

## Data Sharing Over Owner & Data User by Using Kac & Hmac

Kuchi Aswini & Mr.G.Srinivasa Rao

<sup>1</sup>MCA , Sri Vasavi engineering college , Pedatadepalli, Tadepalligudem, West Godavari Andhra Pradesh, Mail Id:- [aswini.kuchi123@gmail.com](mailto:aswini.kuchi123@gmail.com)

<sup>2</sup>Associate professor, Dept. of CSE , Sri Vasavi engineering college , Pedatadepalli, Tadepalligudem, West Godavari, Andhra Pradesh, Mail Id:- [gsr.pdrl@gmail.com](mailto:gsr.pdrl@gmail.com)

### Abstract

*Online information sharing for expanded profitability and effectiveness is one of the essential necessities today for any association. The coming of distributed computing has pushed the points of confinement of sharing crosswise over topographical limits, and has empowered a large number of clients to contribute and work together on shared information. Be that as it may, ensuring on the web information is basic to the accomplishment of the cloud, which prompts the prerequisite of productive and secure cryptographic plans for the same. Information proprietors would in a perfect world need to store their information/documents online in a scrambled way, and delegate decoding rights for some of these to clients, while holding the ability to renounce access anytime of time. A proficient arrangement in such manner would be one that enables clients to unscramble different classes of information utilizing a solitary key of*

*consistent size that can be effectively communicated to numerous clients. Chu et al. proposed a key total cryptosystem (KAC) in 2014 to address this issue, though without formal evidences of security. In this paper, we propose CPA(chosen plaintext attacks) and CCA(chosen ciphertext attacks) secure KAC developments that are proficiently implementable utilizing elliptic bends and are reasonable for usage on cloud based information sharing situations. We lay uncommon spotlight on how the independent KAC plan can be productively joined with communicated encryption to take into account  $m$  information clients and  $m_0$  information proprietors while decreasing the diminishing the safe channel necessity from  $O(mm')$  in the standalone case to  $O(m + m')$ . In this framework we save just the security with Broadcast Encryption. However, we found that we have Data Integrity issue which implies information proprietor is the uprightness of their*

*outsourced records since they never again physically have their information and in this manner lose the control over their information. Also, the cloud server isn't completely trusted and it isn't compulsory for the cloud server to report information misfortune occurrences. So that subsequently, it is vital for implies information proprietors to every now and again check if their outsourced information are put away appropriately. So To ensure the respectability of the document, the information proprietor processes the Hash-based Message Authentication Code (HMAC) signature on each scrambled record.*

**Keywords:** - File Encryption, Data Owner, Data user, authentication

## 1. INTRODUCTION

The current approach of distributed computing has pushed the breaking points of information sharing capacities for various applications that rise above land limits and include a great many clients. [5]Governments and organizations today regard information sharing as a fundamental instrument for upgraded profitability. Distributed computing has changed instruction, medicinal services and long range informal communication. [1]Maybe

the most energizing use case for distributed computing is its capacity to permit numerous clients over the globe offer and trade information, while sparing the throbs of manual information trades, and maintaining a strategic distance from the production of repetitive or obsolete reports. [6]Person to person communication locales have utilized the cloud to make a more associated world where individuals can share an [3]assortment of information including content and interactive media. Community apparatuses ordinarily upheld by cloud stages and are to a great degree famous since they prompt enhanced efficiency and synchronization of exertion. [4]The effect of distributed computing has likewise invaded the circle of medicinal services, with PDA applications that permit remote observing and even finding of patients. To put it plainly, distributed computing is changing different parts of our lives in exceptional ways.

## 2. RELATED WORK

### Existing System

The current framework does not unequivocally address the issue of total key appropriation among numerous clients. In a handy information offering condition to a large number of clients, it is neither viable



nor effective to rely upon the presence of devoted coordinated secure channels for key appropriation. An open key based answer for broadcasting the total key among a subjectively expansive number of clients is thus alluring.

