A Survey on Assured deletion and Access Control

Vaishak sundaresh[1] Asst Professor
Dept of CSE, Acharya Institute of Technology,
vaishaksundaresh@acharya.ac.in

Bharath M N[2] PG Student
Dept of CSE, Acharya Institute of Technology,
bharath.mtcs.12@acharya.ac.in

Abstract: Cloud storage is emerging business model for outsourcing the data backup off site to third party cloud storage services So as to reduce the management cost. However, security guarantee must be there for outsourced data, maintained by third party. This paper summarizes the security issues such as assured deletion and access control for outsourced data in cloud storage, which is maintained by the third party and also provide the analysis of Perlman's concept, Vanish system for Assured deletion, for access control Wang's approach and Attribute based Encryption for Access control.

Keywords: Cloud storage, Security, Assured deletion, Access Control, Backup.

I.INTRODUCTION

Cloud storage is a new business solution for remote backup outsourcing, as it offers pay-as-you-go-manner. Security concerns become more relevant as we now outsource the storage of possibly sensitive data to third parties. Data on cloud storage will not be permanently stored because it is undesirable, as data may be unexpectedly disclosed in the future due to malicious attacks on cloud. This challenge can be achieved by assured deletion. In this paper, we outline some of the security issues of the cloud are Access Control, which allows authorized user to access the outsourced data on cloud and Assured deletion, in which outsourced data is inaccessible to everyone including data owner, upon the user request. The security concerns motivate us to have a system that can enforce access control and assured deletion in fine-grained manner.

II.PERLMAN'S CONCEPT

Perlman's concept [1], provides the high availability of data only for predefined amount of time, once predefined time is completed, automatically data will be assuredly deleted. Perlman introduces the three methods of assured deletion which are time-based file, Individual file deletion by On-demand and On-demand deletion of custom classes.

Time based file: Each file is associated with expiration time. Expiration time is declared during creation of file. An example for this form of assured deletion is contract based employee. Once the employee completes contract period, all information related to the employee will be expunged.

Individual file deleted by On-demand: Here, files can be assuredly deleted upon the demand request. This form of deletion is very dangerous because files can be deleted at any point of time. *On-demand deletion of custom class:* Class is set of files, which can be encrypted with custom keys and policy can be applied for that key at some point of time. For custom keys, ephemerizer needs to hold the keys for each class of file. Here, ephemerizer is treated as a key manager to create, advertise and maintaining the ephemeral public keys. Ephemerizer is reliable because multiple ephemerizer are used in file system with set of independent key but it does not keep backup of keys. Rather than making the ephemerizer robust by having it make *n* copies of its keys , robustness is achieved instead by the file system ,by having the file system use *n* independent ephemerizers, any [2] of whom can unlock the encrypted data .

The main idea here is, the file is encrypted with data key by the owner of the file and then data key is further encrypted with control key by separate key manager (also known as ephemerizer [1]). The encrypted keys are managed by the key manager, which is server. In [1], the control key is time based key maintained by the key manager until the completion of expiration time. Once it reaches expiration time then it is automatically removed by key manager, where the expiration time is specified when the file declared. Without the control key, the data key and data file remain encrypted and reckoned to be inaccessible. Thus, main security property of the file assured deletion is that even if a cloud provider doesn't remove expired files from its storage, those files remains encrypted and unrecoverable. This concept provides the high-availability of data until timeout . The issue in [1] is that it is uncertain whether time-based file assured deletion is feasible in practice, as there is no empirical evaluation.

III .VANISH SYSTEM

Vanish system (also known as Self-destructing [3]) is a feature of email system. This system assures that the copy of email at the client side will be assuredly deleted after reading. This system allows the user to determine the life span of their personal data stored in web such as private messages on facebook, documents on Google Docs, or private photo on Flicker by making the web object self-destruct or vanish automatically.

