Towards a Proof-based SLA Management Framework
The SPECS Approach

Miha Stopar[1], Jolanda Modic[1], Dana Petcu[2] and Massimiliano Rak[3]
[1]*XLAB d.o.o., Pot za Brdom 100, Ljubljana, Slovenia*
[2]*Institute e-Austria Timisoara and West University of Timisoara, Timisoara, Romania*
[3]*Department of Information Engineering, Second University of Naples, Aversa, Italy*

Keywords: Cloud Storage, Cloud Security, Security SLA, SLA Management.

Abstract: We present a framework that allows monitoring of the cloud-based applications and environments to verify fulfilment of Service Level Agreements (SLAs), to analyse and remediate detectable security breaches that compromise the validity of SLAs related to storage services. In particular, we describe a system to facilitate identification of the root cause of each violation of integrity, write-serializability and read-freshness properties. Such a system enables executing remediation actions specifically planned for detectable security incidents. The system is activated in an automated way on top of storage services, according to an SLA, which can be negotiated with customers.

1 INTRODUCTION

Most of the Cloud Service Providers (CSPs) today offer their services through a Service Level Agreement (SLA) and the importance of it is undisputed. However, most of the SLAs are relatively simple and usually cover at most availability, support, and disaster recovery. While having these properties defined in SLAs is absolutely crucial, there are numerous incidents which are not covered. Imagine, for example, the scenario where for some reason the Cloud Service Customer's (CSC's) data on the cloud server gets corrupted. In this case, the service is still available and thus no recovery of service is needed, the support is available, too, but most probably the CSP does not have a mechanism to restore the corrupted file. Moreover, it is most likely that the data corruption would be detected, by the CSC, only when trying to access the file for the next time (which might be months after the corruption date).

Thus, there is a need to introduce new properties in SLAs. In this paper, we focus on security properties that are the most critical in the cloud storage SLAs, i.e., *confidentiality* (C), *integrity* (I), *write-serializability* (i.e., consistency among updates; denoted as WS), and *read-freshness* (i.e., assurance that the requested data is always fresh as of the last update; denoted as RF). Apart from providing C, I, WS, and RF guarantees to CSCs, there are also other needs: i) to *continuously monitor* the system in order to assure that the CSP's commitments with respect to these features are fulfilled, and ii) to *automatically react* in case of detected violations in order to guarantee business continuity. Moreover, in order to enable the CSCs to prove a violation of some commitment to the CSP and to enable the CSP to disprove any potential false accusations from the CSC, developing a *proof-based system* is of utmost importance when it comes to assuring and enforcing security in cloud storage environment. We firmly believe that no SLA should be signed without an assurance of the existence of such a proof-based system (as without it the CSC has no mechanism to claim the compensations). As oppose to availability violations, which are usually quickly noticed and reported in news, the violations of integrity could be noticed only by the affected CSC. Thus, the CSC should have an undisputable proof of the SLA violation and should be notified about it immediately when the violation occurs.

The authors of this paper are involved in a FP7-ICT project SPECS (SPECS, 2013) whose objective is to address the issue related to cloud SLAs as far as security is concerned. To provide a solution to all these issues, we adopt the SPECS SLA management framework, enriching it with a proof system for the security properties that the CSP guarantees for

Stopar, M., Modic, J., Petcu, D. and Rak, M.
Towards a Proof-based SLA Management Framework - The SPECS Approach.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 2*, pages 240-248
ISBN: 978-989-758-182-3

storage services. The SLA management framework can be put on top of any cloud storage provider to enhance it with the above mentioned security features. The framework supports continuous monitoring and automatic responses to violations of assured security properties specified in SLAs. In the following we focus on the innovative features related to storage services, while we suggest (Rak et al., 2015) for more details on the SPECS framework.

Some of the above discussed concerns have already been tackled. How to enforce the above mentioned security features with SLAs and how to detect violations related to them has been discussed earlier. For example, Popa et al. (2011) provided a proof-based system (named *CloudProof*) for enforcing and monitoring each of the properties C, I, WS, and RF. We go one step further and provide an extension of the system that is able to distinguish among different types of attacks in case of their violations (CloudProof only detects that some security property is violated and it does not try to determine the root cause). Root cause analysis is important not only because it gives an insight into what is going on in the system, but also because with that kind of additional information CSPs can choose and apply optimal reactive measures to recover from the incident.