### **Proposed System**

In this Proposed System, we build up a novel component for open key based total key conveyance that decreases the protected channel necessity. We utilize communicate encryption, which is a notable strategy openly key cryptography, to proficiently appropriate the total keys among numerous clients in a safe design. Our broadened KAC development consolidates the essential KAC example gave people in general key based communicate encryption framework to construct a completely open key based online information sharing plan.

## **3. IMPLEMENTATION**

### **New User**

In this capacity information proprietor and client can be enlist with their data as perform approvals. Once on the off chance that they enlisted effectively then they can turn into a validated of this framework. Nobody can enroll with same email and f information proprietor id, client id, however in the event that any one attempt to enter the

copy passages then the framework ready message to clients.

### **Login**

In this capacity after enrollment by client or information proprietor, they can login with their qualifications and this accreditations ought to be substantial then no one but they can login into framework generally the framework can produce alarm message like Incorrect certifications. So in this framework validate clients just can login.

### **Document Encryption**

In this module after login as information proprietor, he can choose a document to transfer. Before transfer information proprietor can scramble the document information utilizing tidy key and ace key and so forth. So the record is put away as encoded information in database, so anyone doesn't see the information without information proprietor share the document to client.

### **Offer**

In this module information proprietor can share his records to specific client, before begin this module information proprietor transfer no less than one document generally this module cannot get to.

### **Shared Files**

In this module after login as client, he can see every mutual record list. Client can see the record name and encoded information. Subsequent to getting keys client can decode the information at that point see and download the record.

### Information Owner

Enroll as information proprietor to fill the points of interest as prerequisite then login as information proprietor. After login as information proprietor he can scramble the record information utilizing demure key and ace key and after that transfer the document into database. Proprietor can share his documents to specific clients to see and download his records. Those clients just access to see and download the records.

### Client:

In this module client can enroll then login to get to his module. After login as client, he see his subtle elements in profile data page email and portable no and so on. Client see the every single shared record at that point decode the document utilizing the proprietor shared keys. After decode the record information client can see and download the document.

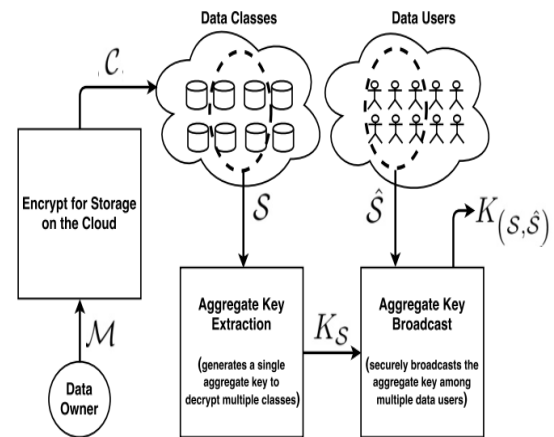


Fig:1 System Architecture

### 4. ALGORITHM EXPLANATION:

The framework for extended KAC with aggregate key broadcast is presented below:

- 1) **SetUp ( $1\lambda; n; m$ ):** Takes as input the number of data classes  $n$ , the number of users  $m$  and the security parameter  $\lambda$ . Outputs the public parameter  $param$ .
- 2) **OwnerKeyGen():** Outputs the public key  $PK$ , the master-secret key  $msk$  and the broadcast secret key  $bsk$  for a data owner registering in the system.
- 3) **OwnerEncrypt ( $param; PK; i; M$ ):** Takes as input a data class  $i \in \{1, \dots, n\}$  and the plaintext data  $M$ . Outputs a partially encrypted ciphertext  $C_0$ . Note that  $C_0$  is not the final ciphertext and is not exposed to the outside world. It is sent to the system administrator via a secure channel for further modification as described next. Note here that any instantiation of this scheme

must ensure that the partial ciphertext  $C_0$  is protected using suitable randomizations so as to leak nothing about the underlying plaintext data  $M$  during transmission to the system administrator.

