UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Vitor Hugo Galhardo Moia

A Study about the Security and Privacy on Cloud Data Storage

Um Estudo sobre a Segurança e Privacidade no Armazenamento de Dados em Nuvens

CAMPINAS

2016

Vitor Hugo Galhardo Moia

A Study about the Security and Privacy on Cloud Data Storage

Um Estudo sobre a Segurança e Privacidade no Armazenamento de Dados em Nuvens

Dissertation presented to the School of Electrical and Computer Engineering, University of Campinas, in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de Computação.

Supervisor: Prof. Dr. Marco Aurélio Amaral Henriques

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Vitor Hugo Galhardo Moia, e orientada pelo Prof. Dr. Marco Aurélio Amaral Henriques

CAMPINAS

2016

Agência(s) de fomento e nº(s) de processo(s): CNPq, 153392/2014-2

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

M727s

Moia, Vitor Hugo Galhardo, 1990-
A study about the security and privacy on cloud data storage / Vitor Hugo Galhardo Moia. – Campinas, SP : [s.n.], 2016.

Orientador: Marco Aurélio Amaral Henriques.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Criptografia. 2. Privacidade. 3. Usabilidade. 4. Computação em nuvem. 5. Computação em nuvem - Medidas de segurança. I. Henriques, Marco Aurélio Amaral,1963-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Um estudo sobre a segurança e privacidade no armazenamento de dados em nuvens
Palavras-chave em inglês:
Cryptography
Privacy
Usability
Cloud computing
Cloud Computing - Security measures
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Marco Aurélio Amaral Henriques [Orientador]
Ricardo Dahab
Christian Rodolfo E. Rothenberg
Data de defesa: 16-02-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Vitor Hugo Galhardo Moia RA: 153843
Data da Defesa: 16 de fevereiro de 2016
Título da Tese: A Study about the Security and Privacy on Cloud Data Storage
Título da Tese em outro idioma: Um Estudo sobre a Segurança e Privacidade no Armazenamento de Dados em Nuvens

Prof. Dr. Marco Aurélio Amaral Henriques (Presidente, FEEC/UNICAMP)
Prof. Dr. Ricardo Dahab (IC/UNICAMP) - Membro Titular
Prof. Dr. Christian Rodolfo E. Rothenberg (FEEC/UNICAMP) - Membro Titular

Ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

I dedicate this master dissertation to my family and many friends. A special feeling of gratitude to my loving parents, Marcelo and Maria Lúcia whose support was fundamental in this journey, as well as for my sister Bianca and my brother Vinicius who have never left my side. I also dedicate it to my relatives and my goddaughter Lara, for always believing in me during these two years.

I also dedicate this work to God, for giving me this life-changing opportunity and also the strength to overcome all the obstacles found in this journey.

Finally, I dedicate this work to my friends from Andradas and also the ones I had the pleasure to meet during these two years studying at Unicamp, specially the ones from the DCA department.

Acknowledgements

I would like to thank my supervisor Professor Marco Aurélio Amaral Henriques, who has guided and helped me during the program. I owe him many thanks for all his patience, support and dedication.

I also would like to thank the comments on this dissertation from the examination team: Prof. Christian Rodolfo E. Rothenberg and Prof. Ricardo Dahab. Thank you for taking the time to read this work and for sharing your experience with me.

I am thankful to all Professors of FEEC who have shared their knowledge with me during the classes.

I would like to acknowledge and thank my school division for allowing me to conduct my research and providing any assistance requested.

I am also grateful for the partial financial support from CNPq (grant number 153392/2014-2).

I am very thankful for all the support that my friends and relatives have given me during these two years. Thanks for always being on my side.

Finally, I want to thank my parents for always believing in me and supporting each decision I have made.

“There is no such thing as privacy anymore; there is secrecy”
Rich Hersh

Abstract

Cloud data storage is a service that brings several advantages for its users. However, in public cloud systems, the risks involved in the outsourcing of data storage can be a barrier to the adoption of this service by those concerned with privacy. Several cloud service providers that claim to protect user's data do not fulfill some requirements considered essential in a secure, reliable and easy to use service, raising questions about the effective security obtained. We present here a study related to user's privacy and data security requirements on public clouds. The study presents some techniques normally used to fulfill those requirements, along with an analysis of their relative costs and benefits. Moreover, it makes an evaluation of them in several public cloud systems. After comparing those systems, we propose a set of requirements and present a proof of concept application based on them, which improves data security and user privacy in public clouds. We show that it is possible to protect cloud stored data against third party (including cloud administrators) access without burdening the user with complex security protocols or procedures, making the public cloud storage service a more reliable choice to privacy concerned users.

Keywords: Public Cloud Storage, Security Requirements, Privacy, Cryptography, Usability.

Resumo

Armazenamento de dados na nuvem é um serviço que traz diversas vantagens aos seus usuários. Contudo, em sistemas de nuvens públicas, os riscos envolvidos na terceirização do armazenamento de dados pode ser uma barreira para a adoção deste serviço por aqueles preocupados com sua privacidade. Vários provedores de serviços em nuvem que afirmam proteger os dados do usuário não atendem alguns requisitos considerados essenciais em um serviço seguro, confiável e de fácil utilização, levantando questionamentos sobre a segurança efetivamente obtida. Apresentamos neste trabalho um estudo relacionado aos requisitos de privacidade dos usuários e de segurança de seus dados em nuvens públicas. O estudo apresenta algumas técnicas normalmente usadas para atender tais requisitos, juntamente com uma análise de seus benefícios e custos relativos. Além disso, ele faz uma avaliação destes requisitos em vários sistemas de nuvens públicas. Depois de comparar estes sistemas, propomos um conjunto de requisitos e apresentamos, como prova de conceito, uma aplicação baseada nos mesmos, a qual melhora a segurança dos dados e a privacidade dos usuários. Nós mostramos que é possível proteger os dados armazenados nas nuvens contra o acesso por terceiros (incluindo os administradores das nuvens) sem sobrecarregar o usuário com protocolos ou procedimentos complexos de segurança, tornando o serviço de armazenamento em nuvens uma escolha mais confiável para usuários preocupados com sua privacidade.

Palavras-chaves: Armazenamento em Nuvens Públicas, Requisitos de Segurança, Privacidade, Criptografia, Usabilidade.

List of Figures

List of Tables

Contents

1 Introduction

Cloud computing is a well-known and consolidated technology nowadays, which can bring many advantages for users. It refers to a model that delivers services over the internet, allowing its customers to access them through any device with internet connection. Most companies understood the potential of this service and started moving their business to clouds. Users can demand computer resources in an elastic, scalable and configurable way, avoiding costs associated to infrastructure and local maintenance, paying only for what they consume (pay-as-you-go model).

There are several ways to provide cloud services, and the most known and used are: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). According to Mather T et al. (MATHER *et al.*, 2009), there are also a few deployment ways to implement cloud services by providers, which can be:

- Public: The cloud infrastructure is managed and belongs to a third party vendor, the Cloud Service Provider (CSP). It is placed outside the establishment, in a controlled area by the CSP. Users data is out of their control and are protected by someone they do not know and cannot trust (normally).
- Private: The cloud infrastructure is managed by and belongs to users. It is placed inside the organization, in an user controlled area. Data access is only granted to reliable parts.
- Hybrid: This model consists of two or more cloud deployment models. Part of the services could run in one infrastructure and the other part in another. For example, companies could run non sensitive applications in a public cloud and sensitive applications in a private infrastructure.

There are still more deployment models of cloud computing: Community cloud, Multi-cloud and Federated cloud. However, they will not be discussed here since they are beyond the scope of this study.

Among all types of services provided by CSPs, the cloud data storage is one of the most used and popular. Users can have access to their data from anywhere at any time, avoid costs of building and maintaining a storage infrastructure, pay only for what they use (the storage space they consume) and avoid the troubles related to backups.

However, after start using the cloud technology, users begin to wonder about the safety of their information. Gradually, they realized how valuable their privacy is, making it one of their main concerns in recent times. News related to attacks and threats surrounding users on the Internet made them more concerned about the lack of control on the data stored in public

clouds (ALLIANCE, 2013), where they do not know where their data is located, if their CSP is learning something about it, if there is any modification on it and so on. They just store their data in clear form, without any extra protection and rely on the security provided by their CSP's protocols. They must trust in these entities.

Besides, there is a risk related to the CSPs sharing users data with marketing companies or use them in a harmful way. Security breaches are also another problem faced in these environments, which allow intruders to get users data. Data life cycle is yet another concern, since its destruction is a complicated matter in the cloud environment, where data replicas are spread over several servers in order to achieve fault tolerance and better performance. These and other concerns create a barrier on those who care about their privacy and the security of their data.

Techniques that can minimize the potential damages caused by unwilling disclosure of sensitive data are necessary and cryptography is being used to this end. Some CSPs realized the potential risks on users privacy and started offering cryptography options on the data storage services to minimize users' concerns. However, providing cryptography services is not trivial. There are obstacles to overcome and sometimes they are just put away by providers who prefer easy solutions or prefer to focus on usability over security. Besides, other factors can increase or decrease the security level of a system, and must be considered in order to create a secure and reliable model, where users can get the desired privacy and security. For users, the main obstacle in this environment might be their lack of expertise on cryptography. They do not know exactly what it is, how this technique can protect them and the importance of managing well the cryptographic keys.

In this scenario, this work raises some problems and concerns related to the security and privacy of users when storing their data on public clouds. We discuss some techniques to mitigate these issues, along with an analysis of their costs and benefits. We also present some requirements considered indispensable for a secure, reliable and easy to use cloud system. Finally, we propose a new cryptographic application to mitigate the risks in public cloud data storage. This application, called Cloud Privacy Guard (CPG), addresses users' concerns about privacy while maintaining a good level of usability to avoid the burden of dealing with complex cryptographic protocols and procedures.

The remainder of the text is structured as follows:

- Chapter II: Presents some problems related to the security and privacy on cloud data services, focusing on data storage. It also discusses some major concerns and techniques used to mitigate them, and an analysis of academic solutions;
- Chapter III: Proposes a method for evaluating the relative costs and benefits of the techniques used to improve users' privacy;

- Chapter IV: Proposes a set of requirements that should be adopted by a CSP in order to improve the data security and the privacy offered to users. It also presents an analysis of existing CSPs according to these requirements;
- Chapter V: Proposes an application to give users more control over the security of their data without having to deal with complex cryptographic procedures;
- Chapter VI: Gives the conclusions and some directions for future works.

2 Main concerns on cloud data storage and some solutions

In this chapter we will discuss some threats in the cloud data storage environment to which stored data is exposed. We will also describe some concerns about the users' privacy in the cloud, considering a more wider approach from a privacy point of view. Some techniques used to address these concerns will also be presented and discussed. Finally, we will present and discuss academic solutions and architecture proposed to mitigate the current problems and to improve the security and privacy in cloud data storage.

2.1 Threats to cloud data storage

The main threats described below are subject of study by several researchers as (BESSANI *et al.*, 2013), (ASHKTORAB; TAGHIZADEH, 2012), (ZISSIS; LEKKAS, 2012), (SUBASHINI; KAVITHA, 2011), (KHALIL *et al.*, 2014) and (ZHOU *et al.*, 2010) and some of them appear in the list of major threats in the cloud environment appointed by the Cloud Security Alliance (CSA), an entity specialized in computing and security technologies for clouds (ALLIANCE, 2013) (HUBBARD; SUTTON, 2010).

- **Data leakage:** Most users store their data in the cloud without any additional protection, mainly trusting in the protocols adopted by their Cloud Service Provider (CSP). Data confidentiality is an important matter since critical, personal and confidential information could be obtained from certain files, and used to attack or blackmail users. If a server is compromised for some reason, the privacy of its users is compromised as well. Another concern related to confidentiality is the fact that users data may be monitored without their knowledge when stored in public clouds. Some CPSs could learn something from users' files, and do advertising based on that, for instance.

- **Unknown data physical location:** When users send their data to be stored in the cloud, they do not know where their files will be located. Some CSPs have their servers spread around the world, and will store users data according to their needs and to economic factors. Therefore, data could be stored in different countries, which may have different laws with respect to privacy. This could be harmful to users, since their data is in possession of a third party (the CSP) spread over different countries and government agencies could

easily get users data from the provider.

- **Lack of guarantee of total data destruction:** When users make a request to delete a certain file stored in the cloud, they do not have a guarantee that all copies of this file were indeed deleted by the provider in their servers. If a CSP has access to user data, it normally uses them for data mining, research and/or advertisement purposes and it will not be willing to remove immediately the data from its servers when the user deletes them. Therefore the user's data may stay available to the CSP much longer than the user expects. Backups are also a reason for keeping excluded data for a while, in cases where users want to recover them.

- **Insecure Applications:** Users data security also depends on the safety of the applications used with them. For this reason, before using any software developed by an unknown source, users have to certify that this software is really secure and reliable, doing indeed what it promises (ASHKTORAB; TAGHIZADEH, 2012).

- **Account or service hijacking:** Attackers will attempt to get access to users' credentials and hence to their accounts. The attack could be executed by brute force, phishing, fraud, exploiting software vulnerabilities or by other means. Usually, the reuse of credentials (login/password) by users makes this threat easier to be explored.

  Users' credentials can also be obtained by malicious codes running on the machine, as Trojans and Keyloggers, among others. Once attackers have access to users' account, they can have access to all their data, including those kept in the cloud. Even those that use cryptography to protect users data, but rely on the authentication password to protect all the stored information, will not be able to protect anything, in such case.

- **Inability to access data:** One of the major characteristics of cloud data storage service is data availability, where users can access their data any time from any place, just using a device with Internet connection. Usually, CSPs have technology to protect them against flaws and being out of service. However, it is inevitable that users data get unavailable due to Internet connection problems. There is also the possibility of outages or attacks, like denial-of-service (DOS) or similar that could interrupt data access (BESSANI *et al.*, 2013).

- **Data lock-in:** Most users rely just on a single CSP to store their data. As all CSPs offer redundancy in their systems, users usually do not worry about this issue and put all their information in just one place. However, they become dependent on their providers

in such case. This dependency, when related to data storage, is called data lock-in (ABU-LIBDEH *et al.*, 2010) and present several disadvantages for users, mostly the risk of a single point of failure. Once users send their files to be stored in the cloud, if their CSP raises the prices charged for their storage services, goes out of business or even reduces the provided service quality, it is too expensive to move all data to a new provider. Besides, they would have to download all their data and then upload everything in the new CSP. This becomes an issue when users have too many files stored. This mostly happens because there is no easy way to move data from one CSP to another directly, since there are no standards for APIs in the cloud environment, limiting the portability of data and application between CSPs. (SCHNJAKIN *et al.*, 2011).

- **Malicious insiders:** This is a big threat to users, where the attackers are the employees at the CSP. Former or current unhappy employees who seek ways to cause damage or to blackmail users or the CSP, are examples of such attackers. The great advantage they have over external attackers is the knowledge about the cloud's infrastructure and the privileged access to it.

- **Common shared infrastructure:** Sharing infrastructure is a common practice in cloud environments and most CSPs adopt this concept as a way to reduce costs. Although users are usually separated at a virtual level, hardware is not at most times. Multiple users will have their data placed in the same hardware in the cloud. Because of this, some risks for an user may appear if the account of another user is compromised and the separation among users is not well implemented at the software level (SUBASHINI; KAVITHA, 2011).

- **Data loss:** There is always a concern about data being lost or corrupted while they are being sent to the cloud, recovered or at rest in the CSP (BESSANI *et al.*, 2013).

- **Sniffer attacks:** Some applications are designed to capture packets travelling in the network. If users data is sent to the cloud in a unencrypted way, attackers using these applications will capture and have access to such data (ASHKTORAB; TAGHIZADEH, 2012).

It is important to highlight that some of these threats are only a concern when using a public infrastructure. In a private cloud environment, the threats faced are those from external sources. Problems as unknown data physical location, lack of guarantee of total data destruction, insecure applications, vendor lock-in and common shared infrastructure are not threats since in

a private cloud the data owner also owns the infrastructure that will hold the data. Table 1 summarizes all the problems discussed in this section and maps them according to the type of infrastructure.

Table 1 – Threats to cloud data storage mapped according to the type of infrastructure

Problems	**Public Cloud**	**Private Cloud**
Data leakage	X	X
Unknown data physical location	X	
Lack of guarantee of total data destruction	X	
Insecure Applications	X	
Account or service hijacking	X	X
Inability to access data	X	X
Data lock-in	X	
Malicious insiders	X	X
Common shared infrastructure	X	
Data loss/Leakage	X	X
Sniffer attacks	X	X

2.2 Privacy on cloud data storage

Nowadays, due to a lack of knowledge or to the complexity involved, most users just send their data to the cloud and trust to the CPS the protection of their information. As a way to avoid this almost blind trust, several researchers have proposed solutions to improve users' privacy in this environment. In this section, we will present some concerns related to users' privacy in the process of storing data in the cloud. We will detail each concern and discuss some techniques that can be used to mitigate it. In our work, we focused on four privacy characteristics (BIRRELL; SCHNEIDER, 2012) (PFITZMANN; HANSEN, 2010), which are:

1. Confidentiality: Capability of controlling who can access the data.
2. Undetectability: There is no way to tell if an item of interest exists or not.
3. Anonymity: Users can not be identified from data.
4. Pseudonymous: Users identification using different names from their real ones.

2.2.1 Data confidentiality

One of the greatest concerns related to the privacy in the cloud data storage environment is the data confidentiality, a very active research topic. This concern is related to the lack of control over data stored in the cloud by its owners. Also, data is usually stored in plaintext, with no additional protection on it, allowing the CSPs or intruders to learn something about it

or even to get some information related to its owner. Having control on who are able to have access to their data, users will not have to trust blindly in their providers and their data will be safer against attacks. The main techniques used to provide this control and give more safety to users are cryptography and data fragmentation. Both techniques are explained bellow, along with a discussion about their benefits and limitations.

2.2.1.1 Cryptography

Cryptography is a technique used to keep an information confidential and accessible only for those who know a secret (a key), used in the codification process. The secrecy is achieved by a codification based on the key, which will generate an unintelligible information. Despite being one of the most important techniques employed nowadays to keep information secret, there are some disadvantages, or difficulties, related to the use of cryptography. Some of the obstacles are the lower performance in general (extra time for encryption and decryption), the difficulty in sharing data with other users and the difficulty in finding and indexing encrypted data. Moreover, there is a complex key management process that the users will have to deal with.

Cryptography could be used in cloud data storage to protect data at rest or in transit. Regarding the protection of data at rest, data could be encrypted before going to the cloud (client side) or after (server side). With respect to the protection of data in transit, data can be encrypted when sent from users' device to the cloud or vice versa. A combination of this two models is also possible. Fig. 27 illustrates a case where data is encrypted before it is sent to the cloud.

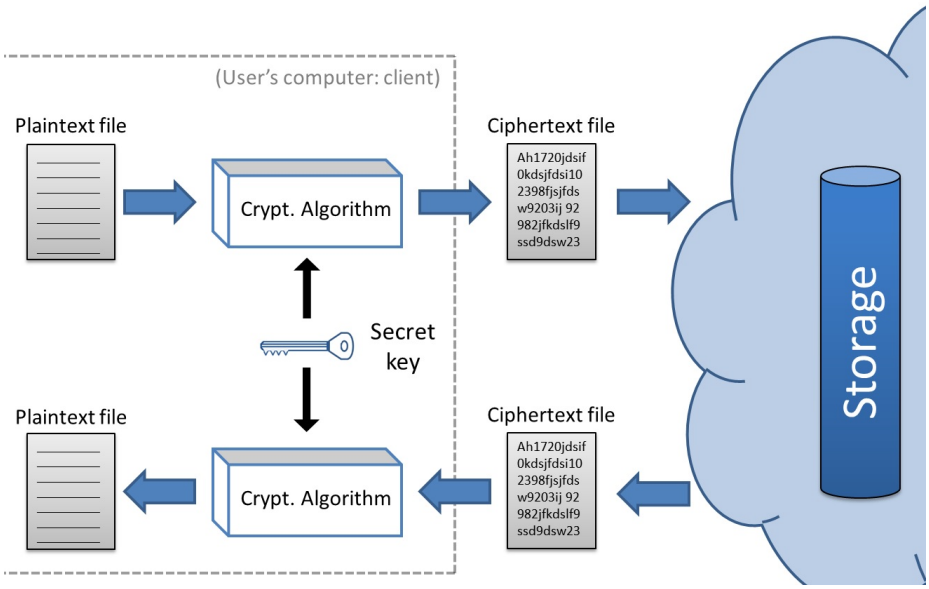


Figure 1 – Cryptographic process in the client

2.2.1.2 Data fragmentation

Other technique that could be used to minimize some concerns about users privacy in cloud data storage is data fragmentation. This concept is similar to the *RAID* (Redundant Array of Independent Disks) technology (PATTERSON *et al.*, 1988), but when it is used in the cloud environment, data is broken in many small fragments which are spread over several CSPs belonging to different companies. Normally, the main reason to use this technique is to provide fault tolerance and/or performance. But here, the objective of spreading data into different clouds is to avoid that CSPs or any attackers get the whole data in case of security breaches in some of them. To reconstruct the original data, normally all fragments are necessary, but, depending on the method used in the fragmentation process, only part of them is sufficient.

There are two ways to use data fragmentation: with or without redundancy. The fragmentation without redundancy consists in splitting the data into n parts, which are all necessary in order to recover the original information. There is no advantages in using this mode in cloud storage, since all data fragments will be necessary in the recovery process, and any failure in getting one of the them could preclude users from reconstructing the original data.

The other mode of data fragmentation is the adoption of a method that creates redundancy in the system. This is very important in cloud environment since it tolerates possible faults. There will be several clouds storing users data, and if one of them gets unavailable for any reason, this fault tolerant model allows users to recover their files using the remaining fragments. There are several methods to implement this technique. One of them is the FRS (Fragmentation-Redundancy-Scattering) (SILVA; RODRIGUES, 1998). This method is very costly in term of storage space, as data is divided and the parts duplicated and scattered in a way that satisfies users fault tolerance needs. Another method used is the Secret Sharing (SHAMIR, 1979), also very costly in terms of storage, since each fragment generated will have the size of the original file. However, this algorithm also provides secrecy in the parts itself, since each of them is a result from a special codification. This is one of the reasons why this method is popular, as most fragmentation algorithms create fragments which can revel some information about the original data (FU; SUN, 2012).

Another option is using a method called Erasure Codes (SCHNJAKIN *et al.*, 2013b), a family of codes used in the cloud data storage environment to solve the storage overload produced by the two previous methods. In particular when *Reed-Solomon* code is used, data is broken into k fragments of equal size that hold the original data and m additional parts of coding information, calculated from the k fragments, resulting in a total of $n = k + m$ parts. The original fragments can be recovered from any k fragments of n. The *Reed-Solomon* method increases the storage costs by a factor of m/k. Fig. 2 illustrates the process of fragmentation and storage in different clouds.

It is important to highlight that some factors can increase or decrease the level of

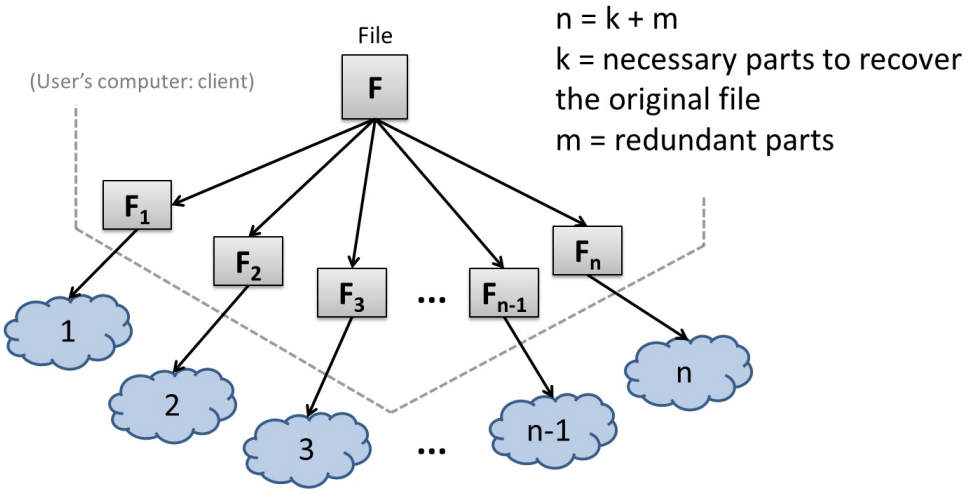


Figure 2 – Data fragmentation and storage in different clouds

privacy got from the use of this technique. One of these factors is geopolitics. When users spread their data into different CSPs whose servers are located in the same country or in geopolitically related ones, it is easier for governors or agencies to get access to all the data fragments (by means of warrants) than in cases where they are located in geopolitically conflicted countries. The latter case can increase users' privacy since one country will not collaborate with another due to their conflicts. On the other hand, users' privacy can decrease when countries have a good relationship and collaborate with each other, making it easier to get all data fragments.

2.2.2 Metadata confidentiality

The metadata confidentiality is also related to users' privacy, providing undetectability for the data stored. It deserves attention because some attributes, as filename, size, creation and modification dates etc, can compromise users' privacy, even if all possible precautions were taken with respect to the content protection using the techniques described above. Let us take the data denomination as an example. Usually, users name their files with content-related words. In this case, the file will be identified and this could be enough to cause some problems to its owner. A compromising situation could be where an user has a text file containing information about his income tax, for example, and the filename would be the word *tax* followed by his social security number and the file format. In order to keep the content of this file secret, he encrypts and stores it in the cloud. However, if the filename and extension are not encrypted, even though its content is, an attacker would have a hint to figure out what information is inside of that file. In the example above, user's social security number would also be revealed. The file format could also suggest something about it.

Another issue comes up when the fragmentation technique is used, since all fragmented data may be renamed as *file.part1, file.part2, ..., file.partN*, revealing that this file was fragmented in, at least, *N* parts. This situation should be avoided, because even though it is hard

to guess how many parts a certain file was broken into, the simple knowledge that the fragmentation occurred could lead to questions about where are those fragments or what is the total number of them.

The example above makes clear that only file content encryption is not enough. A secure way to protect the metadata in the cloud is necessary. A possible solution could be through the metadata encryption, as shown in the Fig. 3. The same key used to encrypt the data content could be applied to encrypt the metadata, without having an extra burden imposed by the use of a different key. This solution would prevent a possible disclosure of the data and add a barrier to attackers, who will have a hard time figuring out which files are important and worth stealing. The fragmentation problem would also be avoided, since the name of a fragment *file.part1* would be illegible and totally different from the name of other fragments, as *file.part2, file.part3, ..., file.partN*, due to the characteristics of cryptographic algorithms.

Another solution that could be used with cryptography is the division of a file into fragments of different sizes, which would improve its security once the parts would be more similar to random files in terms of their size.

Encrypting metadata can also be very useful in the context of Content-Centric Networking (CCN) (JACOBSON *et al.*, 2009).

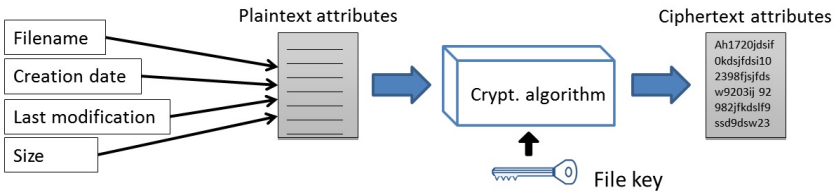


Figure 3 – Data name encryption process

2.2.3 Data access confidentiality

This concern is related to the users' privacy on accessing their data in the cloud. In this case, the privacy is about their location, which they wish to keep in secret from the CSPs (anonymity). There are many resources to this end as, for example, the TOR (MURDOCH; DANEZIS, 2005) network and VPN-Proxy (DUFFIELD *et al.*, 2005). These technologies can protect users from surveillance of governments, service providers that sells statistic data about these accesses to marketing companies and also against possible attackers. Also, some Internet web services use access location to provide specific and customized services to its users. However, these users usually do not have control over these services; tools to mask their location could protect them and offer a higher privacy. In the next subsections, we will explain how TOR and VPN-Proxy can provide anonymity to its users.

2.2.3.1 TOR

TOR (The Onion Router) is an open source project that offers protection against surveillance, and once installed, can mask users' location while surfing on the Internet. This benefit is achieved through a complex mechanism that uses data transmission over multiples machines. According to Murdoch and Danezis (MURDOCH; DANEZIS, 2005), there are several TOR nodes on the network and each of them tries to ensure that the corresponding incoming and outgoing flows are obscured to an attacker. The initiator of the flow creates a circuit through a connection to random TOR nodes, negotiating secret keys and setting up a secure channel between them. The following communication is done through this secure channel, and for each new TOR node connection, new secret keys are exchanged in order to protect the information by multiples encryption layers.

A message is encrypted using all the encryption keys established among the nodes and forwarded to each node, in the reverse order that the encryption occurred. Each node will receive the encrypted message, and it only has to decrypt its corresponding encryption layer and forward the message to the next node. After a series of steps (three by default), the last node connects to a port in the destination node and delivers the message. As the secret keys are erased after the circuit ends, the nodes can not decrypt old messages, and also they do not know who started the communication, since they only know who sent them the message and who it should be delivered to, providing the anonymity to users. All of this is executed on client-side and do not need a trusted third party to get the anonymity in network traffic.