With the help of Vanish Data Object (VDO) target vanish system is achieved. A VDO encapsulates the user's data and prevent its contents from persisting indefinitely and becoming a source of retroactive information leakage .Regardless of whether the VDO is copied, transmitted or stored in internet, it becomes unreadable after a predefined period of time even if any attacker retroactively obtains the pristine copy of the VDO from before its expiration, and all of the user's past persistent cryptographic keys and passwords. VDO abstraction and vanish system make several key assumptions. Each VDO contains the Access Key, Cipher text, Key shares and threshold specified by the user. Life time of the each VDO will be limited period, that must be known to user. Once, VDO reaches the predefined time then VDO will automatically vanish.
During the interaction with VDO's internet connection is needed. Rather than risk exposure to an adversary, the user prefers the VDO object to be destroyed, even if prematurely. Functional goals and Properties of Vanish system are as follows

Destruction after timeout: when the VDO reaches the expiration time, it must automatically vanish, without any action from the explicit users. Once VDO is expired, it is not accessible to anyone.

Accessible until timeout: VDO must be accessible to authorized user, during its lifetime.

Leverage existing infrastructures: The system must leverage existing infrastructure. It must not rely on external .special-purpose dedicated services.

No secure hardware: System doesn't require use of secure hardware.

No new privacy risks: No new privacy risks are introduced to the users.

The main idea of Vanish system is, vanish takes the data Object D, and encapsulates it into a VDO V. Vanish system generate the a random data key K ,which is used to encrypt the D ,called as cipher text C. Data key K is divided into N shares (pieces) i.e., K_1 , $K_{2\,,}$ . . .K_N by using the threshold secret sharing[2] . Threshold can be used as parameter to secret sharing and which set by the user. The threshold determines how many of the N shares are required to construct the original key. The vanish has computed the key shares it picks at random an access Key, L. It then uses a cryptographic secure random number generator [4], Key by L, to derive N indices into the, DHT. Now, V = (L, C, N, threshold) and is sent over email server or stored in the file system. Vanish system perform decryption of V by using following steps:

(1) Extract the Access Key L,
(2) Derives the location of the shares of K,
(3) Retrieves the required number of shares as specified by the threshold,
(4) Reconstruct K,
(5) Decrypt C to obtain D.

Vanish system is feature of email system ,which is helps to protect the privacy of past, archived data from accidental, malicious and legal attacks . This system assures that copy of email at the client will be deleted after reading. The open issues in [3], it supports time-based assured deletion and VDO can access anyone.

IV.WANG'S APPROACH

Wang suggested an approach [5] which is used to achieve flexible access control and large-scale dynamic data management in high secure and efficient. Wang proposed to use data block level symmetric encryption but this mechanism doesn't require any encryption algorithm and also provide the description on handling dynamics in user access rights and data block.

Data blocks are encrypted with key which is determined by using key structure, which shown in Figure 1. Data block is simple smallest information access units. It provides the fine-grained access control by encrypting each block with different keys without using any encrypted algorithm. Suppose, if data contains *n* data blocks and each data block is encrypted with randomly generated secret key K_i (i=1 to n) , but there will storage overhead on the owner . Meanwhile, when the user needs to access the particular data block *i* then there will be communication overhead between the data owner and user for key distribution will also be linear to *i*. So, it requires efficient key management technique. Wang uses efficient key management technique called key derivation method [6]. Basic idea of generating key for encryption of data blocks using hierarchy. Derivation procedure uses the one way function to generate the key in hierarchy by combining the parent node and some public information, can't calculate the secret keys of the parent key and sibling nodes .

The root nodes of hierarchies are maintained by data owner. Data owner can share the secrets to end user based on user access rights i.e. only for the user is authorized, during the key distribution. Data blocks are decrypted by end users by deriving the leaf nodes in hierarchy. Assume that outsourced data contains *n* blocks and $2^{h-1} \leq n \leq 2^h$, so can construct the binary tree with height h. Data owner will choose the root secret as K(0,1) where first s index 0 , indicate the level in the hierarchy , second index 1 , indicate its sequence in the level .Generally , for level *l* then the sequence number are from 1 to 2*l* . Data owner chooses a public hash function Hash() . For any node K(i,j) in the hierarchy , its parent node is K(i-1,j/2) (where i > 0) and its children are K(i+1, 2*j-1) and K(i+1, 2*j) where K(i,j) is not leaf node . In hierarchy, Left child of K(i,j) can be calculated as follows

$$K(i+1 , 2*j-1) = Hash(K(i,j) \| (2* j-1) \| K(i,j))$$

Right child of K(i,j) can be calculated as follows :

$$K(i+1,2*j) = Hash(K(i,j) \| (2*j) \| K (i,j))$$

By applying this function repeatedly, a node can calculate the secrets of all of its descendents.

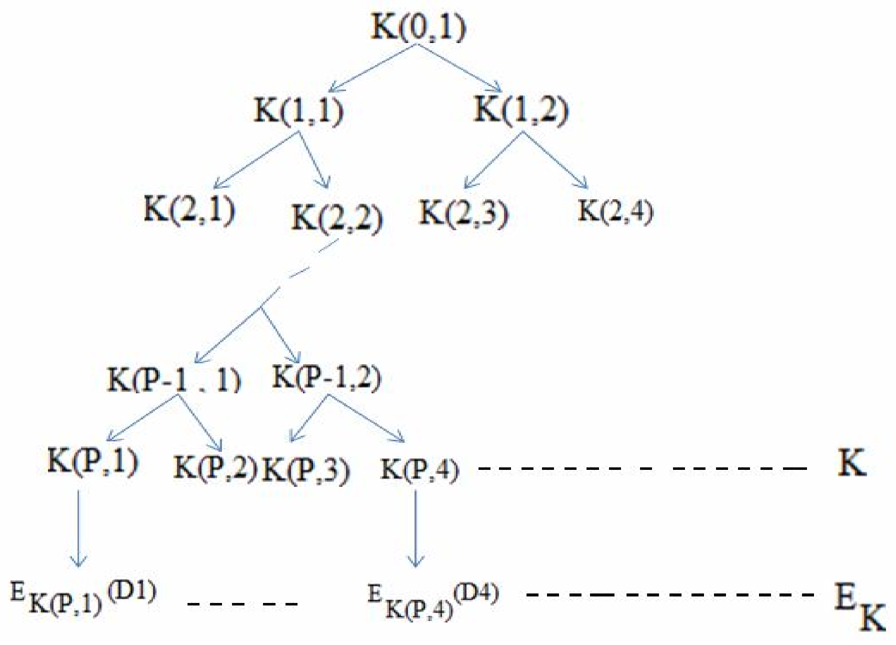

Figure 1: Key derivation hierarchy

Data Access Procedure

when service provider sends the data block to end users , assume that the service provider performed over-encryption [7],on data block .To perform this operation , assume that the service provider and end user share a pseudo random number bit sequence generator P() [8,9]. Given seed P() can generate a long sequence of pseudo random bits.

Notations are *O* represents the data owner, *S* represents the service provider and *U* represents the end users. Assume that *O* shares the pair wise keys K_{OU} and K_{OS} with *U* and *S* respectively.

1. *U* will send a data access request to *O*

$$U \rightarrow O: \{U, O, E_{KOU}(U, O, \textit{request index}, \textit{data block index} , \textit{MAC code})\}$$

Pair wise key K_{OU}, is known to *U* and *O*. O will be able to authenticate the sender. The *request index* will be increased by 1 every time *U* sends out a request and it is used by *O* .The request contains the index numbers of the block that *U* wants to access. The *MAC code* will protect the integrity of the data.

2. When *O* receives this message, it will authenticate the sender and verify the integrity and freshness of the request. It will then examine it access control matrix and make sure that *U* is authorized to read all blocks in the request. If the request passes this checks, the owner will determine the smallest set of key K^{-1} in the hierarchy such that (1) K^{-1} can derive the keys that are used to encrypt the requested data blocks and (2) *U* is authorized to know all keys that can be derived from K^{-1} .The owner will be then generate the reply to the end user.

$$O \rightarrow U: \{O, U, E_{KOU}(O, U, \textit{request index}, \textit{ACM index seed for P()}, K^{-1}, \textit{certificate for S} ,\textit{MAC code}) \}$$

The *request index* is used to uniquely label this reply. The *ACM index* is used by *O* to label the freshness of the Access control Matrix. This index will be increased by 1every time *O* changes some end user's access rights. The Updated *ACM index* will be sent to *S* by *O* to prevent those revoked users from using old certificate to access data blocks. The seed is a random number to initiate P() so that *U* can decrypt the over-encryption conducted by *S* . *U* will use K^{-1} to derive the data block encryption keys. The certificate for the service provider and it has the following format:

{E_{KOS} (*U, request index, ACM index, seed, indexes of data blocks, MAC code*)}

3. The *U* will send {*U, S, request index, certificate}* to the service provider. When *S* receives this packet, it can verify that the *certificate* is generated by *O* since only they know the secret $K_{OS.}$ *S* will make sure that the user name and request index in *certificate* is smaller than the value that S receives from *O* , some changes to the access control matrix have happened and *S* will notify *U* to get new *certificate*. Otherwise, service provider will retrieve the encrypted data blocks and perform the over-encryption as follows. Using *seed* as the initial state of P(), the function will generate a long sequence of pseudorandom bits , *S* will use this bit sequence as one-time pad and conduct the XOR operation to the encrypted blocks. The computational results will be sent to *U*.

4. When *U* receives the data blocks, it will use *seed* to generate the pseudo random bit sequence an use K^{-1} to derive the encryption keys. It will then recover the data block.

The Wang's approach [5] achieves flexible access control with less client responsibilities, less storage overhead and support block deletion, update, insertion and appending. Open issues are, requires support from the cloud side and no multiple policies combination.

V. ATTRIBUTE BASED ENCRYPTION FOR ACCESS CONTROL

The Basic concept of ABE system by Sahai and Waters [10] , in which a user's key and cipher text are labeled with sets of descriptive attributes and a particular key can decrypt a particular cipher text only if attributes of the cipher text is same as attributes of the user's key . This approach [9] , introduces new cryptosystem called Key-policy Attribute based encryption(KP-ABE) , which provides fine-grained access control of encrypted data . The main idea of KP-ABE is cipher text are labeled with sets of attributes and private keys are associated with access permissions which control cipher text a user is able to decrypt.

In KP-ABE [9], which each cipher text is labeled with set of attributes and private key, which is associated with access structure that specifies which type of cipher text the Key can decrypt. Access structure is specified in the private key, while cipher texts are simply labeled with a set of descriptive attributes. Each user's key is associated with a tree-structure where the leaves are associated with attributes. A user is able to decrypt a cipher text if the attributes associated with cipher text satisfy the key's access structure. The new technique is used to implement the fine grained access control, in which the data is stored on server in an encrypted form while different users are still allowed to decrypt different pieces of data per the security policy. So, effectively eliminates the need to rely on the storage server for preventing unauthorized data access .KP-ABE consists of four algorithms, which are as follows:

1. *Setup:* This is randomized algorithm that takes no input other than implicitly security parameter .It outputs the public parameter PK and master Key MK. Define the universe of attributes *U*={ 1,2,. . .n} . For each attribute i Є *U* choose a number t_i uniformly from Z_p. Finally, Choose *y* uniformly at random in Z_p. The public parameters PK are

$$T_1 = g^{t\,1} \ldots\ldots T_{|U|} = g^{t\,|U|},\ Y = e(g\ ,g)^y\ .$$

The Master Key MK is

$$t_1 ,\ldots\ldots t_{|U|}\ .\ y.$$

2. *Encryption*: This is randomized algorithm, which takes the input as message m, a set of attributes ϒ, and the public parameter PK. It outputs a decryption key D. To Encrypt a Message M Є G_2 under a set of attributes ϒ choose a random value s Є Z_p and cipher text as:

$$E = (\Upsilon, E' = MY^{s,}\ \{\ E_i = T_i^s\}_{i\in\Upsilon}\ )$$

3. *Key Generation*: This is randomized algorithm that takes as input, an access structure A, the master key MK , and the public parameter PK. It outputs a decryption key D. This algorithm outputs a key that enables the user to decrypt a message encrypted under a set of attributes ϒ if and only if T (ϒ) = 1. This algorithm proceeds as follows .First choose a polynomial q_x for each node *x* in the tree Ƭ. These polynomial are chosen in the following way in top-down manner starting from the root node r . For each node *x* in the tree ,set the degree d_x of the polynomial q_x to be one less than the threshold value K_x of that node that is $d_x = K_x - 1$. Now, for the root node *r*, set $q_r\ (0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node *x*, set $q_x\ (0) = q_{parent(x)}\ (index(x))$ and chose d_x other points randomly to completely define q_x.

Once the polynomial has been decided, for each leaf node *x*, secret value to the user is calculated as follows:

$$D_x = g^h$$

$$Where\ h = q_x(0)\ /\ t_i\ and\ i = att(x)\ .$$

4. *Decryption*: This algorithm takes as input the cipher text E, that was encrypted under the set ϒ of attributes, the decryption key D for access control structure A and the public parameter PK . It outputs the message M if ϒ Є A. Decryption procedure as recursive algorithm. DecryptNode*(E ,D ,x)* that takes as input the cipher text $E = (\Upsilon\ , E'\ , \{E_i\}_{i\in\Upsilon})$, the private key *D* and a node *x* in the tree . It outputs a group elements of G_2 or $\perp$. Let $i = att(x)$. If a node *x* is leaf node then, if i Є ϒ then

$$\text{DecryptNode}(E,D\ ,x) = e\ (D_x\ , E_i\ )=e(\ g^h\ , g^b\ )=e\ (g\ .\ g)^{\ k}$$

$$Where\ h = q_x(0)\ /\ t_i$$

$$b = s.t_i$$
$$k = s\ .q_x\,(0).$$

Otherwise DecryptNode$(E\ ,D,\ x) = \perp$

VI. CONCLUSION

Perlman achieves the targeted objective by providing high availability of data for predefined period of time and some of the issues are that it is uncertain that time based file assured deletion in cloud, but no empirical evaluation has been done. Vanish system is feature of email system. This system assures that email is sent to the user, after reading it will be vanished. The open issue is that it supports time-based assured deletion and VDO can be accessed by anyone. The Wang's approach, achieves flexible access control with less clients responsibilities, less storage overhead and support block deletion, update, insertion and appending.

The issues here are that it requires support from the cloud side and no multiple policies combination is not supported. In Attribute based encryption, new cryptosystem has been introduced called Key-Policy Attribute based encryption, which provides fine-grained access control of encrypted data. The main idea of this paper is to give comprehensive information about security issues in cloud storage such as assured deletion and Access control .The motto here is to help the novice researchers with objective to work on security issues of cloud storage.

REFERENCES

[1] R.Perlman, “*File system Design with Assured deletion*", In *ISOC, NDSS*, 2007.
[2] A. Shamir, "*How to share secret* ", *CACM, Vol 22*, Nov 1979.
[3] R. Geambasu, T .Kohno, A. Levy and H. M Levy, " *Vanish: Increasing data privacy with self- destructing data* ", In Proc of *USENIX Security symp*, Aug 2009.
[4] M. Blum and S. Micali, "*How to generate cryptographic strong sequences of pseudo-random bits*", In Proceedings of the 23rd *IEEE Symposium on foundation of Computer Science (FOCS ' 82)*, 1982.
[5] Weichao Wang , Zhiwei Li , Rodney Owens , Bharat Bhargava , " *Secure and Efficient access to Outsourced data* " , *CCSW '09* , Nov 13, 2009 .
[6] M. J Atallah , M Blanton , N Fazio and K. B Frikken , " *Dynamic and Efficient Key Management for Access Hierarchy* " , *ACM Trans , Inf.Syst Secure*, 12(3):1-43 , 2009.
[7] S. D .C di Vimercati, S. Foresti, S. Jaiodia, S. Paraboschi and P .Samarati ,"*Over-Encryption : Management of access control evolution on Outsourced data* " , In Proceedings of the *international conference on very large databases*, Pages 123-134 , 2007 .s
[8] A. Chow, W. Coates and D Hopkins, "*A Configurable asynchronous pseudorandom bit sequence generator* ", In *IEEE International Symposium on Asynchronous Circuits and systems*, Pages 143-152, 2007.
[9] P. Li, Z. Li, W. A Halang and G. Chen, "*A Multiple pseudorandom bit generator based on a spatiotemporal chaotic map*", Physics Letter A, 349(6), 467-473, 2006.
[10] A Sahai and B .Waters , "*Fuzzy Identity Based Encryption* ", In *Advances in Cryptology , Euro crypt*, volume 3494 of LNCS , Page 457-473 ,Springer ,2005

ACKNOWLEDGEMENT

We sincerely acknowledge all the cited authors for giving us base paper with strong fundamentals concepts. We would like to thank all the members of Department of Computer Science and Engineering , Acharya Institute of Technology, Bangalore for providing the valuable advises and special thanks to all our friends who supported directly or indirectly for better structuring of the paper .

BIOGRAPHY

1] Mr. Vaishak Sundaresh, currently working as Assistant Professor, Department of Computer science & Engineering, Acharya Institute of Technology, Bangalore, Karnataka, India.

2] Mr. Bharath M N, currently pursuing Master of Technology in Computer science & Engineering, Acharya Institute of Technology , Bangalore , Karnataka , India.