not analyzed by existing solutions. So, in this subsection, we present an overview of this solution and then highlight the threats related to access control violations that have not yet been analyzed by existing solutions (Carvalho et al., 2017b).

2.2.1 Log Analysis

The existing violation detection mechanisms are based on log analysis of cloud transactions. This analysis allows to prove the occurrence or not of violations of security properties. Existing solutions verify the integrity, retrievability, freshness and write-serializability of the data stored in the cloud. The integrity ensures the data modification only by authorized users. The retrievability is related to the data loss verification, and the freshness indicates the reading of the updated file. The write-serializability controls the writing order, ensuring that the new version of a file overwrites the last version of it.

In accordance with (Carvalho et al., 2017a), each log entry, called attestation, represents one cloud transaction and contains the following elements: **UserID**, **UserLSN**, **FileID**, **FileVersion**, **FileHash**, **KeyVersion**, **TransactionType**, **ChainHash** and **Signatures**. The UserID identifies each user and, the UserLSN represents the last sequence number used by each user. The FileVersion is essential to verify the freshness and write-serializability, and the FileHash to check the signature of the received file. The type of the transaction (*i.e.*, reading or writing) indicate whether the expect FileVersion must be replaced during an auditing. The KeyVersion allows a user to derive the correct key, using a Key Rotation scheme. The ChainHash is used to build the chain of attestations and is computed over the data of the current attestation and the ChainHash of the previous one. Last, all involved entities (*i.e.*, broker, user and provider) sign the attestation for non-repudiation purposes.

The verification of security properties is performed in real-time (by monitoring of the cloud transactions) and in auditing. The monitoring enables to detect violations early, reducing the damage resulted from, for example, the use of an invalid file (Hwang et al., 2014a). The users follow a protocol that enables to verify security properties while read or write files in the cloud storage. Figure 3 details, for example, the writing protocol. Based on the attestation sent by the broker, a user verifies the integrity and freshness of the stored file, or prepare a writing request that com-

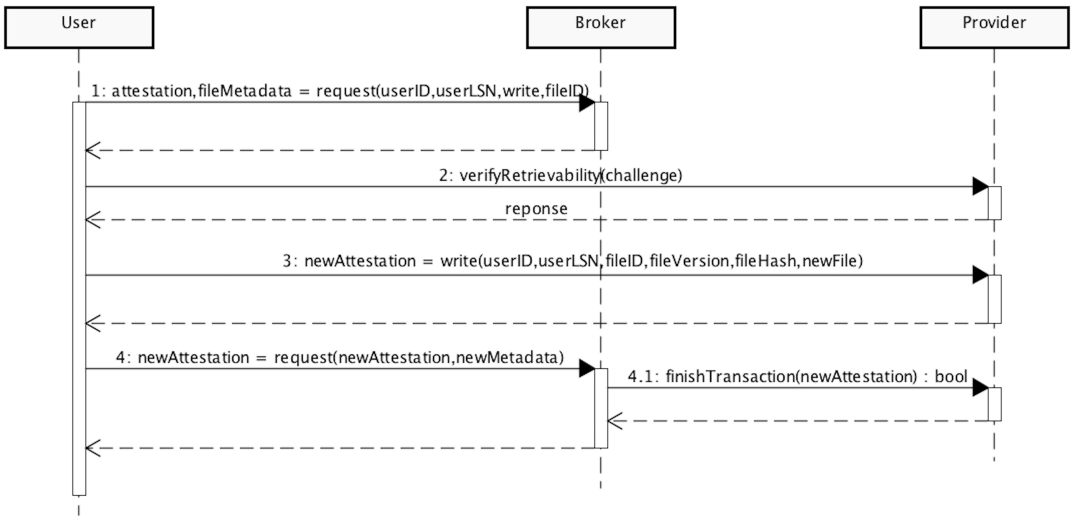

Figure 3: Writing a file.

plies with the write-serializability. However, the user must be sure that the received attestation is really the attestation regarding the last transaction, making indispensable the participation of an honest broker to manage the transactions. After an attack (*e.g.*, in next transaction), an inconsistency between the last attestation and the stored file is identified.