**4) SystemEncrypt ( $C^*$ ;  $msk$ ;  $bsk$ ):** Takes as input the partially encrypted ciphertext  $C_0$ , the master secret key  $msk$  and the broadcast secret key  $bsk$ . Outputs the final ciphertext  $C$  which is made available on the cloud. This step is carried out by the system administrator, who is a trusted third party.

**5) UserKeyGen ( $param$ ;  $msk$ ;  $i$ ):** Takes as input the data user  $id \wedge i \in \{1, \dots, mg\}$  and outputs the corresponding secret key  $d^i$ .

**6) Extract ( $param$ ;  $msk$ ;  $S$ ):** Takes as input the master secret key  $msk$  and a subset of data classes  $S \subseteq \{1, \dots, ng\}$ . Computes the aggregate key  $KS$  for all encrypted messages belonging to these subsets of classes and passes it as input to the Broadcast algorithm for generating the broadcast aggregate key.

**7) Broadcast ( $param$ ;  $KS$ ;  $S^*$ ;  $PK$ ;  $bsk$ ):** Takes as input the aggregate key  $KS$  and the target subset of users  $S \subseteq \{1, \dots, mg\}$ . Outputs a single broadcast aggregate key  $K(S; S^*)$  that allows any user  $\wedge i \in S^*$  to decrypt all encrypted data/messages classified into any class  $i \in S$ .

**8) Decrypt ( $param$ ;  $C$ ;  $K(S; S^*)$ ;  $i$ ;  $\wedge i$ ;  $d^i$ ;  $S$ ;  $S^*$ ):** The decryption algorithm now takes, besides the ciphertext  $C$  and the corresponding data class  $i \in S$ , a valid user  $id \wedge i \in S^*$ . It also takes as input the broadcast aggregate key  $K(S; S^*)$  and the secret key  $d^i$ . The algorithm outputs the decrypted message.

### Algorithm (HMAC)

Table 1: The HMAC Algorithm

STEPS	STEP-BY-STEP DESCRIPTION
Step 1	If the length of $K = B$ : set $K_0 = K$ . Go to step 4.
Step 2	If the length of $K > B$ : hash $K$ to obtain an $L$ byte string, then append $(B-L)$ zeros to create a $B$ -byte string $K_0$ (i.e., $K_0 = H(K) \parallel 00\dots00$ ). Go to step 4.
Step 3	If the length of $K < B$ : append zeros to the end of $K$ to create a $B$ -byte string $K_0$ (e.g., if $K$ is 20 bytes in length and $B = 64$ , then $K$ will be appended with 44 zero bytes $0x00$ ).
Step 4	Exclusive-Or $K_0$ with $ipad$ to produce a $B$ -byte string: $K_0 \oplus ipad$ .
Step 5	Append the stream of data 'text' to the string resulting from step 4: $(K_0 \oplus ipad) \parallel text$ .
Step 6	Apply $H$ to the stream generated in step 5: $H((K_0 \oplus ipad) \parallel text)$ .
Step 7	Exclusive-Or $K_0$ with $opad$ : $K_0 \oplus opad$ .
Step 8	Append the result from step 6 to step 7: $(K_0 \oplus opad) \parallel H((K_0 \oplus ipad) \parallel text)$ .
Step 9	Apply $H$ to the result from step 8: $H((K_0 \oplus opad) \parallel H((K_0 \oplus ipad) \parallel text))$ .
Step 10	Select the leftmost $t$ bytes of the result of step 9 as the MAC.

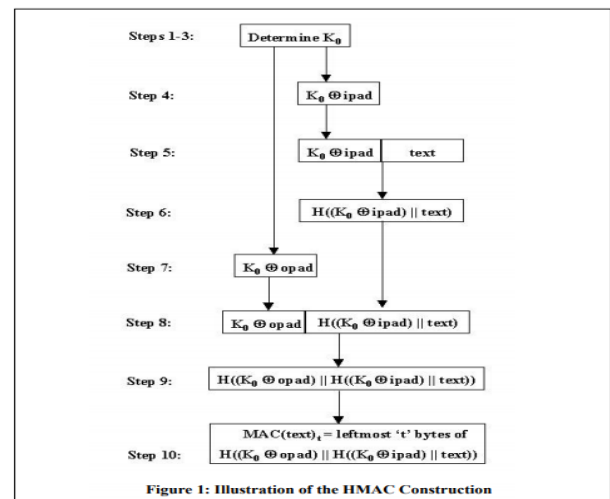


Figure 1: Illustration of the HMAC Construction

## 5. EXPERIMENTAL RESULT

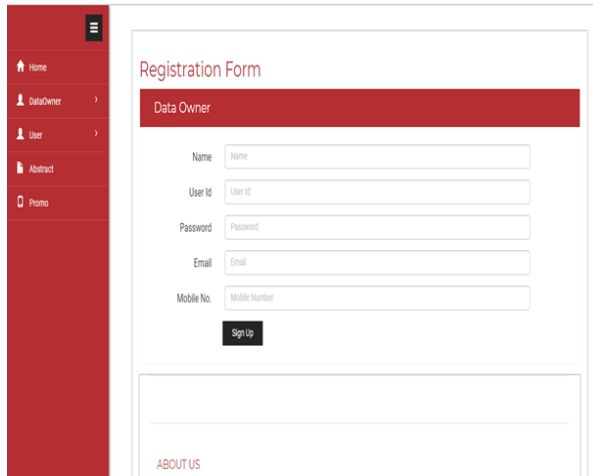


Fig:2 Authentication User has to register with this application to login

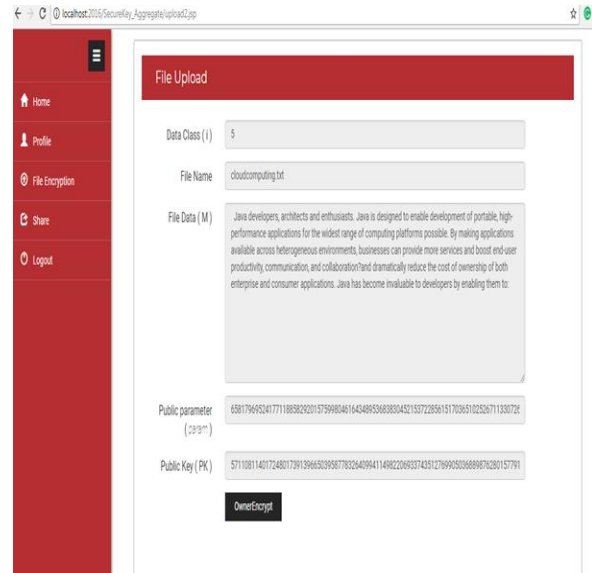


Fig:3 Key Generation Getting a key to encrypt the file data.

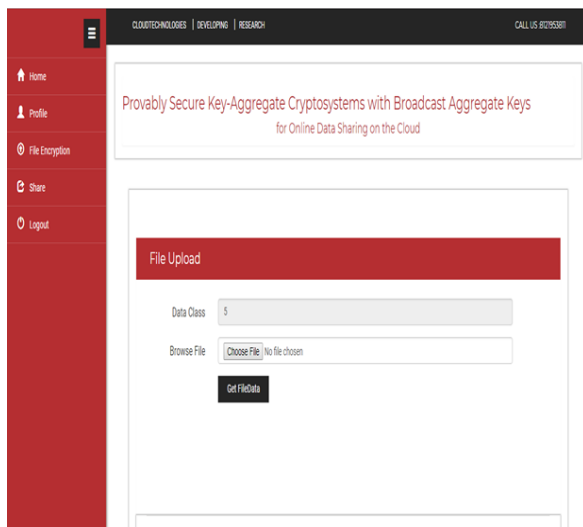


Fig:2 File Upload Upload text, java, css etc file.

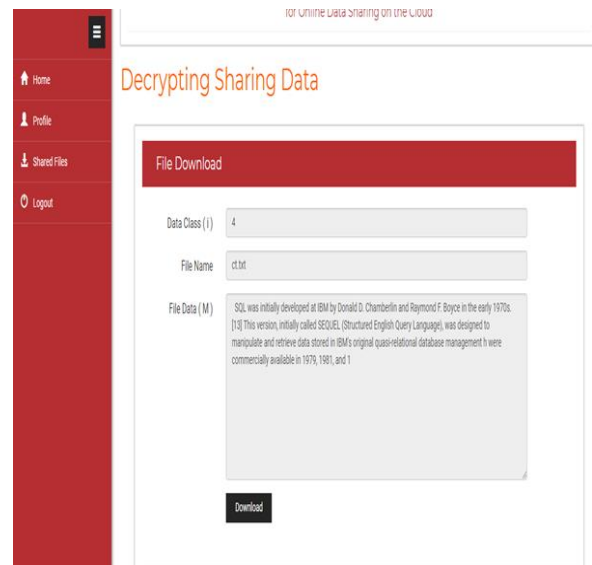


Fig:4 File Download Save files from local repository.

## 6. CONCLUSION

In this paper, we have proposed a productively implementable form of the essential key-total cryptosystem (KAC) in [6] with low overhead ciphertexts and total

keys, utilizing hilter kilter bilinear pairings. Our development fills in as a proficient answer for a few information sharing applications on the cloud, including communitarian information sharing, item permit circulation and medicinal information sharing. We have ended up being completely plot safe and semantically secure against a non-versatile enemy under suitable security suppositions. We have then shown how this development might be changed to accomplish CCA-secure development, which is, to the best of our insight, the main CCA secure KAC development in the cryptographic writing. We have additionally shown how the fundamental KAC system might be proficiently expanded and summed up for safely communicating the total key among different information clients in a genuine information sharing condition. This gives an essential pathway in planning a versatile completely open key based online information sharing plan for substantial scale arrangement on the cloud. We have displayed reproduction results to approve the space and time intricacy prerequisites for our plan. The outcomes set up that KAC with total key communicate beats other existing secure information sharing plans regarding execution and versatility.

## 7. REFERENCES

- [1] IDC Enterprise Panel. It cloud services user survey, pt. 3: What users want from cloud services providers, august 2008.
- [2] Sherman SM Chow, Yi-Jun He, Lucas CK Hui, and Siu Ming Yiu. Spice-simple privacy-preserving identity-management for cloud environment. In Applied Cryptography and Network Security, pages 526–543. Springer, 2012.
- [3] Cong Wang, Sherman S.-M.Chow, Qian Wang, KuiRen, and Wenjing Lou.Privacy-preserving public auditing for secure cloud storage.Cryptology ePrint Archive, Report 2009/579, 2009. <http://eprint.iacr.org/>.
- [4] Sherman SM Chow, Cheng-Kang Chu, Xinyi Huang, Jianying Zhou, and Robert H Deng. Dynamic secure cloud storage with provenance. In Cryptography and Security: From Theory to Applications, pages 442–464. Springer, 2012.
- [5] Erik C Shallman. Up in the air: Clarifying cloud storage protections.Intell. Prop. L. Bull., 19:49, 2014.
- [6] Cheng-Kang Chu, Sherman SM Chow, Wen-GueyTzeng, Jianying Zhou, and Robert H Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. Parallel and Distributed



Systems, IEEE Transactions on, 25(2):468–477, 2014.

[7] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Advances in Cryptology–CRYPTO 2005, pages 258–275. Springer, 2005.

[8] Selim G Akl and Peter D Taylor. Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems (TOCS), 1(3):239–248, 1983.

[9] Gerald C Chick and Stafford E Tavares. Flexible access control with master keys. In Advances in Cryptology CRYPTO89 Proceedings, pages 316–322. Springer, 1990.

[10] Wen-Guey Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. Knowledge and Data Engineering, IEEE Transactions on, 14(1):182–188, 2002.