Secure Cloud Key Management based on Robust Secret Sharing

Ahmed Bentajer[1], Mustapha Hedabou[2], Sara Ennaama[1], and Abderrahim Tahiri[1]

[1]SIGL LAB., ENSA of Tetouan, University Abdelmalek Essaadi Tetouan, Morocco
[2]UM6P Benguerir, Morocco

Abstract. The aim of this paper is to propose a model to strengthen the security of key management in cloud computing, where the model is shared or entirely controlled by a non-trusted third party provider. Key management is not a straightforward matter for IT-teams, in addition to critical issues related to properly managing and securing the keys on providers' infrastructures, they have to deal with concerns specific to multi-cloud key management. Hardware Security Module (HSM) solution that offers a secure on-premise encryption key management turned out be impracticable for widespread cloud deployment. HSM as a Service seems to be the best approach for key management in multi-cloud, but the service is wholly owned and managed by another cloud provider. In This paper, we present an efficient and secure cloud key management that fulfills the requirements of multi-cloud deployment. The proposed design splits the key into a blinded version of n shares that will be stored in encrypted format at the cloud provider side. To demonstrate the efficiency 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 a groundwork for using secret share as a way to protect keys in a multi-cloud environment.

Keywords: Key Management Security, Secret sharing, MultiCloud , Cryptography, Security and Privacy

1 Introduction

According to a study by the International Data Group, 81% of organizations have at least one application or a portion of their computing infrastructure in the cloud [1]. This is due to the economic nature of cloud computing, which can reduce the cost and complexity of owning and managing internal infrastructure in an on-demand and pay-as-you-go metric. However, its adoption lead to data control loss to an unreliable Cloud Service Provider (CSP). The main concerns are about confidentiality, integrity and privacy of the outsourced data [2, 3]

CSPs are leveraging cryptography as lever for mitigating security concerns in order to strength confidence of end users on their services. Cryptography can be involved in two major levels of security, namely secure storage [11, 4–7] and secure

David C. Wyld et al. (Eds): ITCSE, ICDIPV, NC, CBIoT, CAIML, CRYPIS, ICAIT, NLCA - 2021
pp. 149-161, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.110913

computation [12–15]. Recently, some commercial offers of secure storage services with encrypted data were implemented in cloud infrastructures. The most known secure storage services, which encrypt data on the client side prior to outsourcing it, are Spideroak and Dropbox.

As for On-premise infrastructure, protecting digital assets and secure communication depend mainly on cryptography, the increasing of cyber attacks led the management of cryptographic keys more important than the key itself. This means that companies need to be more vigilant in key management at the cloud level. Depending on the type of cloud service in use, most key management functions are partially or fully controlled by the cloud providers. For PaaS and SaaS service delivery, the major part of the key management is processed internally by the cloud providers. Even for IaaS model, keys used for signing virtual machine template are internally managed [8].

Key management encompasses operations like keys generation, storage, archiving, distribution and destruction at the end of their life cycles. Due to their sensitivity, keys must be handled with care. Keys must be generated in a random way, stored in a very safe place and exchanged via secure protocols [8, 9]. Very likely, this is done by making use of hardware facilities. For particular 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 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 master key on n servers hosted on cloud computing providers side that communicate on asynchronous private and authenticated channels. 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

2 preliminaries

2.1 Secret Sharing

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 particular 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 refereed 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 of these 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 $q(x) = a_0 +_l x + \cdots + a_t x^t$ in which $q(0) = s$, and evaluate:

$$s_1 = q(1) \cdots s_i = q(i) \cdots s_n = q(n).$$

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

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

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.

2.2 Model and assumptions

In this paper, we are dealing with scenarios where a software key management system Following the BYOK (Bring Your Own Keys) model is deployed in the cloud providers side as an ad-on facility to an existing on-premise key management system. The keys are managed on the on-premise 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 key exchange are leveraged to achieve security. In the cloud providers side, keys are stored in a distributed way through n servers $S_1, S_2, \cdots, S_n$. Latter, 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 $S_1, S_2, \cdots, S_n$ 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 in order to switch between these Offline and Online periods.

As for on-premise 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 in order 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 server that should be available for restoring the key. In a fully autonomous scenario, the number of malicious servers n-$Avail_{min}$ 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-premise 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.

2.3 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-premise key management system computes the shares of the master key and the MAC and communicates them to the main instance (dealer) in cloud provider side trough a secure channel. The second phase is where the servers enter into an offline period after receiving their shares and the final one consisting in conducting secure computation to reconstruct the secret key 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 $cloud_{KMS}$ consists of three main components, namely the key management system on-premise, an appliance acting as the dealer and n servers $S_1, S_2, \cdots, S_n$. The appliance and servers, owned and managed by cloud users, are located 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 $cloud_{KMS}$ 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 BYOK. For the sake of simplicity, we assume that we are dealing with a single key, the master key K. The protocol $cloud_{KMS}$ 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 S_i in a confidential and authenticated way. The last sub protocol Reconstruction allows to get back the share and to conduct computations and reconstruct the master key. We now introduce the formal model of our protocol $cloud_{KMS}$ following the model BYOK (Bring Your Own Keys). As mentioned above, the protocol consists of three sub protocols.

- Protocol $Set_{up}(k)$: executed by the key management system on-premise, it takes the security parameter k.The protocol outputs the corresponding MAC of k denoted γ, the finite field F_p and the public shares $[s]_i$ of the master key along with the public shares of the MAC γ'_i for $i = 1, \cdots, n$.
- Protocol $Sharing(\gamma, [s]_i, \gamma'_i)$: Executed by the trusted appliance, it takes shares of the master key and MAC along with value of the MAC. The protocol delivers/retrieves the shares $[s]_i, \gamma_j$, for $i, j \in \{1, \cdots, n\}$ to/from the servers. The value of the MAC γ is stored by the trusted appliance.
- Protocol $Reconstruction([s]_i, \gamma'_i)$ takes as inputs the shares of the master key and the MAC. The protocol executed by the trusted appliance outputs the master key s. We note the protocol recover the MAC from its sharing and compares it with the public value γ stored by trusted appliance before reconstructing the master key

3 The proposed protocol

In this section we describe the main 3 sub protocols of our cloud key management system $cloud_{KMS}$, namely SetUp, Sharing and Reconstruction.

We denote the number of servers by n and the security parameter (master 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.

3.1 SetUp protocol

To initiate the process, a user through the on-premise key management system denoted O_{KMS} generates the master key k. Executed by an application on behalf of the key management system on premise, it takes k.The protocol outputs the finite field F_p, the public shares $[s]_i$ of k along with its MAC (γ) and the public shares of the MAC γ'_i for $i = 1, \cdots, n$. The algorithm 1 depicts the implementation of the protocol.

Algorithm 1 Sub protocol Set_{up}

function SETUP(O_{KMS}, A)
  Compute the MAC $\gamma \leftarrow MACk$
  Sample random number $a_1, \cdots, a_n \in Z_p$ and $b_1, \cdots, b_n \in Z_p$
  Set $a_0 \leftarrow s$ and $b_0 \leftarrow \gamma$
  Set $q(x) \leftarrow \sum_{i=0}^{i=t} a_i x^i$ and $p(x) \leftarrow \sum_{i=0}^{i=t} b_i x^i$
  Compute $[s]_i \leftarrow p(i)$ and $\gamma'_i \leftarrow q(i)$ for $i = 1, \cdots, n$
  Send to appliance A: γ and $([s]_i, \gamma'_i)$ for $i = 1, \cdots, n$

3.2 Sharing

This protocol is executed with SetUp and Reconstruction protocols. It takes shares of the master key $[s_i]$ key and MAC γ along with value of the MAC γ'_i. The protocol delivers/retrieves the shares $[s]_i, \gamma_j$, for $i, j \in \{1, \cdots, n\}$ to/from the servers. The value of the MAC γ is stored by the trusted appliance. (Algorithm 2).

Algorithm 2 Sub protocol *Sharing*

function SHARING(A, S)
  Store the MAC γ
  Send each servers S_i: $([s]_i, \gamma'_i)$ for $i = 1, \cdots, n$

3.3 Reconstruction

Takes as inputs the shares of the master key and the value of γ'_1. The protocol executed by the trusted appliance outputs the master key s. We note the protocol recover the MAC from its sharing (γ'_i) and compares it with the public value γ stored by trusted appliance before reconstructing the master key.

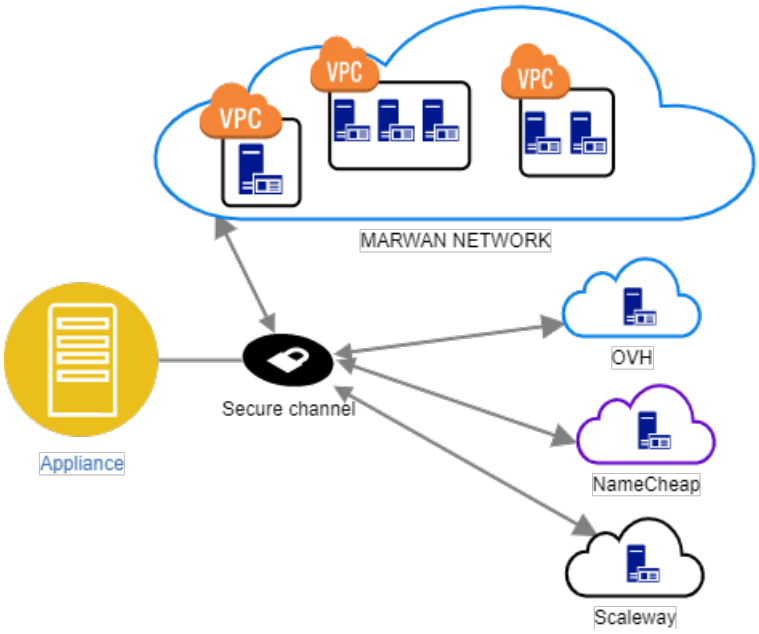
Algorithm 3 Sub protocol *Reconstruction*

function POST(S, A)
  number_share Integer $\leftarrow 1$
  Compute $L_i \leftarrow \prod_{j \neq i}(\frac{x_j}{x_j - x_i})$
  Compute $s \leftarrow q(0) \leftarrow \sum_{i=1}^{i=t+1} L_i[s]_i$
  Compute $\gamma = p(0) \leftarrow \sum_{i=1}^{i=t+1} L_i[\gamma']_i$
  if $\gamma == q(0)$ **then**
    compute k
  else
    PickAnotherShare number_share $\leftarrow 0$
  if *number_share* $== 0$ **then**
    Print : A share has been compromised

4 Security Analysis and Performance Evaluation

Figure 1 shows a presentation of how tests were conducted and where the shares were stored/retrieved.

Fig. 1. General Schema



We implemented our 3 protocols with Java 1.8 using a 64 bits Windows operating system with i7-8565 (1.8 GHz) processor and 16Go 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, RSA-1024, RSA-2048 and RSA-4096) while the t and n of shamir secret sharing were $t = 3$ and $n = 9$.

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 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 1. SetUp and Share protocols performances

Key Size	SSS Computation	Upload	Total	Latency Upload
AES-128	0.005	18	18.005	± 0.8
AES-192	0.005	18	18.005	± 0.8
AES-256	0.005	18	18.005	± 0.8
RSA-1024	0.03	18	18.03	± 0.8
RSA-2048	0.03	18	18.03	± 0.8
RSA-4096	0.03	18	18.03	± 0.8

Table 2. Share and Reconstruction protocols performances

Key Size	SSS Computation	Download	Total	Latency Download
AES-128	0.04	7.86	7.9	± 0.43
AES-192	0.04	7.86	7.9	± 0.43
AES-256	0.04	7.86	7.9	± 0.43
RSA-1024	0.07	7.86	7.93	± 0.43
RSA-2048	0.07	7.86	7.93	± 0.43
RSA-4096	0.07	7.86	7.93	± 0.43

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 (Figures 2 and 3). The latency time depends mainly on network quality for file transfer and also the time taken by the appliance to authenticate to servers.