Before the audit, the provider must send all attestations to the TPA, and the broker and users send their last attestations. For each file, the TPA builds the chain of attestations, ordering them, and analyzes the sequence of the file's versions to prove that no violation occurs. Besides, it also checks the presence of all UserLSNs considering the chains of attestation of all files. The attestations sent by users and broker are used to verify whether the provider hides some transaction. Otherwise, the TPA report a violation. Due to the signature of attestations, no entity can deny a violation. The auditing also includes a Proof of Retrievability (PoR) scheme to check if scarcely accessed files were lost or not (Yang and Jia, 2012). After the audit, the attestations can be discarded, except the last attestation of each file that will be chained with the attestations of the new epoch[1].

2.2.2 Threat Model

The verification performed by existing solutions is not enough to detect all violations resulting from the execution of illegal transactions. The transactions are illegal when the malicious provider allows the reading or writing by an unauthorized user. Normally, the unauthorized users do not have the keys to read files, and an unauthorized writing is detected by existing solutions because the users cannot write files with a valid signature.

On the other hand, a user can be, for example, dismissed, and his permissions being revoked. In this case, this user can try to use his old credentials to access or modify files. Due to the lazy revocation, revoked users can read files if they are not re-encrypt yet. Besides, a malicious provider can ignore the access control, and a revoked user can write new files using old keys, which may be considered valid (Kallahalla et al., 2003). Therefore, besides an access control mechanism, a mechanism to verify the access control is necessary to check if the provider performs only authorized transactions.

These scenarios are not analyzed by existing solutions (see Section 4), not verifying the possibility of a malicious provider allowing the reading or writing by a revoked user. Besides, the broker displayed in Figure 1 can be trustworthy or malicious. So, a malicious broker can collude with the provider, trying to deceive violation detection, and must also be detected by the proposed mechanisms.

Lastly, there are transactions to modify permissions, requested only by the cloud customer. These transactions must be committed by the broker and provider in order to update the ACLs and file's metadata. The malicious entities can also ignore these transactions, allowing unauthorized readings and writings in the future.

[1]The period between two consecutive audits is called **epoch** (Popa et al., 2011)

3 DETECTING ACCESS CONTROL VIOLATIONS

In this paper, we extend our previous solution to identify also the access control violations. This section describes our proposal followed by the modeling and validation of the proposed mechanisms using CPNs (Colored Petri Nets), demonstrating that access control violations are detected.

3.1 Proposal

An access control mechanism is used by a secure storage service to provide data confidentiality and integrity in accordance with the permissions by the cloud customer. In a data sharing environment, authorized users read and write files in the cloud provider while the cloud customer can also update the permissions, granting or revoking privileges to users. The cloud transactions are stored in logs that are scanned for violation detection. Previously, only the reading and writing transactions are addressed.

In this context, it is necessary to specify how to represent the updating of permissions in the log. The attestation of a permission update transaction includes a new field (called **PermissionList**) with a list of the modified permissions, as shown in Figure 4. This field appears only in attestations of transactions to update permissions and each element of this list indicates the UserID and his/her new permission. In order to reduce the size of this list, only the users whose permissions have been changed are added to the list.

It is interesting to mention that the cloud customer has an ID as well as the other users and can request reading and writing transactions. In addition, only the cloud costumer can write a new file and, next, define its permissions.

It is important to mention that the attestation of a permission update transaction must be chained with other transactions of a file. Besides the attestations, the customer and the provider send to the TPA the current permissions for auditing. Thus, the auditor verify the correctness of the access control, observing if the users had or not permission in the moment of each transaction. A malicious provider cannot bypass the violation detection, hiding an unauthorized writing, because any writing modifies the file's signature, and an attestation with this signature is sent to users for integrity verification. It is not possible to inform that different users request the transactions, due to the signature of the attestations, and to change the content of an attestation without breaking the chain of attestations.

On the order hand, a malicious provider can hide an illegal reading without being detected if the unauthorized user also does not report this transaction. However, even if this user receives the file, he/she cannot decipher it without the keys. Thus, the data leakage can occur only if the lazy revocation is applied. In this case, a revoked user can use the old key to access the file if it has not been modified. So, due to the impact of not detecting leaks, the lazy revocation approach is not a security recommendation.

