IBE based file deletion in cloud storage

Safae Elamrani[1],Imane EL Abid[1],Meryem Janati Idrissi[1],
Ali Ouahhabi[1],Yunusa Simpa Abdulsalam[1]Mustapha Hedabou[2*]

November 12, 2018

1. Mohammed VI Polytechnic University Ben Guerir, Morocco 2. Professor at ENSA Safi, Morocco *

Abstract

Instead of using local storage ,individuals and companies tend to save their files and information in remote data servers called cloud .Using a cloud storage provides the costumer with scalable and high capacity resources but it's also important to insure the security of sensitive files shared on the cloud. In this paper we propose a solution to the security issue in the cloud by protecting deleted data with policy based file deletion. In fact we propose a new design of FADE using an IBE based cryptosystem to make deleted files unrecoverable after the revocation of the policy. To demonstrate the performance of the solution proposed,we we implement it with dropbox API .we then draw a brief comparison between this new solution and FADE system.

Introduction

Cloud storage is becoming more common. Companies and individuals have been increasing their use of cloud services as Google Drive,Dropbox,Amazon. Thus,securing the cloud storage is now a main concern of specialists seeking for the confidentiality and integrity of the data hosted by the remote cloud severs(which are third parties that data owners relay on for saving sensitive data. Data in the cloud is nearly always stored in an encrypted form,but the keys of this encryption are held in different emplacements depending on each cloud storage service,thus,protecting those keys is a big security issue. In fact,when a file deletion request is sent to the cloud ,we can't guarantee that the cloud storage provider have removed the file completely because there may have be multiple file backup copies distributed on a certain number of servers to ensure the reliability of the service. In this paper,we present an IBE based system to guarantee the secure deletion of users files from the cloud.The solution works atop of cloud storage services. In fact,the data storage is assured by the cloud while the keys are generated by a trusted entity called the PKG . In brief,the contributions of our paper can be described as follow: -We proposed a new design of FADE based on IBE by developing algorithms for achieving file Download and Upload from and to the cloud storage. -We implemented a prototype of the proposed design atop Dropbox cloud storage. -We gave a brief study of security and efficiency of the proposed scheme. -We evaluated the performance of the new design atop dropbox comparing with FADE.

IBE based Fade design architecture:

To give a deeper insight about the FADE based on IBE design that we propose in this paper,we must start by a detailed description of the Download/Upload functionalities.Our design is based on IBE which stands for Identity based encryption. The main concept behind IBE based FADE system is that the public and private key used to encrypt and decrypt the secret key of the cloud user are generated by a trusted entity called PKG based on the identity of the user(its e-mail address for instance). Each file could have a set of policies. Let's assume now that every file has one policy named DATE that specifies the date of expiration of the file .In other words ,it means that the file will be unrecoverable after this date ,even by the user itself. In the design of our system we make sure to guarantee that the cloud storage provider could not read the file even if it has multiple distributed copies of it in different backup servers.

Upload

When a user wants to upload a file F to the cloud he connects to the PKG first. The communication between the user and the PKG is secured using SSL sockets. The PkG authenticates the user with his ID and password and then generates a pair of public and private $PubK_u$, $PriK_u$ for him based on his ID the policy(after verifying the validity of the policy) of the file he wants to upload. The data owner generates a secret key S_K and encrypts the file F using it (in our case we applied the RC4 encryption algorithm for it's rapidity since it uses cipher stream encryption) .The resulting encrypted file is called $\{F\}_{S_K}$. In the next step ,the secret key S_K is also encrypted with the public key generated by the PKG $\{S_K\}_{PubK_u}$ by applying Boneh–Franklin encryption algorithm. Finally the the policy P ,the encrypted key $\{S_K\}_{PubK_u}$ are all sent with the encrypted file $\{F\}_{S_K}$ to the cloud storage .Once the upload completed,the data owner delete all the keys from its memory.

Download

On the other side ,when a user wants to download a file F. He started also by sending its ID to the PKG to complete the authentication step. After that he requests the private key from the PKG by sending the policy of F as along with its own ID .the PKG then sends the private key back (generated based on the User ID ,the file policy and the master private key after verifying the validity of the policy).After receiving the private key ,the user now can decrypt K Kprivate(using Boneh–Franklin decryption algorithm) to get the secret key K. At this stage ,the FK can also be decrypted with RC4 knowing the secret key K. the figure below gives a graphical illustration of all the steps proposed in the design.

Deletion
By implementig the proposed design we provide a guarantee that the file F will be deleted from the cloud definitely when the policy is revoked (the Date policy in our example) .In fact after receiving the policy of the file from the use ,The PKG verifies the policy ,in our case if the date specified in the policy has been expired (which corresponds to policy based deletion of the file),the PKG will not send the Keypair to the user and in this way nor the user neither the cloud storage cold decrypt and read the file since they don't have the private key .

Implementation of the design atop Dropbox cloud storage

Our implementation of the design is atop Dropbox cloud storage.The application uses Dropox API to insure the following functionalities : Upload(file,policy): after encrypting the file F and the secret key,the secret key and the policy are then put together in a file F* and we send a zipped file composed with the files F and F*. The keys are then deleted from the data owner memory. Download(file): to download d a file ,the user receive the zipped file containing the File F along with the key and the policy.he then extracted the policy and send it to the PKG for getting the private key.In this stage ,the user could decrypt the file F. In case the policy of the file has been revoked ,the PKG doesn't send the private key to the user. -But before uploading or downloading a file a main functionality is executed: Authentication: The data owner is authenticated by the PKG server.we used then SSL sockets to insure develop this functionality.

Brief study of security and efficiency of the proposed design

This IBE based design of FADE Insure the integrity and the confidentiality of the sent files as well as the verification of the user identity.In fact the File F is encrypted with a public key K.and The Key K is also encrypted with private key (using Boneh–Franklin decryption algorithm). Even if an eavesdropper intercept the encrypted file ,he couldn't get the plaintext nor the key K to decrypt it. The cloud storage providers could not read the file too because the private key is only given to the user by the PKG after verifying its identity as well as the file policy. After the expiration or revocation of the policy (which corresponds to a policy based deletion of the file) neither the data owner nor the cloud storage provider could read the file even if the cloud provider could have multiple companies of the file in distributed servers since the PKG won't send the private key . The communication between the user and the PKG is also secured and authenticated using SSL sockets communication.

Evaluation

In this section we draw a quick comparison between the performances of the upload and download time with plaintext sending and our design[see figure 4 in tables section]. Figure 4 illustrate that there isn't a big latency in time even if the design proposed uses two encryptions which is supposed to slow the download and upload functions.

Figures and tables

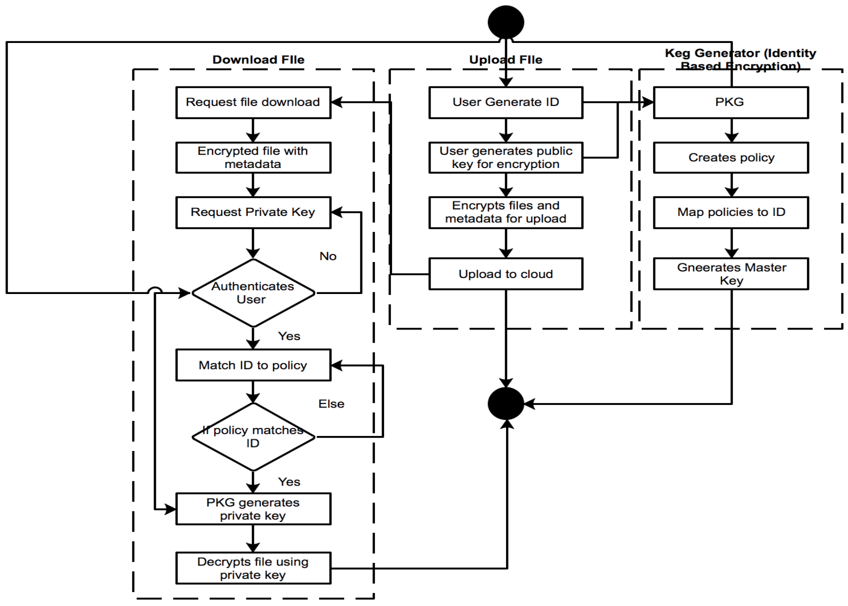


Figure 1: functioning

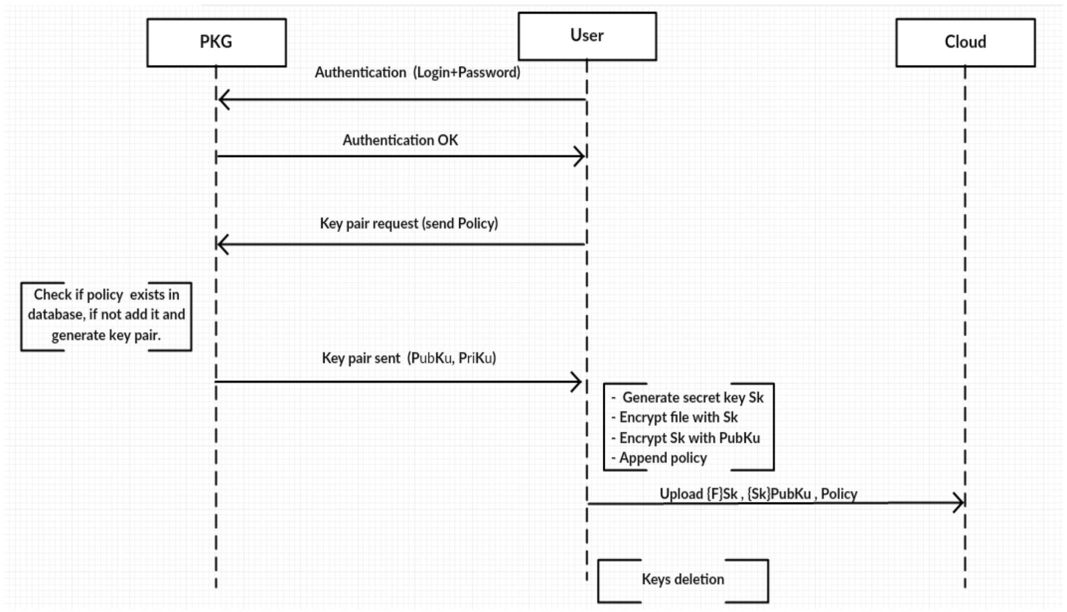

PKG
User
Cloud
Authentication (Login+Password)
Authentication OK
Key pair request (send Policy)
Check if policy exists in database, if not add it and generate key pair.
Key pair sent (PubKu, PriKu)
- Generate secret key Sk
- Encrypt file with Sk
- Encrypt Sk with PubKu
- Append policy
Upload {F}Sk , {Sk}PubKu , Policy
Keys deletion

Figure 2: Data upload

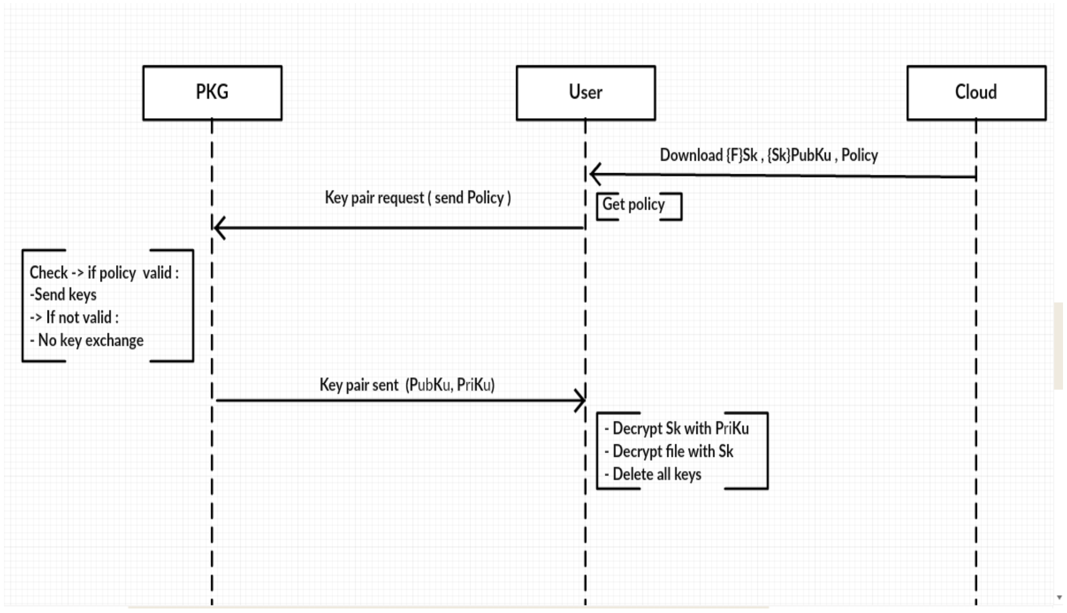
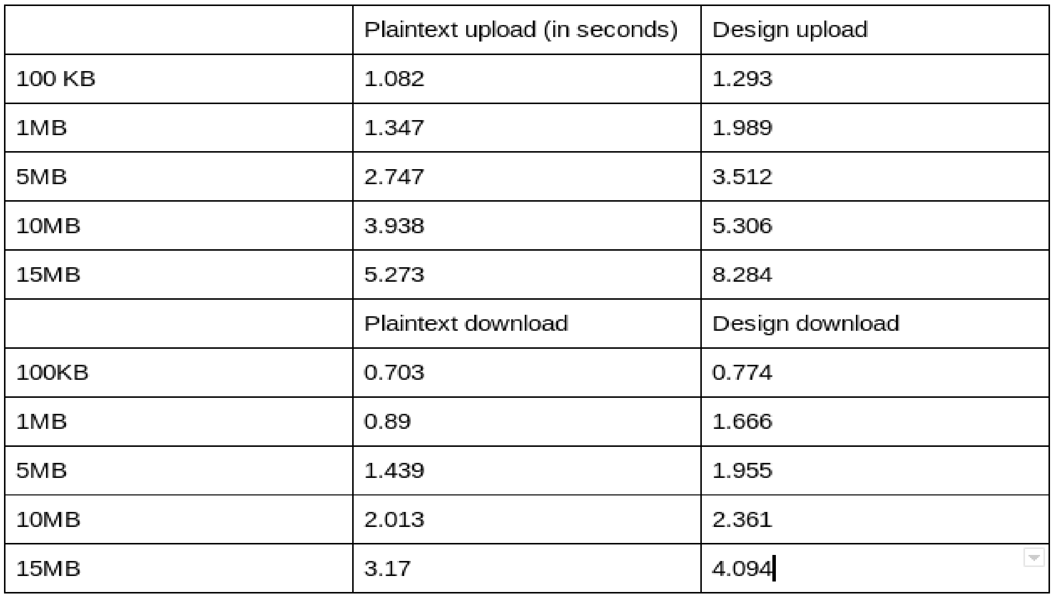


Figure 3: Data download



	Plaintext upload (in seconds)	Design upload
100 KB	1.082	1.293
1MB	1.347	1.989
5MB	2.747	3.512
10MB	3.938	5.306
15MB	5.273	8.284
	Plaintext download	Design download
100KB	0.703	0.774
1MB	0.89	1.666
5MB	1.439	1.955
10MB	2.013	2.361
15MB	3.17	4.094

Figure 4: comparison between FADE and our design