This paper gives the following contributions.

Auditor Extension. Our first novelty is the extension of the CloudProof's Auditor in order to facilitate identification of the root cause of each violation of I, WS and RF properties. Such an extension enables us to develop and automatically apply specific remediation actions.

Proof-based SLA Management Framework. In addition to the extended design of the Auditor, we provide a proof-based SLA management framework that monitors the system to verify fulfilment of SLAs, and analyses and remediates all detectable eventualities that compromise validity of SLAs. The proof-based system also assures proofs of violations which are valuable not only to the CSC which can prove CSP's misbehaviour, but also to the CSP which can disprove any false accusation.

The paper is organized as follows. The current state of the art is discussed Section 2. Further details about SLAs and the introduced SLA management framework are presented in Section 3. The initial CloudProof's Auditor is briefly discussed in Section 4 and its proposed extension is presented in Section 5. The proposed techniques for monitoring SLAs and automatically remediating SLA violations related to I, WS, and RF are elaborated in Section 6. Implementation details are discussed in Section 7, and the conclusions are discussed in Section 8.

2 RELATED WORK

An SLA is essential in formalizing a relationship between a CSC and a CSP. It specifies the way both parties share responsibilities and risks that are attached to them. Security breaches and system failures are just a few of the incidents that can occur. In all of these cases the consequences can include contractual termination and loss of customers, financial penalties and lawsuits, severe damage to CSP's business reputation and CSC's loss of sensitive data. Therefore, a number of SLA standardisation initiatives are working on defining a standard format for cloud SLAs. For example, the European Commission has developed standardisation guidelines for cloud SLAs (European Commission, 2014), Cloud Standards Customer Council published a practical guide to understanding cloud SLAs (Cloud Standards Customer Council, 2015), and ISO/IEC JTC1/SC38 standardisation committee is actively working on defining a standard for cloud SLA framework and terminology (ISO/IEC, 2014).

Every CSC should negotiate the desired and required cloud service and its security level in the form of an SLA, and each CSP should continuously monitor the provisioned service to assure the fulfilment of all commitments specified in the SLA. We consider every CSP to be untrusted, thus even something better than just an SLA management framework is needed, i.e., a proof-based system that guarantees transparency of CSP's operations and thus assures CSP's trustworthiness. For example, SLA management frameworks like SLA@SOI (SLA@SOI, 2009) and mOSAIC (mOSAIC, 2010) can detect SLA violations and are even able to recover from them, but no proof-based system is integrated that would provide proofs of violations. Thus nothing assures CSCs that they will be notified about any SLA violation and, most importantly, they will not be able to prove it and claim compensation.

Some solutions exist that cover these issues for some specific security properties. For example, *direct anonymous attestation* scheme (Brickell et al., 2004) is a privacy enhancing scheme that enables assertion of a physical or a virtual component by a trusted source while preserving confidentiality and privacy. The *proof of data possession* (see (Ateniese et al., 2007); (Erway et al., 2015); (Kaaniche et al., 2014)) and *proof of retrievability* (see (Juels et al.,

2007); (Shacham and Waters, 2013); (Bowers et al., 2009)) notions enable the detection of tampering of the stored data. The first scheme allows the CSC to verify the integrity of its data stored in the cloud and the later scheme enables the CSC to verify that the CSP possesses the originally stored data without retrieving it.

Similarly, the concept of *proof of data ownership* (Halevi et al., 2011) has been introduced to alleviate the CSP from storing multiple copies of the same data, and the *transparency logging* scheme (Pulls et al., 2013) has been introduced to enable data processors to inform users about the actual data processing that takes place on their personal data.

Due to various laws and regulations requiring data to be stored and processed in specific geographic location, it is becoming very important to enable the CSC to have a *proof of data location* (see (Katz-Bassett et al., 2006); (Albeshri et al., 2014); (Ateniese et al., 2011); (Watson et al., 2012)).

CloudProof (Popa et al., 2011) addresses the issue of provable violations related to C, I, WS, and RF. The authors designed a cloud storage mechanism that enables detection (with a specific monitoring system, namely the *Auditor*) and proofs of SLA violations related to these properties.

The Auditor in CloudProof is based on *attestations* which are signed messages that accompany each CSC's request and each CSP's response. The CSP stores CSC's attestations for potential cases when the CSC would trigger false accusations. Similarly, the CSC saves all attestations received from the CSP. Additionally, the CSC sends all attestations to the Auditor, which then checks them in order to verify validity of I, WS, and RF commitments. Once attestations are sent to the Auditor, the Client can delete them.

CloudProof attestations enable detection of I, WS, and RF violations. However, no work has yet been done to either identify root causes or to automatize the remediation actions. Note that once a violation is detected either due to a security breach or a system failure, more of them will most likely follow. So it is crucial to first identify the root cause of the event and then restore the system to the normal state accordingly.

The Auditor can be used as a monitoring component of an SLA management framework. However, as opposed to many SLA management frameworks (e.g., SLA@SOI (SLA@SOI, 2009), mOSAIC (mOSAIC, 2010), SPECS (SPECS, 2013)) that have monitoring component tightly integrated with the rest of the components of the framework that is operated by the CSP, we claim that the monitoring and auditing components need to be independent and not operated by the service provider.

To the best of our knowledge, none of the existing SLA management frameworks deal with automatic remediation of provable SLA violations. The majority of proposed solutions either focus on automatic deployment of cloud services (e.g., (Bonvin et al., 2011); (Addis et al., 2010); (Badidi, 2013); (SLA@SOI, 2009)), SLA monitoring (Sahai et al., 2002), or prediction and detection of SLA violations (like (Emeakaroha et al., 2012); (Leitner et al., 2010)). Some SLA management solutions are focused on detection and remediation of performance related SLA violations (see (SLA@SOI, 2009), (Brandic et al., 2010)). However, a framework that would automatically negotiate, enforce, and monitor security SLAs, and remediate detectable security incidents through a proof-based system does not yet exists.

To this end, the rest of the paper presents a new approach to solving the above discussed issues that was also adopted in SPECS (see (SPECS, 2013) and (Rak et al. 2015)). We present an enhancement of the CloudProof scheme and its integration into a new proof-based SLA management framework that not only detects and analyses security incidents and system failures, but also reacts to them in an automated way.

3 SLA MANAGEMENT FRAMEWORK

An SLA specifies all aspects of the service being provisioned by the CSP to the CSC. The agreement details not only the infrastructure and resources to be provisioned, but also the level of security to be assured for the acquired service, along with remedies for the failure to meet those levels. All these aspects are formalised with *Service Level Objectives* (SLOs) that represent CSP's commitments for a specific security property (i.e., for a specific *security metric*, e.g., WS or RF).

Each SLA management framework and each CSP that offers SLAs are in need of the followings items: i) a system to negotiate CSP's services and their security properties in terms of SLOs; ii) a system that automatically deploys negotiated services in a secure manner; iii) a system that monitors negotiated commitments; and iv) a system that manages detected incidents or system failures that compromise validity of negotiated SLOs.

The SLA management framework that we propose aims at providing monitoring and remediation functionalities in the secure cloud storage domain. We assume that negotiation and deployment activities are managed by the CSP and are out of the scope of this paper.

The framework comprises eight components and involves three entities, as depicted in Figure 1.

The *Client* component handles all uploads and downloads of data and provides client-side encryption (enforcing confidentiality). It operates directly on the CSC's side independently from the CSP.

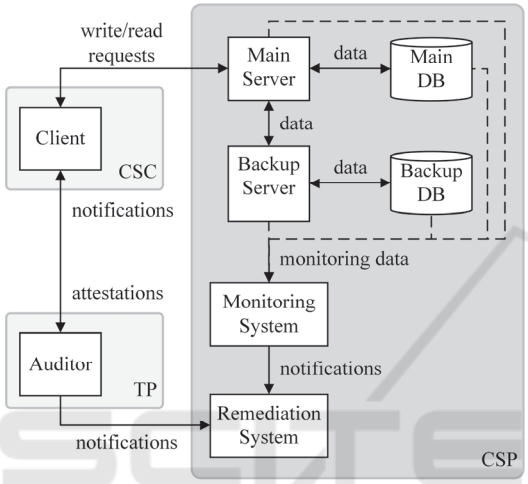

Figure 1: Proof-based SLA management framework.

The *Main Server* orchestrates all upload and download requests from the Client and handles all associated operations (i.e., writes and reads to the *Main DB*, performs backups). Similarly, the *Backup Server* and the *Backup DB* components manage and store, respectively, a copy of all CSC's data that can be retrieved in case of incidents or failures.

The *Monitoring System* component monitors availability of both servers to guarantee business continuity and thus the fulfilment of all undertaken SLAs. In case of any SLA violation, either notified by the Monitoring System or the Auditor, the *Remediation System* component manages entire remediation process. These components reside on CSP's infrastructure.

Note that in the proposed architecture, in order to increase security, both servers should be separated from database components; they should all reside on different VMs. In order to ensure the disaster-proof system, the CSCs' data, the Backup Server and Backup DB should be physically separated from the main components. Additionally, all four components should be separated from the Monitoring and Remediation Systems.

The functionalities of the Monitoring System and Remediation System are further elaborated in Section 6.

The last entity involved in secure storage chain is a trusted *Third Party* (TP) that hosts the Auditor. I, WS, and RF are continuously evaluated by the Auditor. If any violations are detected, both CSP and CSC are notified. It is up to the CSP's *Remediation System* component to identify and apply the optimal corrective measures. Note that violation of I is evaluated and confirmed by the Auditor, but only the Client can detect it.

Some might argue that a CSP cannot be trusted and that there is no guarantee for the CSC that the CSP will handle SLA violations in the CSC's best interest. But since the Auditor is and independent entity, there is no way for the CSP to not react to detected violations and hide it. If I, WS, and RF violations are not handled, the Auditor will keep detecting violations and keep notifying the CSC which might result in a termination of an SLA.

When the CSC signs an SLA with C, I, WS and/or RF guarantees, the CSC is provided with the Client component and an URL of the Auditor.

Confidentiality is enforced with the client-side encryption. But since the Client component which orchestrates encryption resides on the CSC's infrastructure, the CSP has no way of monitoring the code and ensuring its correctness. Thus confidentiality is enforced by the CSP in terms of providing the CSC with the right Client code when the SLA is signed, but it is up to the CSC to maintain the code and assure its validity.

Details about how the Auditor detects I, WS, and RF violations are presented in Section 4.

4 AUDITOR IN CLOUDPROOF

As mentioned in Section 2, the core objects of the auditing process are attestations. Each time the CSC wants to upload a file to the cloud, the Client performs a `put` request which contains the CSC's data to be stored and a *client put attestation*. The CSP (i.e., the Main Server component) stores the data in the Main DB and returns the *cloud put attestation*. The Client has to provide the client put attestation in order to authorize the overwriting of a certain existing data with a new content. The CSP must respond to the request with the cloud put attestation which affirms that CSP received the data unchanged and successfully stored it.

Similarly, every time the CSC wants to download data from the cloud, the Client performs a `get`

request which contains the block ID of the desired data. The CSP (i.e., the Main Server component) returns the requested data along with the *cloud get attestation*. With this attestation the CSP certifies that the returned data is the right one.

The Client automatically sends all cloud attestations to the Auditor. After each *epoch* (i.e., a predefined fixed period of time) the Auditor checks the chain of attestations for the current epoch. The chain of attestations is correct if for each two consecutive attestations A1 and A2 the chain hash in A2 is equal to the hash of A2's data and A1's hash.

For more details about the initial CloudProof system see (Popa et al., 2011). Details about the extension of the initial idea are discussed in the next sections.

5 AUDITOR'S EXTENSION

As mentioned Section 4, the CloudProof's Auditor is able to detect when CSC's commits have not been handled consistently by the CSP (violation of WS or modified illegitimately for some other reason; however, it does not distinguish between the two) and when the CSC has not received data when executing `put` request (violation of RF).

The CloudProof's Auditor can detect violations of I, but only after each epoch, when it checks all attestations (some `get` request attestation would not have the same data as the `put` request attestation for this data and this version). However, the Client can detect such a violation immediately and can trigger the auditing process right away to verify it (sends a notification to the Auditor). The Client can detect violation of I because it can calculate the hash of the encrypted data and compare it with the value in the attestation – also, authenticated encryption should be used which means that decryption handled by the Client would report an error when the data is changed illegitimately.

An additional limit of the existing CloudProof's Auditor is that it is unable to determine what the root cause of violations of WS and RF might be. It cannot determine whether a violation is an attack or a system error. But distinguishing between root causes of violations is crucial because entirely different incident responses are required for a system error and an actual attack. While detection of an attacker requires significant changes like restoring the service on another virtual machine, the detection of a database error might require only switching from a primary database to the backup.

It has to be noted that CloudProof's Auditor does not take into account violations of I detected by CSCs. The Auditor can by itself detect violation of I, but not in real-time. The Client can detect in real-time with a `get` request that a block has been illegitimately changed and send a notification to the Auditor immediately. When the Client sends the `get` request to the server, the cloud get attestation is returned and contains chain hash and block hash of the requested data. With received chain hash the Client can calculate the block hash itself and compare it with the received block hash. If they do not match, the Client detects a violation of I.

Moreover, not only integrity violations can be detected by the Client, a chain hash incorrectness can be discovered as well by checking whether the chain hash returned by the server in the `put` request is correct. The Client has a chain hash from the last `get` request for a block and can calculate the chain hash that is to be returned by the `put` request for the block (the chain hash that is to be contained in the returned cloud put attestation) using the hash of a block and other block metadata. If the Client's calculated chain hash and the one returned from the Main Server are not equal, the Client detected hash incorrectness. Chain hash incorrectness means a WS violation. It can be detected by the CloudProof's Auditor only at the end of an epoch.

Since all this information detected by the Client is crucial for sustaining the security level specified in an SLA, with our approach all integrity violations and any detected chain hash incorrectness are immediately notified to the Auditor for further analysis.

In the following we describe an algorithm executed by the Auditor at the end of each epoch and when (if) the Client detects integrity violations or a chain hash incorrectness. The auditing process is also depicted in Figure 2 where each end node outlines metrics that have been affected by the detected violation (in bold), and remediation actions to be taken to recover from it.

When the auditing process starts (either after an epoch or after a notification from the Client), the Auditor first checks if the chain of attestations (CA) is correct. If the chain of attestations is correct, the Auditor has to consider possible system failures notified by the Client. If the auditing process has not been triggered by the Client, then there is no violation of any SLA and the monitoring process can continue. If the auditing process has been Client triggered due to a detected integrity violation, then the result is a violation of I and WS. In this case, a new Main Server and a new Main DB should be set up.

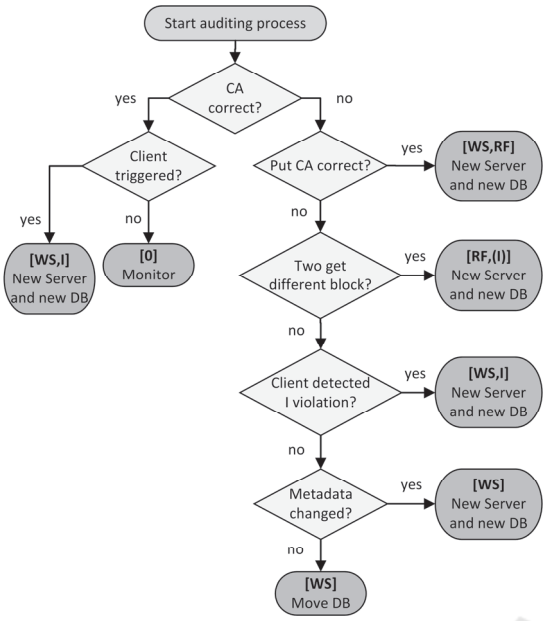

Figure 2: Auditing process.

If the chain of attestations is not correct, the Auditor has to check for the occurrence of the fork attack, where the cloud maintains two copies of the data and conducting some writes and reads on the original data and others on its copy. In order to confirm or rule out the fork attack, the Auditor checks whether each `put` request has a correct chain of attestations behind it.

If all put attestations have a correct chain behind them, then at some point the fork attack occurred which is a violation of WS. This is also a violation of RF since the CSC did not get the latest changes made by some other CSC. The Auditor can determine with which request the data has been forked. This information can later be used to set the system back to the time before the fork attack took place. In this case, a new Main Server and a new Main DB should be set up and the data should be restored from the backup to the version before the attack.

In the case where some `put` request does not have a correct chain of attestations behind it, the Auditor has to check whether there exist two `get` requests for the same block that received the same version number but different block content.

If such a pair of `get` requests exists, an attack might have happened. Two different CSCs received different block data accompanied by the same version and block number (violation of RF). This is with high certainty due to a deliberate attack thus the Main Server and the Main DB should be replaced. Note that also I might be violated in both of these `get` requests. If I is not violated in neither, it means that either keys have been stolen or some old version of the block (with old hash) has been returned – this has to be checked and a special warning has to be sent to the Client if keys are stolen. If I is violated in either of the two `get` requests, this might be due to a system error (e.g., failure of a disk where database resides between the two requests). Regardless, it is better to go with a stricter remediation action (replacing Main server and Main DB).

When such a pair of `get` requests (for the same block that received the same version but different block content) does not exist, this means that some `put` request was not executed correctly and this represents a violation of the WS. In this case, the further process depends on whether the Client detected integrity violation.

If the Client triggered the auditing process after detecting a violation of integrity (which the Auditor now confirms), both the Main Server and the Main DB have to be replaced. On the contrary, if the Client did not detect any issues, the Auditor checks whether the block meta-data (e.g., version number, block number) has been changed in a way that indicates a deliberate attack.

If some element of the block metadata has been changed, e.g., the returned chain hash contains the previous version number and previous block hash, this might indicate a rollback attack. In this case, a new Main Server and Main DB have to be set up. On the contrary, if no metadata has been changed, then there are no indications for an attack. And since the assumption is that all issues are due to a system error, the Main Server should be switched to the Backup DB, which would then take the role of the Main DB, and a new DB should be set up which would take the role of the Backup DB.

Whenever the Auditor detects or confirms a violation of any of the properties I, WS, and RF, both CSP and affected CSCs are notified about the violated property and the required remediation action. It is up to the CSP's Remediation System component to execute remediation actions and it is up to the affected CSCs to claim compensations for the violation of the SLA. Of course, not all detected violations affect all CSCs. For example, a violation of WS only affects CSCs that have this property guaranteed in their SLA.

In the next section we focus on the remediation.

6 SLA REMEDIATION

As described in Section 3, each SLA specifies CSC's security requirements in the form of SLOs that are

built on top of security metrics that the CSP can enforce and monitor. In this paper our focus is on secure storage metrics C, I, WS, and RF.

In the SLA negotiation phase, CSCs have the opportunity to choose which of these properties should be enforced and monitored by the CSP. Once the SLA is signed, and all components are deployed and configured, the CSC can start using the acquired service. In order to fulfil all commitments in the undertaken SLAs, the CSP not only considers and manages notifications from the Auditor, but also uses its own Monitoring System component, which oversees availability of servers and databases.

In case when the Monitoring System detects that one of the servers or databases is unresponsive or unavailable, it has to react since this may not only cause delays in the service but also errors that can affect I, WS, or RF.

Monitoring system continuously checks responsiveness of both servers and databases. If at any moment any of them is unresponsive, the occurrence is notified to the Remediation System component which first tries to restart it. If that solves the issue, monitoring continues. If restarting the unresponsive component does not help, the Remediation System tries to deploy another instance of the unresponsive component. When there is a need to deploy a new database, the Remediation System also triggers backup or restoration of data. If any of these steps fail to recover the system to a normal state, this may threaten the success of future `put` and `get` requests (i.e., validity of SLOs related to I, WS, and RF metrics), thus the CSCs should be notified about the event.

CSP's Remediation System component has to manage notifications of incidents and failures that are sent not only from the Monitoring System, but also the ones sent from the Auditor. As seen previous section, the Auditor not only detects violation of secure storage metrics, but also performs root cause analysis and identifies the proper remediation action. When a notification of a violation is sent to the Remediation System, the Auditor reports about which metrics are violated (so that the CSP can determine the damage with respect to the affected SLAs), what the remediation plan is (to execute it), and which version of the data is the last correct one (to restore the data to the right version).

Remediation actions considered by the Auditor consist of either switching the Main DB to the Backup DB and setting up a new DB to take the role of the new Backup (case 1), or setting up a new pair of Main Server and Main DB components and restoring the data to a certain state (case 2).

In the first case, the Main Server is connected to the Backup DB which takes the role of the new Main DB. A new database is set up which takes the role of the Backup DB. Backup of the entire database is executed immediately.

When there is a need to set up a new Main Server and a new Main DB, all data also have to be restored from the backup DB to a certain version as suggested by the Auditor.

7 IMPLEMENTATION

As seen from the remediation plans discussed above, all activities orchestrated by the Remediation System component (and also those managed by the Monitoring System component) can be easily automatized with one of the existing management and orchestration tools like Chef (Chef, 2008).

In SPECS, remediation process is handled by two components, namely Remediation Decision System which identifies remediation actions needed to recover from SLA violations and Implementation component integrated with Chef which executes remediation plans. Code and further details are provided at (SPECS Team, 2015)).

Other components of the framework described in Section 3 have also been implemented and are available on Bitbucket (SPECS Team, 2015).

8 CONCLUSIONS

The main concerns in today's cloud environment for CSCs and CSPs are security and trustworthiness, respectively. To this end, we have presented a solution that assures security to CSCs in a transparent way and consequently increases trust in cloud providers. We have introduced an SLA management framework that supports a proof-system for security properties particularly related to cloud storage providers, namely confidentiality, integrity, write-serializability, and read-freshness.

The proposed framework is based on the existing CloudProof solution which is able to detect violations of the above mentioned security properties, but has been extended to enable root cause analysis and remediation of detected violations.

In our future work, we aim to extend our root cause analysis approach to include more information. Currently, the root cause analysis is conducted on the basis of the information provided for one single event, whereas in our future research

we intend to include historical monitoring and remediation data to perform broader threat analysis.

ACKNOWLEDGEMENTS

The research was partially supported by the grant FP7-ICT-2013-11-610795 (SPECS).

REFERENCES

Addis, B., Ardagna, D., Panicucci, B., Zhang, L., 2010. Autonomic management of cloud service centers with availability guarantees. In *CLOUD'10, Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, IEEE.

Albeshri, A., Boyd, C., Gonzáles Nieto, J., 2014. Enhanced GeoProof: Improved geographic assurance for data in the cloud. *International Journal of Information Security* 13(2):191-198.

Arcieri T., 2013. What's wrong with in-browser cryptography. http://tonyarcieri.com/whats-wrong-with-webcrypto.

Ateniese, G., Burns, R., Curtmola, R., Herring, J., 2007. Provable data possession at untrusted stores. In *CCS'07, Proceedings of the 14th ACM Conference on Computer and Communications Security*, ACM.

Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., Song, D., 2011. Remote data checking using provable data possessions. *Transactions in Information and System Security* 14(1):1-34, ACM.

Badidi, E., 2013. A cloud service broker for SLA-based SaaS provisioning. In *Proceedings of the 2013 International Conference on Information Society*, IEEE.

Bonvin, N., Papaioannou, T. G., Aberer, K., 2011. Autonomic SLA-driven provisioning for cloud applications. In *CCGRID'11, Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE.

Bowers, K. D., Juels, A., Oprea, A., 2009. Proofs of Retrievability: Theory and implementation. In *CCSW'09, Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ACM.

Brandic, I., Emeakaroha, V. C., Maurer, M., Dustard, S., Acs, S., Kertesz, A., Kecskemeti, G., 2010. LAYSI: A layered approach for SLA-violation propagation in self-manageable cloud infrastructures. In *COMPSACW'10, Proceedings of the 2010 IEEE 34th Annual Computer Software and Applications Conference Workshop*, IEEE.

Brickell, E. F., Camenisch, J., Chen, L., 2004. Direct anonymous attestation. In *CCS'04, Proceedings of the 11th ACM Conference on Computer and Communications Security*, ACM.

Chef, 2008. Chef Software web site. https://www.chef.io/.

Cloud Standards Customer Council, 2015. Practical guide to cloud service agreement version 2.0.

Emeakaroha, V. C., Netto, M. A. S., Calheiros, R. N., Brandic, I., Buyya, R., De Rose, C. A. F., 2012. Towards autonomic detection of SLA violations in cloud infrastructure. *Future Generation Computer Systems* 28(7):1017-1029.

Erway, C. C., Küpcü, A., Papamanthou, C., Tamassia, R., 2015. Dynamic provable data possession. *Transactions on Information and System Security*, 17(4):1-29, ACM.

European Commission, 2014. Cloud service level agreement standardisation guidelines, C-SIG SLA 2014.

Feng, J., Chen, Y., Summerville, D., Ku, W. S., Su, Z., 2011. Enhancing cloud storage security against roll-back attacks with a new fair multiparty non-repudiation protocol. In *CCNC'11, Proceedings of the IEEE Consumer Communications and Networking Conference*, IEEE.

Halevi, S., Harnik, D., Pinkas, B., Shulman-Peleg, A., 2011. Proofs of ownership in remote storage systems. In *CCS'11, Proceedings of the 18th ACM Conference on Computer and Communications Security*, ACM.

ISO/IEC, 2014. Information technology -- Cloud computing -- Service level agreement (SLA) framework and technology (Draft), ISO/IEC 19086.

Juels, A., Kaliski Jr., B. S., 2007. PORs: Proofs of retrievability for large files. In *CCS'07, Proceedings of the 14th ACM Conference on Computer and Communications Security*, ACM.

Kaaniche, N., El Moustaine, E., Laurent, M., 2014. A Novel zero-knowledge scheme for proof of data possession in cloud storage applications. In *CCGrid'14, Proceedings of 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE/AMC.

Katz-Bassett, E., John, P. J., Ksishnamurthy, A., Wetherall, D., Anderson, T., Chawathe, Y., 2006. Towards IP geoolocation using delay and topology measurements. In *IMC'06, Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ACM.

Leitner, P., Michlmayr, A., Rosenberg, F., Dustard, S., 2010. Monitoring, prediction and pevention of SLA violations in composite services. In *ICWS'10, Proceedings of the 2010 IEEE International Conference on Web Services*, IEEE.

mOSAIC, 2010. The mOSAIC project web site. http://www.mosaic-cloud.eu.

Popa, R. A., Lorch, J. R., Molnar, D., Wang, H. J., Zhuang, L., 2011. Enabling security in cloud storage SLAs with CloudProof. In *USENIX ATC'11, 2011 USENIX Annual Technical Conference*.

Pulls, T., Peeters, R., Wouters, K., 2013. Distributed Privacy-Preserving Transparency Logging. In *WPES'13, Proceedings of the 12th ACM workshop on privacy in the electronic society*, ACM.

Rak, M., Villano, U., Casola, V., De Benedictis, A., 2015. SLA-based secure Cloud Application Development:

the SPECS Framework, In *Proceedings of the 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*.

Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A. P. A., Casati, F., 2002. Automated SLA monitoring for web services. In *DSOM'02, Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications*, Springer-Verlag.

Shacham, H., Waters, B., 2013. Compact Proofs of Retrievability. Journal of Cryptology, 26(3):42-483.

SLA@SOI, 2009. The SLA@SOI project web site. http://sla-at-soi.eu.

SPECS, 2013 The SPECS project web site. http://www.specs-project.eu/.

SPECS Team, 2015. SPECS Team Bitbucket account. https://bitbucket.org/specs-team/.

Watson, G. J., Safavi-Naini, R., Locasto, M. E., Narayan, S., 2012. LoSt: Location based storage. In *CCSW'12, Proceedings of the 2012 ACM Workshop on Cloud Computing and Security Workshop*, ACM.