In this context, the customer re-encrypts the file when the permissions are updated. Before the encryption, the customer receives the last attestation, the metadata and the file. In addition, the customer creates the new metadata and the new attestation that includes, respectively, the generated keys and the modified permissions. The re-encryption is only required if some user loses the reading privileges. Thus, it is possible to remove the cost to encrypt the file and to transmit the file from and to the provider.

The auditing is enough to detect access control violations. However, the use of an honest broker enables users to be sure that the attestation and the metadata received are the last ones, making possible the access control monitoring. When a violation is detected, then TPA can perform an audit to avoid that a user falsely accuses the cloud provider, solving any conflict. Figure 5 details the protocol to change the permissions, using a broker to manage the transactions. It is worth mentioning that the last step finalizes the transaction so that the three parties have the attestation signed by everyone.

A malicious provider can ignore the permissions, executing illegal transactions. In this context, a revoked user is able to write files that seem valid for other users because the signatures generated with old keys pass on integrity checks. In a practical way, a malicious agent (software) in the provider can commit a writing request by a revoked user, without the broker's participation during the communication. There is no attestation of this writing because all attestation must also be signed by the broker. Thus, this transaction will not be represented in the chain of attestations. If the broker is honest, the illegal writings are detected when the FileHash of the last attestation is different from the received file.

On the other hand, it is possible, in collusion with

UsedID	UsedLSN	FileID	FileVersion	FileHash	KeyVersion	Transaction Type	Permissions List	ChainHash	Signatures

Figure 4: Structure of an attestation.

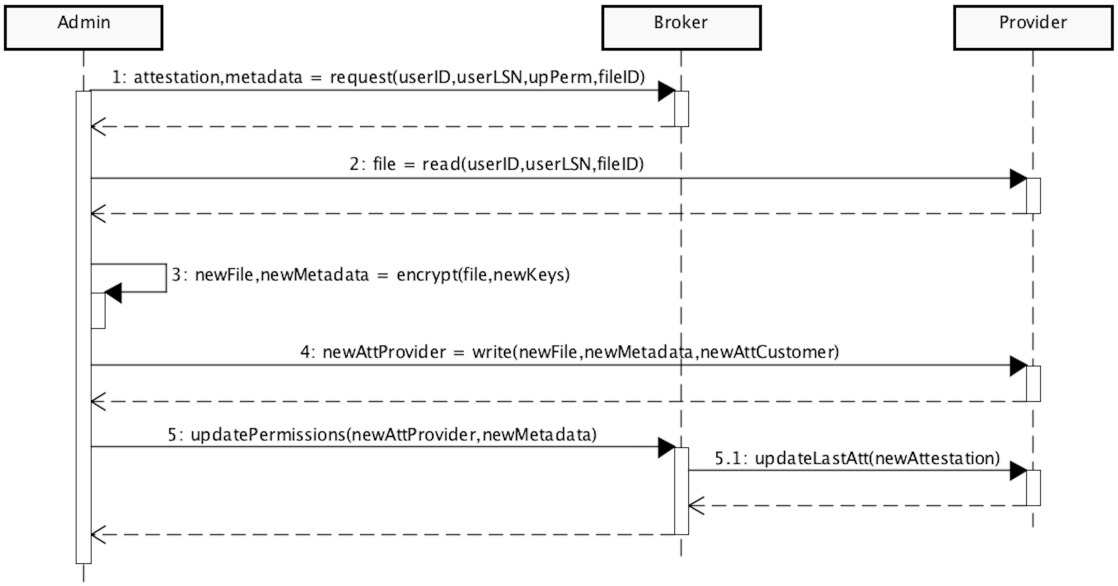

Figure 5: Updating file's permissions.

the broker, to bypass the violation detection. In this case, a valid attestation is created and signed by the attackers (revoked user, provider and broker), performing the writing as defined in writing protocol (see Figure 3). The difference is that the broker also ignores the permissions and accepts this transaction, signing its attestation. The broker also sends an old metadata, indicating the same FileVersion of the file. In next transaction, a user receives the file and the attestation with the same FileHash. Thus, if the broker is compromised, the users may not detect, in the monitored data, a trace of the violations because they do not have the correct information about the current state of the system.