It is obvious that the proposed protocol adds a new security layer for the secret key confidentiality. Based on our proposed design security threats may be a :

- Malicious insider user who may attempt to gather information through side channel attacks

Fig. 2. SetUp and Share Time

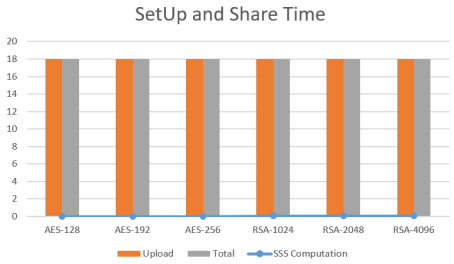
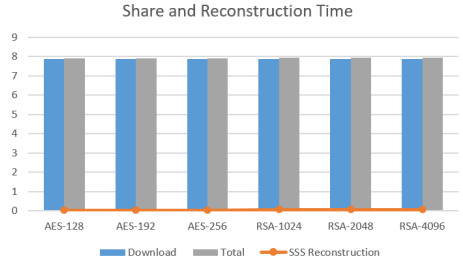


Fig. 3. Share and Reconstruction Time



– Malicious external attacker who may try to intercept communication and steal the shares.

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 k_i 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-premise 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 it self. 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.

5 Conclusion and future work

In this paper, we proposed a secure cloud key management based on the robust secret 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 through the use of 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 in order to improve the security of the secret.

References

1. IDG, "IDG Cloud Computing Survey", IDG (2020). https://www.idg.com/tools-for-marketers/2016-idg-enterprise-cloud-computing-survey/
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 IBE-based 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.
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 Cloud Computing, 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: 10.1109/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 Multisignature 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.

Authors

Ahmed Bentajer received his M.S. degree in 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.

Mustapha Hedabou received his M. Sc degree in Mathematics from the university of Paul Sabatier, Toulouse, France. In 2006, he received his Ph.D degree in computer science from INSA de Toulouse, France. He was a professor at ENSA of Safi, from Cadi Ayyad University Marrakech in Morroco, And now he is Associate Professor at Mohammed VI Polytechnic University, Benguerir, Morocco. His area interest covers Information Security, Public Key Cryptography based on Elliptic Curves, Identity based cryptography and cloud computing.

Sara Ennaama Sara ENNAAMA completed her Master's Degree in Business Intelligence and Big Data Analytics from Chouaib Doukhali University in 2020. Before that, she got her Bachelor's degree in Mathematical and Computer Sciences from

Cadi Ayyad University. She is currently pursuing a PhD in Computer Science at Abdelmalek Essaadi University and is passionate about cryptography, cloud storage, secure deletion and cloud computing.

Abderrahim Tahiri Obtained the Engineer degree in Computer Sciences in 2000 from Abdelmalek Essaädi University (UAE) in Morocco and the Master degree in Telematics Engineering in 2007 from Polytechnic University of Cartagena (UPCT) in Spain, and the PhD degree in Computer Sciences from UAE and UPCT in 2009. Currently he is professor at the National School of Applied Sciences in the UAE, he is a full member of the Computer Sciences Engineering Department, specialized in Internet object models and applications and a full member of Information System and Software Engineering Laboratory of UAE. His research interests include software architecture integration and smart application models. He has cooperated in, and coordinated several projects on national level and on European level. His research output includes 35+ co-authored articles. He has been chair of multiple conference tracks related to Information System Engineering and Wireless Sensor Network.