IT in Industry, vol. 9, no.2, 2021 Published online 25-Jul-2021

ROBUST GROUP-BASED SECRET SHARING FOR SECURE CLOUD KEY MANAGEMENT

A. Bentajer, Y. Said, Z. Igarramen, M. Hedabou

Abstract - Cloud services are now ubiquitous in each organization. However, they are facing different challenges for using Key Management System (KMS) with those cloud solutions. Besides, they must deal with concerns specific to multi-cloud key management. In This paper, we present a schema based on robust secret share based on group of shares that fulfills the requirements of multi-cloud deployment. The system split the encryption keys into a blinded version of n shares that will be stored at cloud side and deals with a group of t+1 shares to compute the initial secret. To demonstrate the practicality of the proposed design, we implement a fully featured prototype and evaluate its performance. Results analysis shows that the proposed design is highly efficient and can serve as groundwork for using secret share to protect keys in a multi-cloud environment.

Keywords **- About four key words or phrases in alphabetical order, separated by commas.**

I. INTRODUCTION

CLOUD computing services are in high demand because they reduce the cost and complexity of owning and managing an on-premise IT infrastructure. Moreover, cloud users have no investment in information technology infrastructure, purchase hardware, or buy software licenses because these charges are covered by cloud providers. However, many chief executives and IT managers are not willing to lose the control of their data by offloading it to cloud computing platforms. Their main concerns are about confidentiality, integrity, and privacy of the outsourced data [1]-[3].

CSPs and researchers in the field are leveraging cryptography to prevent data loss/breach and guarantee data confidentiality [4]-[7], [11]. Cryptography can be involved in two major levels of security, namely secure storage [4]-[7] and secure computation [12]-[15].

A. Bentajer is with the SIGL Laboratory at National School of Applied Science, Tetouan, Morocco.
Y. Said., is with the SIGL Laboratory at National School of Applied Science, Tetouan, Morocco.
Z. Igarramen is with the MTI Laboratory at National School of Applied Science, Safi, Morocco.
M. Hedabou is with the UM6P-CS Laboratory at UM6P University, Benguerir, Morocco.

Key Management System (KMS) is built around a cryptographic module and leverages the cryptographic functionality such as keys generation, storage, archiving, distribution, and destruction at the end of their life cycles [21]-[24]. Due to their sensitivity, the use of FIPS 140-2 Level 2 or higher certified cryptographic modules for protecting critical cryptographic keys is required [9], [25]. Very likely, this is done by making use of hardware facilities. For users, smart card or TPM [20] can be applied, whereas HSM can fit more companies and government needs. Needless to say, that HSM has been developed before the advent of cloud computing paradigm, therefore they must go along with a key management system as it occurs on-premise infrastructures.

To alleviate cloud users from managing keys, which are their main goal of embracing cloud services, cloud providers offer HSM as a service. With AWS CloudHSM, Amazon provide HSM appliances in data centers as a service to users [16]. Undoubtedly, the physical HSM limitations related to lack of elasticity and operability have been addressed by HSM as-service, still there is need for software infrastructure, owned, and procured by cloud services providers, to drive HSM as service. In a nutshell, HSM as-service has brought some desired security and easy management properties, but the HSM technology was not originally developed for cloud and still presents limitations from cloud users' perspective.

Recently, another approach achieving effective cloud key management, based on the use of homomorphic encryption, was put forward. It was dedicated expressly to cloud services and was designed in such a way to meet the five characteristics of the cloud computing paradigm including elasticity, On demand self-service and availability. The solution is already integrated with Web Services (AWS) and RedHat, but it works with any cloud platform.

Security can only proved in semi-honest model. The solution provider, namely Porticor, must be trusted to implement the protocol as specified and cloud providers' platforms executing the implementation to have a neutral behavior. Semi-honest

 ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

models are believed to be non-trivial task and thus may undermine the security gain provided by the solution. The lack of detailed technical information about the solution and about the span of its adoption by final cloud users make the final statement very difficult.

This paper introduces a new design for secure and efficient software cloud key management system. Based on (t, n) robust secret sharing mechanism, the proposed design splits up the encryption key on n servers hosted on cloud computing providers side that communicate on asynchronous private and authenticated channels. The system deals with t+1 shares to compute the initial secret which, compared to the initial proposed solution [26], enhance the performance of the verification process. The design tolerates up to (n-t-1) faulty servers. In addition, the storage of public information, namely the Lagrange coefficients, speeds up the computation of the secret (master key) reconstruction making the design more efficient. Furthermore, we construct a formal model of our design and prove its security in semi-honest model and finally we report on a prototype of implementation along with the performance study. Well established trusted computation and execution facilities will be leveraged to share, store, and securely compute the key shares and reconstruction.

The remainder of this paper is structured as follows. Section 2 presents preliminaries and basic design. Section 3 aims to present each protocol in the design, while section 4 presents the design implementation, security analysis and performances. Finally, we come-up with our conclusions and assumptions.

II. PRELIMINARIES

A. Secret Sharing

Please submit your manuscript electronically for review as e-mail attachments. When you submit your initial full paper version, prepare it in two-column format, including figures and tables.

The secret sharing theory is a very attractive research field. It has many applications; multiparty computation is by far the most relevant one. In this paper, we focus on Shamir based secret sharing schemes [10]. We assume that a dealer wants to share a secret s amongst n parties so that no less than t + 1 parties can recover the secret, whereas it can easily be recovered by any t + 1 or more parties. This is referred to as (t; n) secret sharing. Shamir based secret sharing scheme is built upon polynomials over finite field F, with |F| > n. For the sake of correctness and simplicity, we suppose that $F = F_p$ with $p > n$.

Whereas it can easily be reconstructed from any t+1 or more shares. Both facts are proved using Lagrange interpolation.

Shamir's early idea [10] of distributing shares of a secret as evaluations of a polynomial has become a standard building block in threshold cryptography. The scheme is based on polynomial interpolation. Given k couples (x_i; y_i), with distinct x'_i s, there is one and only one polynomial q(x) of degree k -1 such that $q(x_i) = y_i$ for all *i*. This basic statement can be proved by using Lagrange interpolation. Without loss of generality, we can assume that the secret *s* is (or can be made) a number. To divide it into pieces $[s]_i$, we pick a random k - 1 degree polynomial $q(x) = a_0 + a_l x + \cdots + a_t x^t$ in which q(0) = s , and evaluate (1):

$$s_1 = q(1) \dots s_i = q(i) \dots s_n = q(n) \qquad (1)$$

Given any subset of t + 1 of these [s]i values (together with their identifying indices), we can find the coefficients Li of q(x) by interpolation, and then evaluate (2) :

$$s = q(0) = \sum_{i=1}^{i=t+1} L_i[s]_i \, , where \; L_i = \prod_{j \neq i} \frac{x_j}{x_j - x_i} \qquad (2)$$

The basic secret sharing scheme will have some flaws if some participants are dishonest [17]. For withstanding malicious participants, a new type of secret sharing scheme was proposed by Fieldman [18], called the verifiable secret sharing (VSS) scheme. The coefficients of this polynomial hidden in the exponent of the generator of a group in which the discrete-log assumption holds, are published. This allows that the participants can validate correctness only of their own shares distributed by the dealer in the distribution phase. In [19], Stadler introduced the publicly verifiable secret sharing (PVSS) scheme that allows that anyone can verify the validity of shares without revealing any secret information.

B. Model and Assumption

In this paper, we are dealing with scenarios where a software key management system following the Control Your Own Keys (CYOK) model is deployed in the cloud providers side as an ad-on facility to an existing on-premises key management system. The keys are managed on the on-premises side, following best practices, keys are exported, stored, and handled in the cloud provider sides in a secure way. While they are in transit, conventional and well-established techniques, including SSL, SSH DH

 ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

key exchange are leveraged to achieve security. In the cloud providers side, keys are stored in a distributed way through *n* servers S_1, S_2, … ; S_n. Later, keys are reconstructed, and their integrity is verified by using secure computation approaches. The proposed protocol can be modeled as follows:

- **Secure storage**. During the lifetime of the application, all servers possess some sensitive information to be stored in a secure and authenticated way. This could be shares of the keys or pieces of MAC's.
- **Secure computation**. The n servers S_1; S_2; · · · ; S_n are involved in some secure computation phase taking place in the cloud providers side.
- **Online and Offline phases**. The protocol goes through periods where servers are active and other where they are idle. In the active periods, the servers are requested to send back their keys and MAC's shares to a dealer who conduct the secure computation for reconstructing and checking the validity of recovered keys. These periods, called on online phases, alternate with other where the servers are inactive. The latter periods are called Offline phases. The Offline and Online phases must be synchronized to switch between these Offline and Online periods.

As for on-premises settings, the proposed key management system is implemented as a stand-alone application. The servers in cloud computing sides are fully autonomous, that is they can switch between offline and online phases without any interactions with outside the cloud instances. The servers can only communicate with each other to conduct the whole process. In other words, the only players involved are the servers themselves. This model comes with a limited level of confidentiality, availability that can be provided. This level is tightly related to the number of servers required for restoring the secret key without the leakage of any information about it.

The confidentiality threshold can be defined as the minimal number $Conf_{min}$ of server an adversary can break into to learn the secret key, whereas the availability threshold $Avail_{min}$ as the minimal number of uncorrupted servers that should be available for restoring the key. In a fully autonomous scenario, the number of malicious servers $n\text{-}Avail_{min}$ must be at most n/2 for ensuring confidentiality and availability of the protocol. This limitation which is mainly due to the requirement that servers are not allowed to communicate with any instance from outside the cloud. The requirement about the number of malicious servers can be relaxed by limited interaction with on-premises key management system.

We make standard assumptions about the well-established cryptographic techniques regarding the ability of an adversary to undermine their security. The techniques used for establishing secure and private channels or for authenticating parties, including SSL, SSH, DH key exchange are assumed secure in standard models.

C. Basic Design

We here give an informal description of our protocol that implement cloud key management system based on Robust Verifiable Secret Sharing with fully autonomous servers. The protocol consists in three main phases. A set up phase where the on-premises key management system computes the group of shares of the encryption key and the MAC. The second phase is where the servers enter an offline period after receiving their shares and the final one consisting in conducting secure computation to reconstruct the initial secret after they return to online period. The two subsequent phases are launched by the main instance.

Our protocol for key management in cloud computing, denoted CloudKMS consists of three main components, namely the key management system on-premise, an appliance acting as the dealer and n servers S1, S2, …, Sn. The appliance and servers, owned and managed by cloud users, are in cloud side. We assume that the appliance is a trusted component. It is a semi-honest component (passive), which means that it behaves as prescribed by the protocol. This goal can be achieved by issuing remote attestation for its software. The protection against passive attackers (an eavesdropper) is provided by a leveraging a trusted execution mode such as SGX enclave and by using standard cryptographic tools like SSL, SSH.

The protocol CloudKMS assume that a key management system is already active on the user side (on premise). The KMS is responsible for generating the keys that will be used on the cloud computing side following the model CYOK. For the sake of simplicity, we assume that we are dealing with a single key, the encryption key K. The protocol CloudKMS can be conducted in three sub protocols. A Set up protocol generating the public parameters and computing the shares of the key k, denoted [s]i. The sub protocol Sharing delivers the shares [s]i to the servers Si in a secure and authenticated way. The third sub protocol Reconstruction allows to get back the share and to conduct computations and reconstruct the K. We now introduce the formal model of our protocol

 ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

CloudKMS following the model CYOK. As mentioned above, the protocol consists of three sub protocols.

- Protocol Setup (k) : executed by the key management system on-premise, it takes the security parameter ***k*** and outputs the finite field F_p and the groups of public shares $[s_i]_j$ (for i in 1 … n/(t + 1) and j in 1 … n) of the *k* along with the public shares of the MAC $[\gamma]_i$ for each group of shares (for i in 1 … n/(t + 1)).
- Protocol Sharing ($[s_i]_j$; $[\gamma]_i$): Executed by the trusted appliance, it takes each groups of shares of *k* ($[s_i]_j$) and $[\gamma]_i$. The protocol delivers/retrieves the groups of shares $[s_i]_j$; $[\gamma]_i$, for i in 1, … , n/(t + 1) and j in 1… n to/from the servers
- Protocol Reconstruction ($[s_i]_j$; $[\gamma]_i$) takes as inputs the group of shares of k and the MAC. The protocol executed by the trusted appliance compute the MAC of received shares (γ') and compare it with the value of received MAC $[\gamma]_i$, if values match then it outputs the secret k, else it rejects the received group of shares and retrieve next one.

For example, if we have n = 12 and t = 3. The group of shares will be [s_1; s_2; s_3; s_4] with MAC γ_1, second group [s_5; s_6; s_7; s_8] with MAC γ_2 and third group [s_9; s_{10}; s_{11}; s_{12}]with
MAC γ_3

III. THE PROPOSED PROTOCOL

If you are using Word, use either the Microsoft Equation Editor or the MathType add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation or MathType Equation). "Float over text" should not be selected.

In this section we describe the main 3 sub protocols of our cloud key management system cloudKMS, namely SetUp, Sharing and Reconstruction.

We denote the number of servers by n and the security parameter (encryption key) by k. We assume that there is one k from which we derived the specific shares. The k will be used during the life of the system. As mentioned before, the traffic between the components of the system is encrypted and authenticated using standard cryptographic tools.

A. Setup Protocol

Executed by an application on behalf of the key management system on premise, it takes k. Outputs the finite field Fp and the groups of public shares [si]j (for i in 1 … n/(t+1) and j in 1 … n) of the k along with the public shares of the MAC [γ]i for each group of shares for i in 1 … n/(t + 1). The algorithm in figure 1 depicts the implementation of the protocol:

Algorithm 1 Sub protocol Set_{up}

function SETUP(k)
  Sample random number $a_1, \cdots, a_n \in Z_p$
  Set $a_0 \leftarrow k$
  Set $q(x) \leftarrow \sum_{i=0}^{i=t} a_i x^i$
  Compute $[s]_i \leftarrow p(i)$ for $i = 1, \cdots, n$
  Compute $[\gamma]_j$ for each group of shares
  Invoke Share protocol

Fig. 1 : Setup protocol algorithm

B. Sharing Protocol

This protocol is executed with SetUp and Reconstruction protocols. It takes groups of shares of k ($[s_i]$) and MAC $[\gamma]_j$. The protocol delivers/retrieves the group of shares $[s]_i$; γ_j, to/from the servers. The value of the MAC γ0 for a group of shares is computed and compared to $[\gamma]_j$ before to reconstruct the initial secret. (Algorithm in figure 2)

Algorithm 2 Sub protocol *Sharing*

function SHARING($[s]_i$, γ_j)
  Send/Retrieve $[s]_i$, γ_j to Servers or Appliance

Fig. 2 Sharing protocol algorithm.

C. Reconstruction Protocol

Takes as inputs the group of shares of k and the MAC. The protocol executed by the trusted appliance compute the MAC of received shares (γ0) and compare it with the value of received MAC [γ]i, if values match then it outputs the secret k, else it rejects the received group of shares and retrieve next one (Algorithm figure 3).

Algorithm 3 Sub protocol *Reconstruction*

function POST(S, A)
  Compute $L_i \leftarrow \prod_{j \neq i} (\frac{x_j}{x_j - x_i})$
  Compute $\gamma' \leftarrow \text{MAC}[s_i]$ for $i = 1 \cdots n/t + 1$
  if $\gamma' == gamma_j$ **then**
    compute k
  else
    Retrieve next group of Share
    State the group of shares as compromised

Fig. 3 Reconstruction protocol algorithm.

 ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

IV. SECURITY ANALYSIS AND PERFORMANCE EVALUATION

Figure 4 shows a presentation of how tests were conducted and where the shares were stored/retrieved.

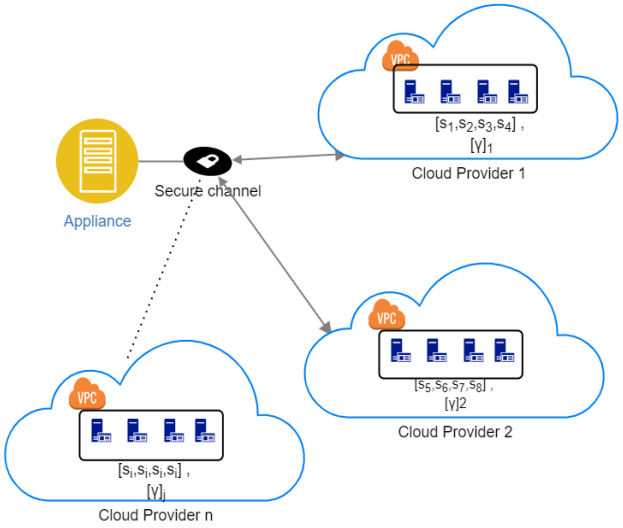

Fig. 4 : General Schema

We implemented our 3 protocols with Java 1.8 using a 64 bits Windows operating system with Intel Xeon Gold 5217 (3.0 GHz) processor and 256Go installed RAM. The Sharing protocol has been developed using JCraft library which is a pure Java implementation of SSH2 that is known to have more defensive mechanisms to avoid vulnerabilities. Experiments are performed on different key size (AES-128, AES-192, AES-256) while the *t* and *n* of shamir secret sharing were t = 3 and n = 12.

We measured the performance of the proposed protocol using our developed prototype. We divided the overhead time of each measurement into:

- Setup and Share: The split of secret into groups of shares, MAC computation and the Upload time to Servers
- Share and Reconstruction: The download time from Server, computation of MAC and reconstruction of secret.

Tables 1 and 2 show the results of the running time for computation and file upload/download in seconds.

Table I. Setup And Share Protocols Performances

Key Size	SSS Computation	Upload	Total	Latency Upload
AES-128	0.03	9	9.03	± 0.5
AES-192	0.03	9	9.03	± 0.5
AES-256	0.03	9	9.03	± 0.5

Table II. Share And Reconstruction Protocols Performances

Key Size	SSS Computation	Upload	Total	Latency Upload
AES-128	0.007	5.42	5.427	± 0.33
AES-192	0.007	5.42	5,427	± 0.33
AES-256	0.007	5.42	5.427	± 0.33

Analysis proves that data transmission is a dominant factor while shares computation do not heavily penalize the performance of the proposed design for different key size. The latency time depends mainly on network quality for file transfer and the time taken by the appliance to authenticate to servers.

The secret share addresses the specific need to enhance the security of the key during its lifetime. The use of secret share schema establishes a mechanism of sharing sensitive data securely amongst an untrusted network. Besides, our proposed design inherits some properties related to Shamir's (k,n) thresholds as:

- The system is Information-theoretic security meaning that an attacker with high computational power cannot break the secret without having minimum number of thresholds required to reconstruct the key.
- The system is extensible, where ki could be dynamically added/removed without affecting other shares.
- The size of each share does not exceed the size of the original data.

However, during the upload/download of the shares it was noticed that the system freezes or takes longer than usual to upload or download the shares, this is usually due to the network connection and/or the interactions of the Appliance with the servers to authentication management. In addition, if a share is compromised, it will be difficult to know which part was affected.

Analysis proves that data transmission is a dominant factor while shares computation do not heavily penalize the performance of the proposed design for different key size. The latency time depends mainly on network quality

for file transfer and the time taken by the appliance to authenticate to servers.

The secret share addresses the specific need to enhance the security of the key during its lifetime. The use of secret share schema establishes a mechanism of sharing sensitive data securely amongst an untrusted network. Besides, our proposed design inherits some properties related to Shamir's (k, n) thresholds as:

- The system is Information-theoretic security meaning that an attacker with high computational power cannot break the secret without having minimum number of thresholds required to reconstruct the key.
- The system is extensible, where ki could be dynamically added/removed without affecting other shares.
- The size of each share does not exceed the size of the original data.

However, during the upload/download of the shares it was noticed that the system freezes or takes longer than usual to upload or download the shares, this is usually due to the network connection and/or the interactions of the Appliance with the servers to authentication management. In addition, if a share is compromised, it will be difficult to know which part was affected.

The mere fact that the Appliance is hosted in on-premises does not mean that it is completely trusted. A malicious insider can still tamper with the application. This issue depends mainly on the organization hosting the KMS itself. Intel SGX may be leveraged to offer hardware-based memory encryption and isolates the running Appliance code and data in memory from processes running at a higher privilege level.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a secure cloud key management based on the robust secret group share. The protocol is based on Shamir secret sharing that securely distribute fragments of secret key amongst a different distributed cloud server. We have also implemented a prototype of our proposed prototype to demonstrate its practicality. The results are promising, the computation of shares and MAC are very negligible compared to data transfer. In the future, we plan to improve the proposed Appliance using Intel SGX which will give it more protection from disclosure or modification. And implementing a sub-Appliance that will split the share of a server into other shares to improve the security of the secret.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in American English is without an "e" after the "g." Use the singular heading even if you have many acknowledgments. Avoid expressions such as "One of us (S.B.A.) would like to thank ... ." Instead, write "F. A. Author thanks ... ." Sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page.

REFERENCES

[1] M. Albanese, A. De Benedictis, D. D.J. de Macedo, F. Messina, "Security and trust in cloudapplication life-cycle management", Future Generation Computer Systems (2020), Volume 111,Pages 934-936, doi:10.1016/j.future.2020.01.025.

[2] P. J. Sun, "Security and privacy protection in cloud computing: Discussions and challenges", Journal of Network and Computer Applications 160 (2020) 102642. doi:10.1016/j.jnca.2020.102642.

[3] A. Bentajer, M. Hedabou, K. Abouelmehdi, Z. Igarramen, S. El Fezazi, "An IBEbased design for assured deletion in cloud storage", Cryptologia 43 (3) (2019) 254-265.doi:10.1080/01611194.2018.1549123.

[4] A. Bentajer, M. Hedabou, K. Abouelmehdi, S. Elfezazi, CS-IBE: AA data confidentiality system in public cloud storage system, in: Procedia Computer Science, Vol. 141, Elsevier B.V.,2018, pp. 559-564. doi:305 10.1016/j.procs.2018.10.126.

[5] Z. Igarramen, M. Hedabou, FADETPM: Novel approach of file assured deletion based on trusted platform module, in: Lecture Notes in Networks and Systems, Vol. 49, Springer, 2019, pp. 49-59. doi:10.1007/978-3-319-97719-5_4.

[6] J. Xiong, Y. Zhang, S. Tang, X. Liu, Z. Yao, Secure Encrypted Data with Authorized Deduplication in Cloud, IEEE Access 7 (2019) 75090-75104. doi:10.1109/ACCESS.2019.2920998.

[7] R. Chandramouli, D. Pinhas, Security Guidelines for Storage Infrastructure, Tech. rep., National Institute of Standards and Technology, 315 Gaithersburg, MD (oct 2020).doi:10.6028/NIST.SP.800-209.

[8] R. Chandramouli, M. Iorga, S. Chokhani, Cryptographic Key Management Issues and Challenges in Cloud Services, Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD (sep 2013). doi:10.6028/NIST.IR.7956.

 ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

[9] Bentajer A, Hedabou M, Chapter 6. Cryptographic Key Management Issues in Cloud Computing, in: Victoria M. Petrova (Ed.), Advances in Engineering Research, 34th Edition, Nova Science Publishers, Inc., 2020,

[10] A. Shamir, How to share a secret, Communications of the ACM 22 (1979) 612-613. doi:10.1145/359168.359176.

[11] W. Shi, T. Liu and M. Huang, "Design of File Multi-Cloud Secure Storage System Based on Web and Erasure Code," 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2020, pp. 208-211, doi: 10.1109/ICSESS49938.2020.9237703.

[12] Catrina O., Saxena A. (2010) Secure Computation with Fixed-Point Numbers. InProceedings: Sion R. (eds) Financial Cryptography and Data Security. FC 2010. Lecture Notes in Computer Science, vol 6052. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14577-3_6

[13] M. Nassar, A. Erradi, F. Sabry and Q. M. Malluhi, "A Model Driven Framework for Secure Outsourcing of Computation to the Cloud," 2014 IEEE 7th International Conference on CloudComputing, Anchorage, AK, USA, 2014, pp. 968-969, doi: 10.1109/CLOUD.2014.145.

[14] Q. Wang, F. Zhou, C. Chen, P. Xuan and Q. Wu, "Secure Collaborative Publicly Verifiable Computation," in IEEE Access, vol. 5, pp. 2479-2488, 2017, doi: 10.1109/ACCESS.2017.2672866.

[15] A. Bilakanti, Anjana N.B., Divya A., K. Divya, N. Chakraborty and G. K. Patra, "Secure computation over cloud using fully homomorphic encryption," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, 2016, pp. 633-636, doi: 10.1109/ICATCCT.2016.7912077.

[16] X. Huang and R. Chen, "A Survey of Key Management Service in Cloud," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 916-919, doi: 0.1109/ICSESS.2018.8663805.

[17] Schoenmakers B. (1999) A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In: Wiener M. (eds) Advances in Cryptology | CRYPTO' 99.CRYPTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48405-1_10

[18] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), Los Angeles, CA, USA, 1987, pp.427-438, doi: 10.1109/SFCS.1987.4.

[19] Stadler M. (1996) Publicly Verifiable Secret Sharing. In: Maurer U. (eds) Advances in Cryptology | EUROCRYPT '96. EUROCRYPT 1996. Lecture Notes in Computer Science, vol 1070. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-68339-9_17

[20] M. Hedabou and Y. S. Abdulsalam, "Efficient and Secure Implementation of BLS Multi signature Scheme on TPM" 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 2020, pp. 1-6, doi: 10.1109/ISI49825.2020.9280511.

[21] E. Barker, M. Smid, D. Branstad, S. Chokhan, " SP 800-130 : A Framework for Designing Cryptographic Key Management Systems" 2013, NIST, doi: 10.6028/NIST.SP.800-130

[22] E. Barker, D. Branstad, M. Smid, "SP 800-152 A Profile for U.S. Federal Cryptographic Key Management Systems (CKMS)" 2015, NIST, doi: 10.6028/NIST.SP.800-152

[23] E. Barker, "SP 800-57 Part 1 Rev. 5 Recommendation for Key Management: Part 1 General"2020, NIST, doi: 10.6028/NIST.SP.800-57pt1r5

[24] E. Barker, W. Barker, SP 800-57 Part 2 Rev. 1 Recommendation for Key Management: Part 2 Best Practices for Key Management Organizations" 2019, NIST, doi:10.6028/NIST.SP.800-57pt2r1

[25] NIST, " FIPS 140-3 Security Requirements for Cryptographic Modules" (2019). doi: https://doi.org/10.6028/NIST.FIPS.140-3

[26] A. Bentajer, M. Hedabou, S. Ennaama, A. Tahiri, "Secure Cloud Key Management based on Robust Secret Sharing", 10th International Conference on Cryptography and Information Security (CRYPIS 2021), Sydney, Australia, 2021, pp. 149-161, doi:10.5121/csit.2021.110913

AUTHORS

A. Bentajer received his M.S. degree from National School of Applied Sciences from Cadi Ayyad University in 2012. In 2019, he received his Ph.D degree in computer science from EST of Safi from Cadi Ayyad University, Marrakech, Morocco. Currently he is a professor at ENSA of Tetouan. His area interest covers Information Security, Security architecture, Identity based cryptography and cloud computing.