Otherwise, the violation will be identified in auditing. Although the attestation of the illegal writing is rightly chained with the others attestations, the TPA can report the violation, identifying that a user does not have permission in the moment of the transaction. The provider and broker can also try to hide the attestation that indicates the revocation of the user's permission. In this case, the TPA cannot build a valid chain of attestations, or one UserLSN of the cloud customer is absent. Therefore, the auditing indispensable to identify the undetected violations analyzing the historical transactions.

The broker is also responsible for controlling concurrent transactions, allowing access to different files due to the no possibility of violation resulting from such access. On the order hand, simultaneous access to the same file can result in violations if the transactions end in a different order from which they started. In this case, the blockade of simultaneous access ensures the order of execution of the transactions, avoiding, for example, the reading of the previous version of a file and the requests of writings with the same FileVersion. However, concurrent readings do not result in violations and can be allowed.

Due to the signature of the attestations and exchanged messages, any involved entity can detect others attacks (e.g., impersonation and replay attacks) and cancel the unauthorized transaction. However, a malicious behavior of the provider and/or broker results in a security violation. In this case, a penalty and a recovery procedure can be applied as specified in an SLA (Service Level Agreement). For example, if an integrity violation is detected, it is possible to restore the most up-to-date version in a backup, reducing the damage. Last, it is interesting to mention the holding of the users' privacy because the metadata and attestations do not have any sensible information available to malicious entities.

3.2 Modeling and Validation

The evaluation of the security solution is an essential task of its development to demonstrate its security. Due to the difficulty of demonstrating the security only with experiments, the literature highlight the advantages of using formal methods to model solutions and validate their security (Armando et al., 2014). However, there are solutions for secure cloud storage in which the evaluation is limited to an informal discussion about their security and a performance analysis (Popa et al., 2011) (Hwang et al., 2014b) (Jin et al., 2016).

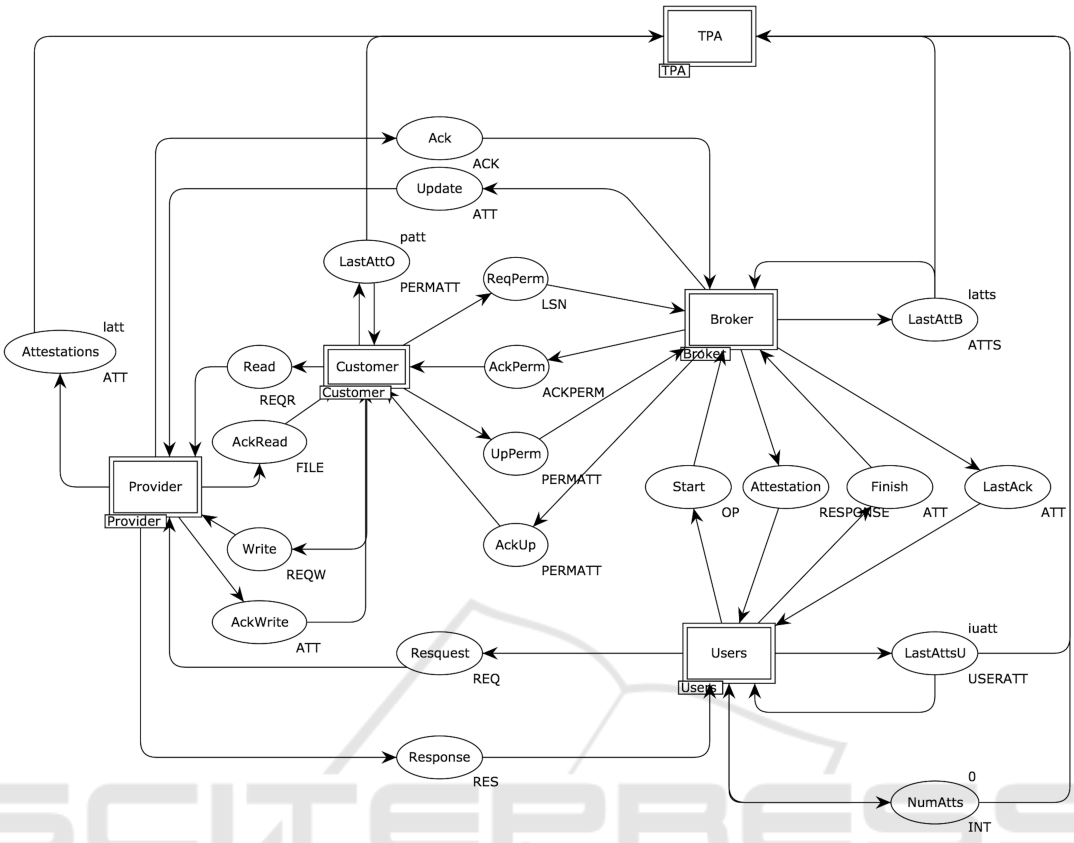

Figure 6: Overview of the simplified CPN.

In this research, CPN Tools[2] is being used to formally model and validate the proposed mechanisms. The choice was based on the ease of use of this tool, of the extensive documentation on CPNs, and its suitability for specifying security protocols (Carvalho et al., 2017a).

The modeling of a system in Petri Nets is done by describing a graph composed of nodes that indicate the places and the transitions of the system. Arcs connect places to transitions or vice versa, but never two places or two transitions. The places in a Petri Net contain tokens, which can enable transitions to be fired. When a transition occurs, tokens from the input places are consumed, and new tokens are added at the output places, according to the arc expressions. The choice for the transition to be fired is nondeterministic. The initial positioning of the tokens in places is called the initial marking, and each marking during the execution of a system represents a state of this system.

The Colored Petri Net is an extension of the Petri Net, which allows you to assign colors or values to the tokens. Similar to programming languages it is possible to define what type of information can be stored in each place and also assign or verify the values of the tokens. Thus, the size of the created models is reduced. There are also other extensions, including hierarchical and timed Petri Nets, which respectively organizes a system into modules and adds timers to the transitions. This topic is detailed in (Jensen and Kristensen, 2009), with details on the types of CPNs and formal definitions.

Here, we present the modeling of the proposed, as well as, the security evaluation, showing the detection of the attacks enumerated in Section 2.2.2. The modeled CPN includes the protocols for auditing and for monitoring cloud transactions (*i.e.*, writing, reading and permissions updating) to analyze the detection of violations.

A simplified CPN of our proposal is available at https://sites.google.com/site/candrebc/scss_ac.zip[3] and the overview of this modeling is shown in Figure 6. The simplifications do not change it functioning and include: i) the storage of a single file; ii) the execution of a single transaction each time; iii) the re-

[2]http://cpntools.org

[3]The original file can be used for proper analysis and simulations. A *.pdf* file is also available.

presentation of the chain hash as a sequence number; and iv) the assumption that all messages are authentic. The modeled CPN is parametrized so that several users can be easily added making our proposal theoretically scalable. These simplifications facilitate the analysis of the modeling due to the reduction of its complexity and of the space state.

This modeling represents the correct behavior of the involved entities and simulations were realized to verify the correctness and to evaluate the modeled mechanisms. Thus, no violation is detected because the analysis of the space state reveals that no token is stored in error places in any possible state. This result is expected when the entities are trustworthy, but it is also necessary to analyze if all possible attack is detected, enabling the use of the proposed mechanisms to demonstrate the security of a storage service.

Thus, it is essential to model the attacks that violate the access control, showing if they are identified. We modeled malicious behaviors of the provider, broker and users in different CPNs, simulating the violation scenarios. The changes in the modeling represent an entity's ability to execute malicious software, making possible to perform attacks that result in violations. Now, it is possible to verify if the attacks performed by malicious entities can be detected. We also define special places, indicating the execution of malicious actions and the violation detection. Thereby, if some attack is triggered, the users or auditor detect it when a violate place is achieved.

Since the unauthorized users cannot obtain the keys, they are unable to read a file or write with a valid signature. Thus, we focused our analysis in verifying if revoked users can access the files in accordance with the threats highlighted in Section 2.2.2. Due to the file's re-encryption when updating the permissions, the revoked users can no longer read the file. So, it was verified if, using lazy revocation, the unauthorized readings are allowed.

In the experiments, revoked users can read a file since this has not been updated. The unauthorized readings were only detected in auditing and if the transactions were registered in the log. Otherwise, the illegal reading attestation is not chained with the others attestations, and the violation detection is bypassed. In order to allow lazy revocation and avoid data leakage, solutions of trusting computing must be used so that accountability tools register all events in logs that are tampered-free and cannot be deleted (Ko et al., 2011). However, the study of these tools is beyond the scope of this work.

In addition, a revoked user can write files using the old keys with the help of the malicious provider. If the broker is honest, this transaction cannot be performed through the broker and is not registered in the log. The experiments demonstrated that a user detects the violation as soon as he/she executes a transaction, comparing the signature of the received file and the one indicated in attestation sent by the broker.

Although this modeling includes the broker, it is not possible to be sure about its honesty, making necessary the auditing to detect collusion attacks. In this case, the broker can collude with the provider and revoked user, generating a valid attestation of an illegal writing. Thus, in the following transactions, the users do not have enough information to detect the attacks. It is interesting to mention that the file's metadata is also sent to users by the broker. So, make this attack possible if the lazy revocation is not used, the broker must send, after the attack, the last attestation and an old metadata. With the lazy revocation approach, it is not necessary to send an old metadata because, with the current metadata, the users can extract the previous key, used by the revoked user.

Experiments with and without lazy revocation were performed, and the collusion attacks were identified in auditing. It is interesting to mention that, even if the broker is malicious it is possible to identify violations in real-time under certain conditions. The users can detect attacks early when comparing the FileVersion of the current transaction with that of the last transaction. The current FileVersion cannot be smaller than the previous one. Therefore, the proposed monitoring and auditing mechanisms of the access control detect these violations.

4 RELATED WORK

The monitoring and auditing will increase the transparency of cloud storage services, enabling the detection and proof of violations. These violations result from attacks that could not be avoided, especially the ones resulting from malicious acts of the stakeholders, including end users. The existing solutions combine, mainly, access control mechanisms with violation detection mechanisms to offer a secure storage solution. Table 2 presents a comparison of related work, focusing on violation detection mechanisms. Our proposal is highlighted in the last line of this table.

These mechanisms can detect malicious behaviors that result in violations of integrity, retrievability, freshness, or write-serializability. While the verification of integrity checks if a received file is corrupt, the others properties are related to data loss. The verification of retrievability identifies, in auditing, the loss of data scarcely accessed. Data freshness indicates the reading of the most updated file, and the write-

Table 2: Comparison of related work.

Work	Violations Detected					Real-time Detection	Collusion Attacks
	I	F	W	R	*AC*		
Popa et al. 2011	Yes	Yes	Partially	No	*No*	I	Unfeasible
Albeshiri et al. 2012	Yes	Yes	Inefficient	Yes	*No*	I	Unfeasible
Hwang et al. 2014b	Yes	Yes	Partially	No	*No*	I	Unfeasible
Hwang et al. 2014a	Yes	Yes	Partially	No	*No*	I, F, W	Not detected
Tiwari and Gangadharan 2015a	Yes	Yes	Inefficient	Yes	*No*	I	Unfeasible
Jin et al. 2016	Yes	Yes	Inefficient	Yes	*No*	I, F, W	Not detected
Carvalho et al. 2017a	Yes	Yes	Yes	Yes	*No*	I, F, W	Yes
Proposed Mechanism	**Yes**	**Yes**	**Yes**	**Yes**	***Yes***	**I, F, W, AC**	**Yes**

I - Integrity; F - Freshness; W - Write-serializability; R - Retrievability; AC - Access Control

serializability controls the writing order (Jin et al., 2016).

The secure storage service, proposed by (Popa et al., 2011), is based on Access Control Lists (ACLs) and audited for integrity, freshness and write-serializability verification. (Jin et al., 2016) add a scheme for verifying the data retrievability. The discard of old logs and management of concurrent transactions are discussed by (Hwang et al., 2014b). (Hwang et al., 2014a) specify a trusted third party, called synchronization server, to enable real-time violation detection.

The detection of write-serializability violations performed by (Albeshri et al., 2012), (Tiwari and Gangadharan, 2015a) and (Jin et al., 2016) is inefficient because the reading is always required before each writing. Besides, (Popa et al., 2011), (Hwang et al., 2014b) and (Hwang et al., 2014a) do not detect all violations, because only the logs are analyzed to check the write-serializability, as discussed in our preliminary study (Carvalho et al., 2016). Thus, we proposed, in (Carvalho et al., 2017a), the use of a Proof of Retrievability (PoR) scheme to check the file stored by the provider without recovery the whole file (Yang and Jia, 2012), improving the write-serializability verification.

The real-time detection is possible when a broker (or other third entity) manages the cloud transactions and informs users the current state of the system. In (Jin et al., 2016), a broker is not specified, and the freshness verification depends on the knowledge of the last root signing key. However, the real-time verification is possible only if the users are sure that the key received is the last one, and the authors do not discuss it. If it is deployed in a private cloud, it is valid to assume that the broker is honest. On the order hand, an untrustworthy broker makes feasible the collusion attacks, in which the broker informs a previous state of the system, deceiving the real-time violation detection. The auditing is indispensable to detect these attacks, as discussed in (Carvalho et al., 2017a).

These mechanisms do not verify the data confidentiality because the protection against data leakage is already provided by the access control mechanism. However, these mechanisms do not detect the attacks described in Section 2.2.2. This limitation is addressed by the proposed mechanism that verifies if the transactions are authorized.

5 CONCLUSION

The literature reveals solutions that address security issues in cloud storage, especially regarding the detection of data corruption, loss and leakage. In general, the solutions combine different security mechanisms to protect the system against various threats. For example, an access control mechanism is necessary to allow data sharing, preventing unauthorized access, while monitoring and auditing mechanisms permit to detect attacks that cannot be avoided. The attacks are identify in real-time if the broker is trustworthy. Otherwise, the auditing detect the violations. Thus, these mechanisms improve the transparency of security storage services, proving the honesty of the provider or its malicious behaviors.

However, existing solutions do not detect access control violations, especially resulted from transactions performed by revoked users. In this paper, we described mechanisms to verify the correctness of the access control. The transactions for permissions updating must also be registered in the read/write transactions log. Thus, it is possible to analyze if only authorized users accessed the files.

The proposed mechanisms detect all violations enumerated in Section 2.2.2. However, a solution of trust computing must also be used to avoid the reading by revoked users if the lazy revocation is applied. An honest broker enables the real-time detection, and the auditing solves conflicts and identifies malicious brokers, detecting collusion attacks. The modeling with CPNs enables a formal validation of the proposed mechanisms, demonstrating that detection of attacks in

accordance with the specified threat model. The proposed mechanisms improves the violation detection and can be used with an SLA solution that specifies security guarantees (Luna et al., 2015).

Although our evaluation with CPNs demonstrate the security of our proposal, it is interesting, as future work, to deploy it in a cloud infrastructure in order to evaluate its functioning in a real scenario and analyze performance aspects. After, it is possible to propose changes to improve the efficiency, without losing the security. The proposed mechanisms can be adapted to be combined with others access control mechanisms (*e.g.*, proxy re-encryption or Attribute-Based Encryption). A robust solution should also include mechanisms to address other security properties such as availability and location. The broker, for example, can manage the storage in multiple providers, improving the service availability.

ACKNOWLEDGEMENTS

This work was partially supported by the STIC-AmSud project SLA4Cloud. Carlos André Batista de Carvalho was also supported by CAPES/FAPEPI Doctoral Scholarship.

REFERENCES

Albeshri, A., Boyd, C., and Nieto, J. G. (2012). A security architecture for cloud storage combining proofs of retrievability and fairness. In *3rd International Conference on Cloud Computing, GRIDS and Virtualization*, pages 30–35.

Amazon (2017a). How amazon simple storage service (amazon s3) uses aws kms. http://docs.aws.amazon.com/kms/latest/developerguide/services-s3.html. Accessed: 2017-07-06.

Amazon (2017b). Managing access permissions to your amazon s3 resources. http://docs.aws.amazon.com/AmazonS3/latest/dev/s3-access-control.html. Accessed: 2017-07-06.

Ardagna, C. A., Asal, R., Damiani, E., and Vu, Q. H. (2015). From security to assurance in the cloud: A survey. *ACM Comput. Surv.*, 48(1).

Armando, A., Carbone, R., and Compagna, L. (2014). Satmc: A sat-based model checker for security-critical systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 31–45. Springer.

Barker, E. (2016). Recommendation for key management, part 1. NIST Special Publication 800-57.

Boneh, D., Gentry, C., and Waters, B. (2005). Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer Berlin Heidelberg.

Carvalho, C. A. B., Agoulmine, N., Castro, M. F., and Andrade, R. M. C. (2017a). How to improve monitoring and auditing security properties in cloud storage? In *35th Brazilian Symposium on Computer Networks and Distributed Systems*, pages 559–572.

Carvalho, C. A. B., Andrade, R. M. C., Castro, M. F., and Agoulmine, N. (2016). Modelagem e deteco de falhas em solues para armazenamento seguro em nuvens usando redes de petri coloridas: Um estudo de caso. In *XIV Workshop de Computao em Clouds e Aplicaes (WCGA/SBRC)*, pages 17–30.

Carvalho, C. A. B., Castro, M. F., and Andrade, R. M. C. (2017b). Secure cloud storage service for detection of security violations. In *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 715–718.

Hwang, G.-H., Huang, W.-S., and Peng, J.-Z. (2014a). Real-time proof of violation for cloud storage. In *CloudCom'14*, pages 394–399.

Hwang, G.-H., Huang, W.-S., Peng, J.-Z., and Lin, Y.-W. (2014b). Fulfilling mutual nonrepudiation for cloud storage. *Concurrency and Computation: Practice and Experience*.

Jensen, K. and Kristensen, L. M. (2009). *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media.

Jiang, W., Wang, Z., Liu, L., and Gao, N. (2014). Towards efficient update of access control policy for cryptographic cloud storage. In *International Conference on Security and Privacy in Communication Systems*, pages 341–356.

Jin, H., Zhou, K., Jiang, H., Lei, D., Wei, R., and Li, C. (2016). Full integrity and freshness for cloud data. *Future Generation Computer Systems*.

Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., and Fu, K. (2003). Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42.

Ko, R. K., Jagadpramana, P., Mowbray, M., Pearson, S., Kirchberg, M., Liang, Q., and Lee, B. S. (2011). Trustcloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 584–588.

Luna, J., Suri, N., Iorga, M., and Karmel, A. (2015). Leveraging the potential of cloud security service-level agreements through standards. *IEEE Cloud Computing Magazine*, 2(3):32 – 40.

Popa, R. A., Lorch, J. R., Molnar, D., Wang, H. J., and Zhuang, L. (2011). Enabling security in cloud storage slas with cloudproof. In *USENIXATC'11*.

Rong, C., Nguyen, S. T., and Jaatun, M. G. (2013). Beyond lightning: a survey on security challenges in cloud computing. *Computers and Electrical Engineering*, 39(1):47–54.

Stallings, W. (2016). *Cryptography and network security: principles and practices*. Pearson, 7 edition.

Tassanaviboon, A. and Gong, G. (2011). Oauth and abe based authorization in semi-trusted cloud computing: aauth. In *Proceedings of the second international workshop on Data intensive computing in the clouds*, pages 41–50.

Thilakanathan, D., Chen, S., Nepal, S., and Calvo, R. A. (2014). Secure data sharing in the cloud. In *Security, Privacy and Trust in Cloud Systems*, pages 45–72. Springer.

Tiwari, D. and Gangadharan, G. (2015a). A novel secure cloud storage architecture combining proof of retrievability and revocation. In *ICACCI'15*, pages 438–445.

Tiwari, D. and Gangadharan, G. (2015b). Secure sharing of data in cloud computing. In *International Symposium on Security in Computing and Communication*, pages 24–35.

Yang, K. and Jia, X. (2012). Data storage auditing service in cloud computing: challenges, methods and opportunities. *World Wide Web*, 15(4):409–428.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS