Development in Key Share Management to Protect Data over Cloud

Priyanka Ambatkar
Dept. of Computer Sci & Engg
TGPCET, RTMNU
Nagpur, India.
piyuambatkar@gmail.com

Prof. P. Velavan
Head, Dept. of Computer Sci &Engg
TGPCET, RTMNU
Nagpur, India.
sppvels@gmail.com

Prof. Arun Katara
Dept. of Electronics Engineering
DMIETR, Sawangi (M) RTMNU.
Wardha, India.
arunkatara@gmail.com

***Abstract*—**User data may be stored in a cloud to take advantage of its scalability, accessibility, and economics. However, data of a sensitive nature must be protected from being read in the clear by an untrusted cloud provider. This triggered a lot of research activities, resulting in a quantity of proposals targeting the various cloud security threats. A key management scheme is proposed where encrypted key shares are stored in the cloud and automatically deleted based on passage of time or user activity. The process does not require additional coordination by the data owner, which is of advantage to a very large population of resource-constrained mobile users. The rate of expiration may be controlled through the initial allocation of shares and the heuristics for removal. A simulation of the scheme and also its implementation on commercial mobile and cloud platforms demonstrate its practical performance.

Index Terms*—Distributed systems, mobile computing, security, cryptography, scalability*.

*****

I. INTRODUCTION

There is a critical need to securely store, manage, and analyze the massive data. Cloud computing systems offer nearly area of vast storage and computation for clients. In many applications, however, the provider of cloud services cannot be deemed to be sufficiently trustworthy to permit storing and processing of data in the clear. Given that contemporary cloud applications are accessed by potentially thousands of mobile device users, an encrypted cloud storage solution requires scalable key management. In addition, because many users will be operating resource-constrained devices, any security protocol employed must minimize the amount of communication sessions required. Current key management practices typically focus on key generation and distribution among a large population of users.

Key management scheme is proposed where encrypted key shares are stored in cloud. The primary concern is that as authorized users join and leave a system, current keys must be re-generated and re-distributed to valid users, which is an unrealistic cost for mobile device users. As a key is generated it automatically gets deleted based on passage of time. The accessibility of the data gradually expires and revocation occurs as a result of the loss of sufficient key share. Some approaches suggest performing computationally-intensive key regeneration operations within the cloud to take advantage of its scalability, but these computations may prove too expensive in certain applications where processing overhead is undesirable. This work suggests concentrating on the utility of another highly economical asset of a cloud system: its permanent replicated storage, which can scale according to client demand, and is typically billed at a small fraction of a dollar per GB of data per month [1].

The key design factors for a cloud-based secure storage system that motivate this work include: no additional server-side logic being required on the cloud provider end; fine-grained data access; highly scalable sharing among multiple readers and writers; minimal computation required by mobile users; minimal communication required with the cloud provider; and no inherent trust of the provider existing, in terms of the administrator having unrestricted access to stored user data.

II. PROPOSED WORK

In this Proposed System we used various access control techniques have been proposed for encrypted file and key storage in the cloud. The cloud provider typically controls key management activities, or the data owner does so if the provider is untrusted, requiring additional network communication and components [2]. In some mechanisms where control rests within the domain of the client, such as cloud-based data re-encryption, the ability of the provider to scale for computation has been exploited by performing intensive cryptographic computation in the cloud [3]. The cloud's potential for scalable storage, however, has apparently been under-utilized for key management operations.

NIST (National Institute of Standards and Technology), in its Electronic Authentication Guideline [4], recommends secret sharing as a technique to be used to protect long-term credentials in its level 3 security definition for a CSP (Cloud Service Provider). Secret key sharing allows a secret such as key information to be divided into multiple shares [5]; these shares may be distributed among key generators using the concept of threshold decryption [6], or portions of a private key are distributed among users [7]. The challenge is that the client must assemble a key from multiple sources, potentially resulting in expensive communication overhead.

Rather than key shares being distributed on demand by some authority, it has been proposed that they be distributed across a network of nodes whose accessibility is subject to degradation over time. The Vanish system [8] distributes shares onto a DHT (Distributed Hash Table) that underlies a peer-to-peer file sharing network. It suggests the concept of "self-destructing data," where copies of data become unreadable over time due to the effect of user churn on the index. The problem with adapting the scheme to a cloud-based context is that it relies upon the availability of the shares among the nodes, which cannot be guaranteed. It requires that each user obtain key shares from multiple other nodes that form the index, which is an expensive proposition if the user is operating a mobile device. In the DEPSKY [9] storage system, shares are necessarily distributed across multiple clouds to form distributed trust and to restrict access. Each cloud provider has access to a single share and thus cannot decode the stored data; this requires support for a cloud-of-clouds. Also, because the data shares are unencrypted, each cloud must be independent and collusion assumed to be impossible. A straightforward approach employing PGP and AES encryption [10] would encounter challenges with scalability; for instance, the symmetric key used for encryption of user data may need to be encoded with the public key of each recipient. Rather, it is preferable for a data owner to perform a one-time encryption. If the same private key is shared by all users, then revocation would require some form of authentication to prevent access; the enforcement of it would require trust in the provider. So we will provide a key for amount of data store and then it is deployed on cloud.

III. SYSTEM AND CLOUD SERVICE MODELS

The main contribution of this proposed work is the novel utilization of a cloud's centralized data storage facility to store encryption key material as shares such that the provider cannot use them to decode user data also stored in the cloud. Unlike other key sharing techniques, the proposal makes use of the cloud's economical storage cost to maintain key material, and to degrade it over time, so that the cost of key re-generation is minimized. The proposed work is the first that the authors are aware of that exploits self-eroding key material in the cloud to achieve highly scalable access management for mobile users (see Fig. 1).

A. System model

Consider a large population of mobile device users that accesses data in the cloud. The users are highly constrained in terms of the number of communication sessions that they may engage in, due to the costly energy drain associated with wireless transmission. Users are expected to only communicate directly with the cloud, which is assumed to be nearly always available. Communication between users and the cloud takes place over an insecure wireless medium subject to the risk of eavesdropping; hence, communication security is necessary. The permanent cloud data store is accessed through a key-value mechanism, in which a valid key index must be supplied to retrieve the value stored at the index location.

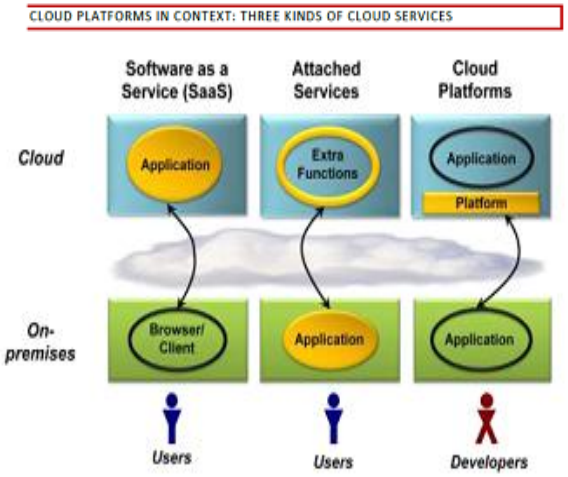

Fig. 1. Three types of cloud service

B. Cloud service models

Cloud Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure, which is accessible from various client devices, such as a Web browser. The consumer does not manage or control the underlying cloud infrastructure, or even individual application capabilities, with the possible exception of initial user-specific application configuration settings. The cloud provider is assumed to be an untrusted entity. It is honest, but may be curious, in that it will provide reliable service to users, including the provision of persistent storage capacity on demand, data replication to the extent that it is paid for, and will honors all data upload and download requests. However, the cloud administrator must be assumed to technically have full access to all data stored in the cloud, and may share knowledge of it with any party unbeknownst to the client. Thus, the supposition is that data must remain in encrypted format rest in the cloud. The provider may have malicious intent, and serve incorrect or

modified data to a user upon request; any such modification must be detected.

The mobile device users all belong to the same organization and can freely share information. Once a user's access rights are revoked, any valid key information in the user's possession may continue to provide access to encrypted user data. However, this apparent vulnerability is deemed to be only temporary in nature; in practice, mobile users have limited storage capacity and are unable to cache large data, including numerous key materials, for extended periods of time. This is especially true of a data storage system consisting of fine-grained access, where even individual data records may be encrypted with unique keys, and the storage of key material itself is onerous. Also, it is assumed that mobile device users cannot become compromised; other techniques related to computer security are required to ensure that secret information is not divulged between a mobile user and an outside attacker. Even if such an attack occurs, then the information illicitly gained must eventually become stale and unusable.

A supporting public key infrastructure is required to provide optional verifiability of key material, as will be discussed.

IV. CLOUD FRAMEWORK ARCHITECTURE

The architectural pattern described in the previous enables the cloud user to get some evidence on the integrity of the computations performed on a third-party's resources or services.

The architecture introduced in this section targets the risk of undesired data leakage. It answers the question on how a cloud user can be sure that the data access is implemented and enforced effectively and that errors in the application logic do not affect the user's data?

To limit the risk of undesired data leakage due to application logic flaws, the separation of the application system's tiers and their delegation to distinct clouds is proposed (see Fig. 2). In case of an application failure, the data are not immediately at risk since it is physically separated and protected by an independent access control scheme. Moreover, the cloud user has the choice to select a particular—probably specially trusted—cloud provider for data storage services and a different cloud provider for applications.

It needs to be noted, that the security services provided by this architecture can only be fully exploited if the execution of the application logic on the data is performed on the cloud user's system. Only in this case, the application provider does not learn anything on the users' data. Thus, the SaaS-based delivery of an application to the user side in conjunction with the controlled access to the user's data performed from the same user's system is the most far-reaching instantiation. Besides the introduced overhead due to the additionally involved cloud, this architecture requires, moreover, standardized interfaces to couple applications with data services provided by distinct parties. Also generic data services might serve for a wide range of applications there will be the need for application specific services as well.

The partitioning of application systems into tiers and distributing the tiers to distinct clouds provides some coarse-grained protection against data leakage in the presence of flaws in application design or implementation. This architectural concept can be applied to all three cloud layers. In the next section, a case study at the SaaS-layer is discussed.

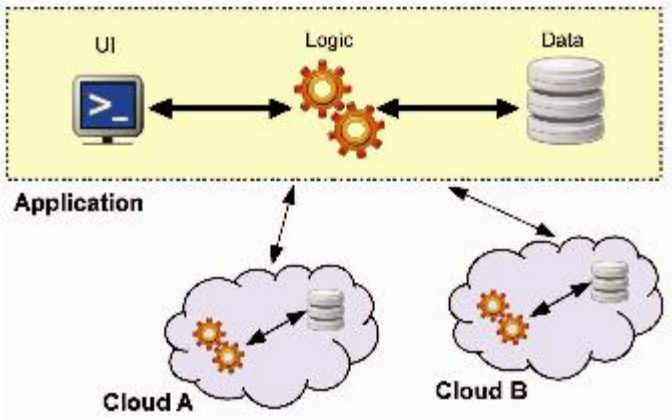

Fig. 2. Cloud service model

Only needs to rely on the assumption, that the cloud providers do not collaborate maliciously against herself.

Assume that n > 1 clouds are available (like, e.g., Clouds A and B in Fig. 1). All of the n adopted clouds perform the same task. Assume further that f denotes the number of malicious clouds and that n f > f the majority of the clouds are honest. The correct result can then be obtained by the cloud user by comparing the results and taking the majority as the correct one. There are other methods of deriving the correct result, for instance using the TurpinCoan algorithm [13] for solving the General Byzantine Agreement problem.

Instead of having the cloud user performing the verification task, another viable approach consists in having one cloud monitoring the execution of the other clouds. For instance, Cloud A may announce intermediate results of its computations to an associated monitoring process running at Cloud B. This way, Cloud B can verify that Cloud A makes progress and sticks to the computation intended by the cloud user. As an extension of this approach, Cloud B may run a model checker service that verifies the execution path taken by Cloud a on-the-fly, allowing for immediate detection of irregularities.

This architecture enables to verify the integrity of results obtained from tasks deployed to the cloud. On the other hand, it needs to be noted that it does not provide any protection in respect to the confidentiality of data or processes. On the contrary, this approach might have a negative impact on the confidentiality because—due to the deployment of multiple clouds—the risk rises that one of them is malicious or compromised. To implement protection against an unauthorized access to data and logic this architecture needs to be combined with the architecture described in Section V.

The idea of resource replication can be found in many other disciplines. In the design of dependable systems, for example, it is used to increase the robustness of the system especially against system failures [14]. In economic business processes—and especially in the management of supply chains—single-source suppliers are avoided to lower the dependency on suppliers and increase the flexibility of the business process [15]. In all these cases, the additional overhead introduced by doing things multi-ple times is accepted in favor of other goals resulting from this replication.
This architectural concept can be applied to all three cloud layers. A case study at the SaaS-layer is discussed in Section V.1.

V.1 CASE STUDIES: REPLICATING OF APPLICATION TASKS

Imagine a cloud provider named Instant Reporting that provides the service of creating annual accounting reports automatically out of a given set of business data. This is a very typical scenario of cloud usage, because such a report has to be published by all commercial entities once a year. Hence, the resources required to create such reports are only necessary for a small period of time every year. Thus, by using a third-party cloud service for this, in-house resources can be omitted, which would run idle most of the year. On the other side, by sharing its service capabilities among a large set of companies—all of which have to create their reports at different times of the year—a cloud service provider gains large benefits from providing such a shared service "on the cloud."

However, as promising as this scenario seems to be in terms of using the cloud computing paradigm, it contains a fundamental flaw: The cloud customers cannot verify that the annual report created by the cloud service is correct. There might have been accidental or intentional modifications of the source data for the report, or the processing logic that creates the reports from the source data might contain errors. In the worst case, the cloud system itself was compromised (e.g., by a malicious competitor) and all reports are slightly modified so that they look conclusive but contain slightly reduced profit margins, intended to make a competing company look bad or even insolvent.

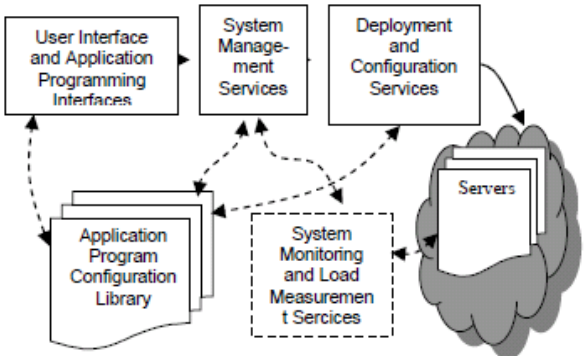

Fig. 3. Cloud framework architecture

V.1.1 DUAL EXECUTION

In such a situation, a first and trivial approach for verification might be that a cloud customer triggers the creation of its annual accounting report more than once. For instance, instead of giving the same request to one cloud provider only (called Cloud A hereafter), a second cloud provider (called Cloud B) that offers an equivalent type of service is invoked in parallel. By placing the same request at Clouds A and B, a cloud user can immediately identify whether his request was processed differently in Clouds A and B. Hence, this way, a secret exploitation of either side's service implementation would be detected. However, besides the doubled costs of placing the same request twice, this approach additionally relies on the existence of at least two different cloud providers with equivalent service offerings and comparable type of result. Depending on the type of cloud resources used, this is either easily the case—as even today there already exist many different cloud providers offering equivalent services (see Section 1)—or difficult in cases in which very specific resources are demanded

V.1.2 n CLOUDS APPROACH

A more advanced, but also more complex approach comes from the distributed algorithms discipline: the Byzantine Agreement Protocol. Assume the existence of n cloud providers, of which f collaborate maliciously against the cloud user, with $n > 3f$. In that case, each of the n clouds performs the computational task given by the cloud user. Then, all cloud providers collaboratively run a distributed algorithm that solves the General Byzantine Agreement problem (e.g., the TurpinCoan [16] or Exponential Information Gathering [16, 6.2.3] algorithms). After that it is guaranteed that all no malicious cloud providers know the correct result of the computation. Hence, in the final step, the result is communicated back to the cloud user via a Secure Broadcast algorithm (e.g., plain flooding, with the cloud user taking the majority as the result). Hence, the cloud user can determine the correct result even in presence of f malicious clouds.

V.1.3 PROCESSOR AND VERIFIER

Instead of having Clouds A and B perform the very same request, another viable approach consists in having one cloud provider "monitor" the execution of the other cloud provider. For instance, Cloud A may announce intermediate results of its computations to a monitoring process run at Cloud B. This way, Cloud B can verify that Cloud A makes progress and sticks to the computation intended by the cloud customer. As an extension of this approach, Cloud B may run a model checker service that verifies the execution path taken by Cloud a on-the-fly, allowing for immediate detection of irregularities.

One of the major benefits of this approach consists in its flexibility. Cloud B does not have to know all details of the execution run at Cloud A—especially not about the data values processed—but is able to detect and report anomalies to the cloud customer immediately. However, the guarantees given by this approach strongly depend on the type, number, and verifiability of the intermediate results given to Cloud B.

VI. PARTITION OF APPLICATION SYSTEM INTO TIERS

The architectural pattern described in the previous Section 4 enables the cloud user to get some evidence on the integrity of the computations performed on a third-party's resources or services.

The architecture introduced in this section targets the risk of undesired data leakage. It answers the question on how a cloud user can be sure that the data access is implemented and enforced effectively and that errors in the application logic do not affect the user's data?

To limit the risk of undesired data leakage due to application logic flaws, the separation of the application system's tiers and their delegation to distinct clouds is proposed (see Fig. 2). In case of an application failure, the data are not immediately at risk since it is physically separated and protected by an independent access control scheme. Moreover, the cloud user has the choice to select a particular—probably specially trusted—cloud provider for data storage services and a different cloud provider for applications.

It needs to be noted, that the security services provided by this architecture can only be fully exploited if the execution of the application logic on the data is performed on the cloud user's system. Only in this case, the application provider does not learn anything on the users' data. Thus, the SaaS-based delivery of an application to the user side in conjunction with the controlled access to the user's data performed from the same user's system is the most far-reaching instantiation. Besides the introduced overhead due to the additionally involved cloud, this architecture requires, moreover, standardized interfaces to couple applications with data services provided by distinct parties. Also generic data services might serve for a wide range of applications there will be the need for application specific services as well.

The partitioning of application systems into tiers and distributing the tiers to distinct clouds provides some coarse-grained protection against data leakage in the presence of flaws in application design or implementation. This architectural concept can be applied to all three cloud layers. In the next section, a case study at the SaaS-layer is discussed.

VII. CONCLUSION

The use of multiple cloud providers for gaining security and privacy benefits is nontrivial. As the approaches investigated in this paper clearly show, there is no single optimal approach to foster both security and legal compliance in an omniapplicable manner. Moreover, the approaches that are favorable from a technical perspective appear less appealing from a regulatory point of view, and vice versa. The few approaches that score sufficiently in both these dimensions lack versatility and ease of use, hence can be used in very rare circumstances only.

As can be seen from the discussions of the four major multicloud approaches, each of them has its pitfalls and weak spots, either in terms of security guarantees, in terms of compliance to legal obligations, or in terms of feasibility. Given that every type of multicloud approach falls into one of these four categories, this implies a state of the art that is somewhat dissatisfying.

However, two major indications for improvement can be taken from the examinations performed in this paper. First of all, given that for each type of security problem there exists at least one technical solution approach, a highly interesting field for future research lies in combining the approaches presented here. For instance, using the n clouds approach (and its integrity guarantees) in combination with sound data encryption (and its confidentiality guarantees) may result in approaches that suffice for both technical and regulatory requirements. We explicitly do not investigate this field here due to space restrictions; however, we encourage the research community to explore these combinations, and assess their capabilities in terms of the given evaluation dimensions.

Second, we identified the fields of homomorphic encryption and secure multiparty computation protocols to be highly promising in terms of both technical security and regulatory compliance. As of now, the limitations of these approaches only stem from their narrow applicability and high complexity in use. However, given their excellent properties in terms of security and compliance in multi-cloud architectures, we envision these fields to become the major building blocks for future generations of the multi-cloud computing paradigm.
It has been demonstrated that scalable key management may be attained by leveraging the inexpensive storage capacity and high accessibility offered by a cloud provider. One of

the benefits of using centralized and reliable cloud storage for key shares is that there is full control over share management; it is not subject to outside factors such as user churn. Thus, various additional heuristics for key share deletion may be explored. For instance, high-priority or trustworthy users could retain key shares for longer or be assigned a greater number.

REFERENCES

[1] Amazon. (2012) Amazon S3 Pricing. [Online]. Available: http://aws.amazon.com/s3/pricing/

[2] [2] S. Jahid, P. Mittal, and N. Borisov, "EASiER: encryption- based access control in social networks with efficient evocation,," in Proceedings of the 6th ACM Symposium on Information, Computer and Communica-tions Security, ASIACCS '11. New York, NY, USA: ACM, 2011, 411–415.

[3] P. Tysowski and M. A. Hasan, "Towards Secure Communication for Highly Scalable Mobile Applications in Cloud Computing System, " Centre for Applied Cryptographic Research, University of Waterloo, Tech. Rep. CACR 2011-33, 2011.

[4] W. E. Burr, D. F. Dodson, E. M. Newton, R. A. Perlner, W. T. Polk, S. Gupta, and E. A. Nabbus, "Electronic Authentication Guideline," National Institute of Standards and Technology (NIST), Tech. Rep. Special Publication 800-63-1, December 2011.

[5] A. Shamir, "How to share a secret," Commun. ACM, vol. 22 no. 11, 612–613, Nov. 1979.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in Advances in Cryptology — CRYPTO 2001, Lecture Notes in Computer Science, J. Kilian, Ed. Springer Berlin / Heidelberg, 2001, vol. 2139, pp. 213–229.

[7] J. Baek and Y. Zheng, "Identity-Based Threshold Decryption," in Public Key Cryptography – PKC 2004, ser. Lecture Notes in Computer Science, F. Bao, R. Deng, and J. Zhou, Eds. Springer Berlin / Heidelberg, 2004, vol. 2947, pp. 262–276.

[8] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in Proc. Of Ment To Multivalued Byzantine Agreement," Information Processing Letters, vol. 18, no. 2, pp. 73-76, 1984.

[9] A. Bessani, M. Correia, B. Quaresma, F. Andre,´ and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," in Proceedings of the sixth conference on Computer systems,.

[10] P. Zimmermann, "Pretty good privacy: public key encryption for the masses," in Building in big brother, L. J. Hoffman, Ed. New York, NY, USA: Springer-Verlag New York, Inc., 1995, pp. 93–107.

[11] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud—Protocols and Formats for Cloud Computing Interoperability," Proc. Int'l Conf. Internet and Web Applications and Services, pp. 328-336, 2009.

[12] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation," Proc. IEEE Third Int'l Conf. Cloud Computing (CLOUD), pp. 337-345, 2010.

[13] R. Turpin and B.A. Coan, "Extending Binary Byzantine Agree-ment To Multivalued Byzantine Agreement," Information Proces-sing Letters, vol. 18, no. 2, pp. 73-76, 1984.

[14] I. Koren and C.M.C. Krishna, Fault-Tolerant Systems. Morgan Kaufmann, 2007.

[15] J.D.J. Wisner, G.K.G. Leong, and K.-C. Tan, Principles of Supply Chain Management: A Balanced Approach. South- Western, 2011.

[16] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud—Protocols and Formats for Cloud Computing Interoperability," Proc. Int'l Conf. Internet and Web Applications and Services, pp. 328-336, 2009.