Fig. 4 illustrates a basic communication using TOR. A circuit is created with three TOR nodes, and after the message leaves the origin, it goes through the three selected nodes, where it is decrypted and then delivered to the next. In the third and last node, the message is decrypted and sent to the destination. All messages that should be delivered to the initial node do the opposite way back. The destination send a message to the final node, which encrypts it and send it back to the second TOR node. The message is encrypted and sent to the first one, responsible for putting the third encryption layer and forwarding it to the initial node. After receiving the message, this node decrypts it with the secret keys belonging to each node, and then reads the message.

2.2.3.2 VPN-Proxy

Another tool that can be used to protect users' location is a VPN (Virtual Private Network) service combined with a proxy server. The use of this technology creates a communication tunnel (with the use of cryptographic protocols, such as TLS) between users' device and a VPN-Proxy server (VPNBook.com, frootvpn.com, hide.me, torvpn.com and freevpn.me for example). The main function of this server is to receive, decrypt and forward users' packets to the cloud, but using as the source address its own IP in place of the original one, providing IP (localization) anonymity (DUFFIELD *et al.*, 2005). Although all communication between

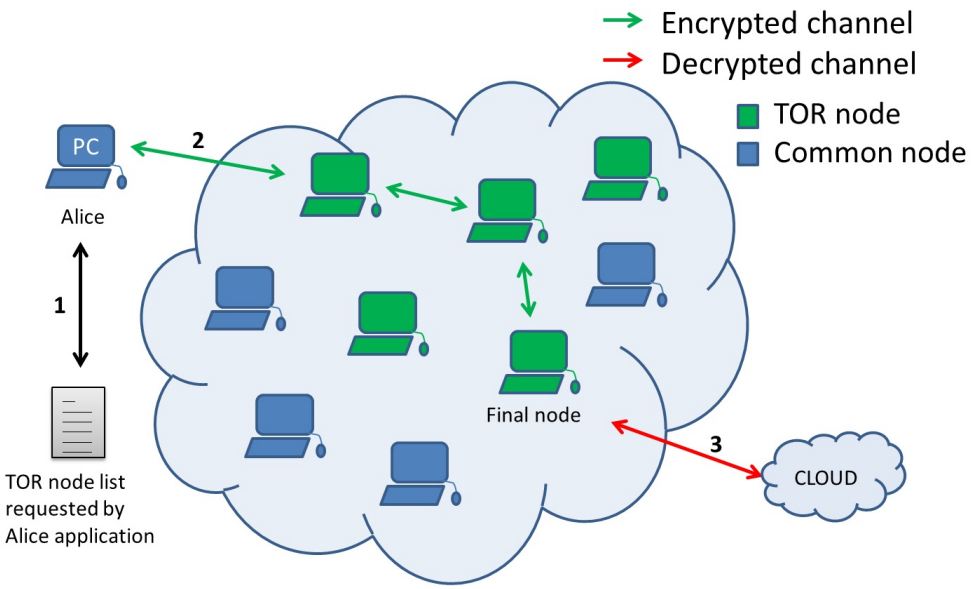


Figure 4 – Establishing a new TOR communication

users' device and the VPN-Proxy servers goes through a secure communication channel, those between VPN-Proxy and cloud are not necessarily encrypted. However, using this technology requires a trusted third party which will have access to all user information, and could be compromised. To protect users' privacy it is recommended that no information related to their accesses is stored in VPN-Proxy servers. The use of this technology is presented in Fig. 5.

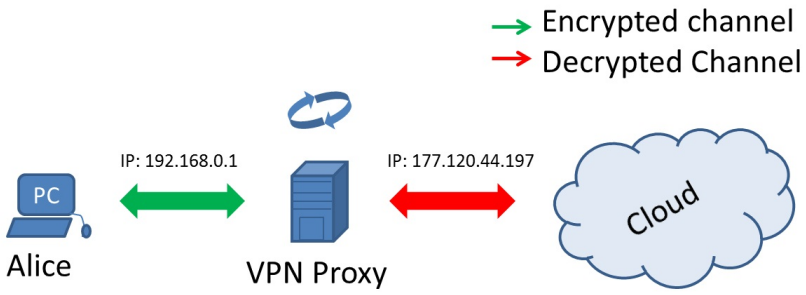


Figure 5 – VPN-Proxy communication

2.2.4 Data possession confidentiality

Data possession confidentiality is another point of concern. A CSP could easily link the data stored in their servers to its respective owner, which could bring some privacy problems to users. One way to solve this matter is through the anonymity, masking the possession of a certain file by an user. To this end, the CSPs could adopt the concept of federated identity (CHADWICK, 2009) in their solutions. It refers to the delegation of users' authentication process to a trusted third party, the Identity Provider (IdP). Besides the authentication, this third

party needs to keep user attributes and sometimes pass them to the Service Providers (SP), the CSPs. With the federated identity adoption, users only need to authenticate once to get access to several services in different places. When accessing cloud services, users authenticate with an IdP, which creates tickets (pseudonyms) for them. They may be therefore known to that CSP only by these pseudonyms and the access to their files in the cloud is done without the CSP linking the data stored to the owners' true names. In order to provide a higher privacy level, the IdP would not keep any information related to users' access to the CSP neither forward their real attributes, but only those related to the pseudonyms. Fig. 6 illustrates a basic scheme where users authenticate themselves in an IdP and receive a token containing the pseudonym, which is delivered to the CSP to allow them to have access to their data.

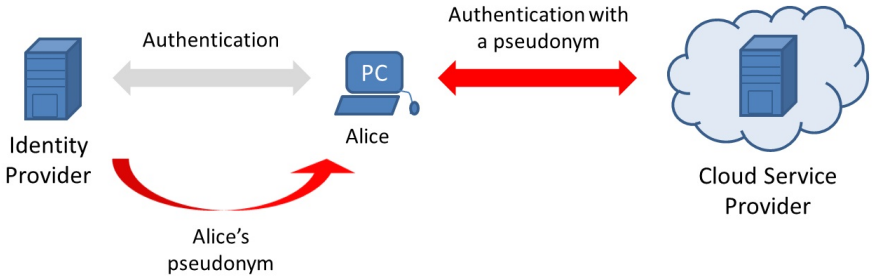


Figure 6 – Using a identity provider to access data in the cloud

2.3 Related work

In this section we describe the solutions and proposals found in the literature to mitigate some problems available in cloud data storage in order to improve users' privacy and the security of their data. Most of work focus on data confidentiality concerns, using techniques as cryptography at data at rest, data fragmentation or a combination of these two approaches. For each one, we present in the next subsections some related work and a brief discussion about the benefits and obstacles in their use. However, solutions employing the other techniques discussed previously for the specific purpose of providing privacy in the cloud storage were not found. Only a similar idea to the use Federated Identity technique to improve anonymity was found in Wang's work [Wang et al. 2009]. The basic principle of his proposal is to use an algorithm before data is sent to the cloud to make it anonymous, yet allowing it to be easily recovered later. Data is sent to the cloud with auxiliary information (external knowledge derived from other sources, as web, public domains etc.). In this way, it will not be possible to claim that certain data belongs or not to a certain user. This solution is client-side and does not need a trusted third party to keep the anonymity.

2.3.0.1 Cryptography

The most common way to achieve security and privacy on cloud data storage is through cryptography. Several proposals use this approach to avoid CSPs and intruders learning any sensitive information. This is one of the oldest techniques and, among its benefits, the greater is confidentiality. However, there are a few obstacles regarding the use of this technique in the cloud environment, which are the low performance and the difficulty of sharing, indexing and searching encrypted data. Another obstacle is the cryptographic key management which could be a great burden for system administrators or users, since the whole security rely on the protection of such keys.

The cryptography is adopted in several ways to improve users' privacy in the cloud. Some works prefer to stick with classical solutions using RSA, as those proposed by Kalpana (KALPANA; SINGARAJU, 2012) and Padmaja (PADMAJA; KODURU, 2013). In these proposals, all data is first encrypted and then sent to the cloud (client-side), but the authors do not describe how to manage users' keys and how to allow data sharing, since there is no protocol for these features described. A similar proposal is presented by Yin (YIN *et al.*, 2014) and Kumar (KUMAR *et al.*, 2012), where the authors use Elliptic Curve Cryptography (ECC) instead of the RSA for performance issues, arguing that it provides comparable security with small key-length, requiring less computation and being more efficient. In their proposal they allow users to share data through certificates, where users have to store the data intended to be shared in a public area. However, every user once authenticated in the system can access all data stored in the public area, since data is encrypted using user's private key. Data not intended to be shared is stored in private areas accessible only by their owners.

Kamara and Lauter (KAMARA; LAUTER, 2010) propose an architecture aimed to be open-source and allow secure data sharing through tokens created by data owners. In their model, also client-side, they use symmetric cryptography (AES) to encrypt users data and a searchable encryption scheme to create and encrypt indexes on data. Moreover, they propose an attribute-based encryption scheme to encrypt the symmetric keys based on appropriate policy, where only the ones who meet this policy can decrypt them. There is also a test to verify data integrity through a proof of storage technique. Access to shared data can not be revoked.

Another proposal using a client-side approach is presented by Xu et al (XU *et al.*, 2012b) with focus on data sharing, eliminating the key escrow problem and the need for certificates. The proposed scheme uses a re-encryption algorithm. Data is sent already encrypted to the cloud along with the DEK (Data Encryption Key) encrypted with the owner's public key and a list of the users that can have access to this data. Data is shared through a re-encryption key for each record, generated from data owner's private key and the recipient's public key. When an user requests this data, a proxy server in the cloud will use a re-encryption algorithm and the re-encryption key provided by the owner to transfer the encrypted DEK into a format that the recipient could decrypt using its private key. Neither the cloud nor the proxy (presented in the

cloud) have access to the DEK in unencrypted format.

A different approach is proposed by Gasti et al (GASTI *et al.*, 2010). It adopts client-side encryption and also allows users to share their data. A deniable cryptography scheme is used aiming to protect users' privacy even under coercion. With the use of this technique, users can decrypt their data and the resulting plaintext could be something completely different from the original data content if users use a second key. This is helpful in cases where they are coerced to decrypt sensitive information and did not want to reveal the original content. