Investigations

A Proposal for Standardized Data Attribution and Ownership in the Cloud Environment

Talal Talib Jameel

Department of Dentistry, Al Yarmouk University College, Baghdad, Iraq

Article history
Received: 27-10-2016
Revised: 09-12-2016
Accepted: 26-12-2016

Email: talal.alhabeeb@gmail.com

Abstract: Since the earliest days of cloud computing there has been a steady migration of data from local data stores to the cloud. As more and more cloud platforms become available, this inflow of data has only increased dramatically. By some estimates, "the cloud" will hold 50% of all data by 2020. Localized management of data is typically handled using tools, policies and access control methods that are appropriate to the local environment in question. Once data has been migrated to the cloud, these local tools and policies are rarely applicable and a new approach must be taken. Other works have focused on the problems associated with cloud security. This paper try to determine attribution and ownership for data in the cloud. Just because data is stored in account X that does not necessarily mean that X owns all the data in that account. An approach based on the use of a Public Key Infrastructure (PKI) is addressed to provide cryptographically strong data attribution and attestation for data in the cloud.

Keywords: Cloud Computing, PKI, Certificate Authority

Introduction

PKIs have been deployed for a variety of difference forms of distributed data management. The cloud PKI (CPKI) proposed in this study allows cloud users to manage their data as a resource, or as a set of resources. In this CPKI the resources managed are blocks of data as defined by the user. The semantics of the data under management is completely separated from the syntax of the data under management (Aye *et al*., 2013). Thus, an individual user can declare that any type or collection of data is a "resource" to be managed. This allows for very fined grained control of data attribution and ownership with the CPKI. Attribution and ownership are asserted through a special type of cryptographic certificate, namely a customized ×509v3 certificate. In order use a certificate-based PKI one must also deal with the infrastructure associated with such a PKI, namely Certificate Revocation Lists (CRLs) (Kent *et al*., 2000), trust anchors (Tas), publication points and so forth. Given the large amount of data that is already in the cloud, there are very different implementation challenge from a typical PKI, in that in the CPKI every data user needs to validate every certificate and CRL at time of use. This level of granularity, when imposed over a vast array of data objects makes validation performance in the CPKI a very high priority. In a typically PKI one can often rely on a transaction rate on the order of hours, but in the CPKI the transaction rate may be seconds or less. This paper describes software under development that can be used to perform data attestation and ownership for data objects stored in the cloud, in an efficient manner, with a special focus on the means and methods used to realize a high performance design. Theoretical discussions are augmented with actual performance data.

Background

An ×509 certificate is a digital certificate that is used by a PKI to assert that a digital object possesses certain properties. Such certificates are issued by a Certificate Authority (CA), which is considered to be the parent of the certificate (Housley *et al*., 1999). In the PKI model the chain of certificates that form the parent, grandparent, etc. of a given certificate must terminate in a top level certificate, known as a Trust Anchor (TA) (Montana and Reynolds, 2008). A TA must be a self-signed certificate issued by a well-known trusted authority, such as one of the five top-level Regional Internet Registries (RIRs). An ×509 certificate may be given the authority to issue subordinate certificates (so that it known as a CA certificate), or it may be a leaf node in the tree of certificates (known as an End Entity (EE) certificate) (Cittadini *et al*., 2010). An ×509 certificate is bound to a distinguished name, an issue, a validity period and, in our proposed implementation (Jensen *et al*., 2009; Muñoz *et al*., 2004), with issuer and subject unique identifiers. There have been three versions of the ×509 standard; we will restrict ourselves to only using v3, since that is the only version that permits user-defined extensions. An extension is expressed by an Object Identifier (OID) which is merely a set of values with semantics that may be interpreted by the user in a context-specific manner (Oppliger, 2001).

Certificate contents, in particular, the contents of specific extensions, is expressed using a data definition language known as abstract syntax notation 1 (ASN.1) (Rose and McCloghrie, 1990; Huff *et al*., 1998). There are a small set of predefined OIDs, as expressed in the IETF RFC5280, but none of these will overlap with the OIDs that will be used for the CPKI.

Many solutions (Fujishiro *et al*., 2010; Kent, 2006) have been put forth to strengthen cloud security and to provide stronger forms of attribution and ownership information than is currently available. All of these solutions have been predicated upon the existence of some form of PKI that binds data resources to the entities to which they have been allocated, e.g., the owner of the account in which the data has been stored. These solutions have typically not been adopted due to the changes required cloud account management and the associated infrastructure requirements that would thereby be imposed (Nasreldin *et al*., 2015; Liu *et al*., 2015). We propose a method for creating the requisite infrastructure without any changes required by cloud vendors. This approach is based new, digitally signed object, the Data Attribution Object (DAO), together with a PKI to validate, manage and process such objects. Associated with the DAO is another, already existing, digitally signed object, the Manifest. The manifest has been created to help protect the contents of CPKI object repositories. Manifest processing (in the context of other PKIs) has proven to be significantly more complicated that its straightforward nature would imply (Ghazi *et al*., 2016; Zhao *et al*., 2012). Thus, performance optimization therefore continues to be a critical requirement of the proposed software architecture. This paper describes the function of DAOs and manifests within the CPKI. It then discusses the challenges associated with efficiently processing very large collections of such objects together with their associated certificates and certificate revocation lists. It then describes in detail the key components of the implementation that will provide for high performance, scalable processing. In a typical PKI the validation problem for each data user is fairly simple in concept, although it may be complex in practice. Typically, a relying party receives an End Entity (EE) certificate that must be validated prior to verifying the signature on an object (in this case, a data object). The data user may be provided with additional Certification Authority (CA) certificates needed to complete the certificate path to one or more Trust Anchors (TA) recognized by that relying party.

The validation of a certification path from a TA to an EE certificate, including processing of revocation status data contained in one or more CRLs (Housley *et al*., 2002), is well defined and specified in IETF standards. The non-standard part of the process is the discovery of a suitable certification path. Given this typical task for a user, strategies for optimizing the performance of certificate validation within a typical PKI are under development. They are based on the assumption that a user will, within a reasonable time interval (say, 24 h), validate only a very small fraction of all the certificates issued in the context of the traditional PKI. This is a reasonable assumption for most applications, which will only user a portion of the data under management. Presents a very different challenge for relying parties with regard to validation. In this CPKI it is anticipated that every user may need to validate a large number of certificates within (roughly) a 24 h interval. This dramatic difference in the scope of validation motivated the development of a novel performance-optimized approach.

The recent addition of manifests, which are designed to authoritatively assert the contents of a repository of CPKI objects using a signed list of file hashes, adds an additional layer of complexity and makes it even more imperative that CPKI processing be highly optimized. This paper focuses on the software design and implementation choices associated with providing high performance processing that have arisen as part of the proposed implementation of the CPKI. Our approach uses several strategies for providing high performance; chief among these is the use of a relational database to cache information about all digital objects in the CPKI in order to minimize costly disk accesses and also to almost completely eliminate duplicate operations on such objects.

DAOs and Manifests

A DAO is a digitally signed object asserting that the organizational entity associated with the OID named in the DOA is the authorized owner of the data associated with that OID. Block (s) contained in the DAO. As such, a DAO is an integral component of the overall strategy toward securing data in the cloud. A DAO is a binary ASN.1 encoded data structure consisting of an envelope and a body. The envelope is specified using the Cryptographic Message Syntax (CMS) and, generally speaking, contains metadata regarding the DAO. In particular, we propose that the DAO's CMS envelope contains the EE certificate, the public key of which is used to verify the signature on the DAO. The body of the DAO contains the binding association between OIDs and data blocks. The goal of the CPKI software under development is to determine which DOAs are valid, using a set of rules to be discussed shortly, in order to produce output that can be used to assist with access control decisions. DOAs enable a user to verify that the being used as input to some cloud application is authentic, without needing to refer to coarser grained attributes, such as the account owner of the cloud resources where the data is stored. An attacker can attempt to assert ownership of data by forging account credential, for example, but will not be able to break the digital signature on the DAO and thus will not be able to create false attributions or ownership claims. Given just the DAO file itself, a significant number of validity checks can be performed.

The syntactic structure of the CMS envelope and DAO body can be checked against their ASN.1 definitions; the field values in the envelope and body can be checked against the specifications, the EE certificate can be checked against its syntactic definition and specifications and, finally, the DAO's signature can be checked. If any of these checks fail we say that the DAO is locally invalid, while if all pass then we might say that the DAO is locally valid. Local validity is a strict subset of validity, since a path must still be discovered from the DAO's EE cert to a trust anchor in order for it to become (globally) valid and thus be used to allow access. Observe that a locally valid DAO may become (globally) valid via this process; it may also become locally invalid, if, for example, its EE cert is revoked. Observe carefully, however, that a locally invalid DAO can never be rehabilitated: It will always be locally invalid and therefore need not enter into any further processing. This seemingly trivial observation has proven to be a critical component of the proposed implementation of validation processing for all digital objects in the CPKI system, not just DAOs. A data repository is a hierarchical collection of files rooted at a distinguished directory.

Within the context of the CPKI, a data repository is rooted at a publication point for the digital objects that constitute the CPKI. A manifest is a digitally signed object that makes a positive assertion about the contents of a repository. Specifically, a manifest lists all files at a repository publication point (other than itself) and provides a hash for each file on the list (A manifest cannot list itself since there would be no way to correctly compute its own hash). A manifest actually makes two assertions. For each file in the manifest, it asserts that said file should be present in the repository and should have the specified hash. It also asserts that if a file is not listed on the manifest, it should not be present at the publication point. The goal of having a manifest is to enable users to detect tampering with repository contents, thus removing the need for repositories to be absolutely trusted. Tampering is detected when a local copy of a repository is brought into synchronization with a remote copy of that same repository.

Note that a manifest is an absolute list of contents, not a relative or incremental list of contents; it asserts membership (and hashes) or non-membership of all files within the repository tree at a given instant of time. A manifest is also a signed object. Like a DAO, a manifest must be verified using an End Entity certificate. This EE certificate is included within a CMS envelope that wraps the manifest. At first glance it might seem that a significant amount of local validity processing could be performed using a manifest as a means for culling files with bad hashes.

Regrettably, a more detailed look below the surface reveals that this naïve strategy would leave the system open to a different type of adversarial influence, namely a denial of service attack. Specifically, since a manifest is a signed object, in order for it to be valid it must be both locally and globally valid and therefore it must have a chain to a trust anchor. If a manifest were only locally validated and that manifest were used to eliminate objects with bad hashes, an adversary could construct a locally valid manifest and provide deliberately bad hashes for those files which he wished to deny to the user. Thus, a manifest must be (globally) valid before it can even enter into local validation of objects that it names. There are actually many more points in the state space for manifest processing than this brief analysis indicates. Manifest processing is complicated and therefore must be carefully implemented in order not to have an unduly adverse effect on performance.

CPKI Software: Design Considerations

In order to create a high-performance implementation for the CPKI software, while fully encompassing the needs of the system's end users, several optimizations must be performed (Fig. 1). Two types of optimization were used: One based on a particular type of functional partitioning of the software and a second based on a particular choice of implementation strategy. The first optimization is to segment the system into components with orthogonal functionality, so that those components could be distributed across the operational timeline of the user. Within the nominal twenty four hour processing interval, several operations need to be performed. The local repository needs to be synchronized with the remote repositories; new and modified objects need to be processed; and the side effects of deleted objects must be handled. The presence of new or modified objects provides the opportunity for new validation paths to be discovered and thus for objects to move from an incompletely validated state into a globally validated state.

A new or modified object may also be a CRL, however, which can invalidate objects and therefore can disassociate previously formed paths, causing objects to move from a valid state to an incomplete state. Finally, time passes and as a consequence objects expire. Expiration also can have the side effect of disrupting previously established paths.

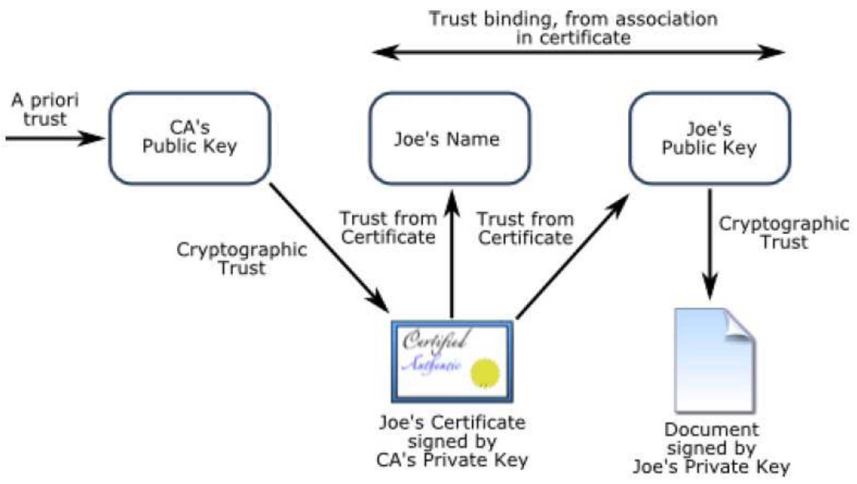

Fig. 1. Proposed approach

It is important to realize that while all these actions (synchronization, local and global object validation, expiration and revocation) must be performed before a user may ask the system for access, it is certainly not necessary or desirable to perform all these actions at once. A suitable segmentation of the software components helps distribute the processing burden more evenly over the twenty four hour processing interval. A second design optimization concerns the detailed nature of the data being processed. All four types of digital object (certificates, CRLs, DAOs and manifests) in the CPKI can be thought of as a collection of (variable, value) pairs, the values of which are immutable. Thus, once it has been determined, for example, that the expiration date on a certificate is 00:00:00 01-Jan-2017, that certificate will always have that expiration date. There are two direct implications from this elementary observation. The first implication is that since certificates are stored in files and since disk access is intrinsically more costly than in-memory operations, if an immutable part of a file is needed, it may make sense to extract and store that field (once) in a type of storage more highly optimized for structured access, e.g., a database.

In order to read any field, in addition to the file access cost (which must in any event be borne at least once), there is also the cost associated with finding and extracting the field in question. Certificates, CRLs, DAOs and manifests are all defined by means of complex, nested, data dependent structures, so that accessing a particular field in such a digital object involves a considerable amount of data structure traversal; it certainly isn't a random access operation. Naturally, there is a performance tradeoff implicit in the caching approach. Inserting an item in a database itself has a cost and searching for it subsequently also has a cost, so one must be ask whether a file-based approach or a database approach has the least overall cost.

Performance Testing

Functional, operational and performance testing has been conducted on an early prototype of the system. A substantial suite of unit tests has been constructed and executed, verifying that all software requirements have been met. In addition, subsystem tests designed to traverse the entire state space of possible processing variants has also been created and successfully executed. Most critically, a suite of performance tests has been run in order to validate that the system is truly scalable to real-world operational parameters. There are two different scenarios of interest from the viewpoint of performance. The first is the initial synchronization and loading, when the software starts from a clean state and does a full transfer of data into the cloud. The second is an incremental update, when the software starts with a local repository and database that reflects the state at the time of last execution and then reads only the changes to the state of the remote repositories.

The amount of work required for an incremental update depends on the number of objects added, updated, or removed since the previous update, which, in turn, depends in large part on the time since the previous update. We anticipate that an update will be performed roughly once a day; since the number of objects updated in this time period is typically only a fraction of the total number of objects, we focused on the initial synchronization and load as the performance bottleneck.

In order to construct a comprehensive performance test, a very large number of digital objects stored in multiple remote repositories are needed. Since DAOs and are a new types of object, there is no real data available. As a result, the author constructed a set of data for performance testing based on a characterization of the statistics of a typical cloud based account with a twenty thousand object test repository. We found that the entire processing chain, including loading, garbage collection, chasing and query processing took 34 min. This figure is a worst case time, as the structure of the test repository ensured that the manifest would be delivered last. Because each signed object is checked against the corresponding manifest, retrieving the manifest last requires that each newly fetched object be re-checked once the manifest arrives.

As a result, we fully anticipate that in a real-world scenario the average processing time for a repository of the same size be approximately 17 min (if, on average a manifest is fetched in the middle of the retrieval process). Under normal circumstances, the entire set of repository contents would not be processed; only the changes from the last processing cycle would need to be processed. If one assumes a 5% turn-over rate per day, then the total processing time would be less than a minute. Finally, it is worth noting that our test results were collected on a relatively slow (1.2 GHz) machine.

Theoretical Implications

The proposed solution adds to the current data attribution and ownership architecture the ability to accept long term storage of archival information by transferring the ownership service with attribution to the original source. In addition, CPKI can be used to provide the necessary clues about the state of the received data at a certain time period. This as a result would ensure the efficiency of the provided information by determining whether the certificate is issued by the original author as specified in the certificate or not. On the other hand, the proposed solution would help in archiving the information through technical mechanisms and appropriate procedures in which it verifies the received information by the time the private key is used to sign a document.

CPKI can also help increase the data attribution from the user side through the use of end entities based on the certificate to determine the public key of another entity.

Conclusion

This paper has described a proposed implementation of a software suite for a resource PKI in which the resources are certificates, CRLs, manifests and, most importantly Data Attributions Objects (DAOs). The CPKI software performs all the syntactic and semantic validation steps necessary in order to arrive at a set of trusted access control decisions. In the course of creating the early prototype CPKI software, several performance-optimized algorithms were developed for the "validate everything" paradigm of the CPKI. Performance testing indicates that even for very large repositories it will be possible to perform a complete validation run on a daily basis with little computational impact on the cloud operation center resources.

Acknowledgement

Thanks to the reviewers for their attention to detail and many valuable suggestions.

Funding Information

This research received no specific grant from any funding agency.

References

Aye, N., H.S. Khin, T.T. Win, T. KoKo and M.Z. Than *et al*., 2013. Multi-Domain Public Key Infrastructure for Information Security with use of a Multi-Agent System. In: Intelligent Information and Database Systems, Selamat, A., N.T. Nguyen and H. Haron (Eds.), Springer, pp: 365-374.

Cittadini, L., W. Mühlbauer, S. Uhlig, R. Bush and P. Francois *et al*., 2010. Evolution of internet address space deaggregation: Myths and reality. IEEE J. Selected Areas Commun., 28: 1238-1249. DOI: 10.1109/JSAC.2010.101002

Fujishiro, T., A. Sato, Y. Kumagai, T. Kaji and K. Okada, 2010. Development of hi-speed X.509 certification path validation system. Proceedings of the 24th International Conference on Advanced Information Networking and Applications Workshops, Apr. 20-23, IEEE Xplore Press, pp: 269-274. DOI: 10.1109/WAINA.2010.20

Ghazi, Y., R. Masood, A. Rauf, M.A. Shibli and O. Hassan, 2016. DB-SECaaS: A cloud-based protection system for document-oriented NoSQL databases. Eurasip J. Inform. Security, 2016: 16-16. DOI: 10.1186/s13635-016-0040-5

Housley, R., W. Ford, W. Polk and D. Solo, 1999. RFC 2459: Internet X. 509 public key infrastructure certificate and CRL profile. Network Working Group, Internet Engineering Task Force.

Housley, R., W. Polk, W. Ford and D. Solo, 2002. Internet X. 509 public key infrastructure certificate and Certificate Revocation List (CRL) profile. RFC 3280.

Huff, S.M., R.A. Rocha, H.R. Solbrig, M.W. Barnes and S.P. Schrank *et al*., 1998. Linking a medical vocabulary to a clinical data model using Abstract Syntax Notation 1. Methods Inform. Med., 37: 440-452. PMID: 9865042

Jensen, M., J. Schwenk, N. Gruschka and L.L. Iacono, 2009. On technical security issues in cloud computing. Proceedings of the IEEE International Conference on Cloud Computing, Sept. 21-25, IEEE Xplore Press, pp: 109-116. DOI: 10.1109/CLOUD.2009.60

Kent, S., 2006. An infrastructure supporting secure internet routing. Proceedings of the 3rd European Conference on Public Key Infrastructure: Theory and Practice, Jun. 19-20, Turin, Italy, pp: 116-129. DOI: 10.1007/11774716_10

Kent, S., C. Lynn and K. Seo, 2000. Design and analysis of the secure border gateway protocol (S-BGP). Proceedings of the DARPA Information Survivability Conference and Exposition, Jan. 25-27, IEEE Xplore Press, pp: 18-33. DOI: 10.1109/DISCEX.2000.824939

Liu, Y., Y. Sun, J. Ryoo, S. Rizvi and A.V. Va-silakos, 2015. A survey of security and privacy chal-lenges in cloud computing: Solutions and future di-rections. JCSE, 9: 119-133.

Montana, D. and M. Reynolds, 2008. Validation Algorithms for a Secure Internet Routing PKI. Lecture Notes Comput. Sci., 5057: 17-30. DOI: 10.1007/978-3-540-69485-4_2

Muñoz, J.L., J. Forne, O. Esparza and M. Soriano, 2004. Certificate revocation system implementation based on the Merkle hash tree. Int. J. Inform. Security, 2: 110-124. DOI: 10.1007/s10207-003-0026-4

Nasreldin, M.M., M. El-Hennawy, H.K. Aslan and A. El-Hennawy, 2015. New secure communication de-sign for digital forensics in cloud computing. Int. J. Comput. Sci. Inform. Security, 13: 8-17.

Oppliger, R., 2001. Secure Messaging with PGP and S/MIME. Artech House, Boston, ISBN-10: 158053161X, pp: 305.

Rose, M.T. and K. McCloghrie, 1990. Structure and Identification of Management Information for TCP/IP-based internets. Structure, United States.

Zhao, M., J. Walker and C.C. Wang, 2012. Security challenges for the intelligent transportation system. Proceedings of the 1st International Conference on Security of Internet of Things, Aug. 17-19, Kollam, India, pp: 107-115. DOI: 10.1145/2490428.2490444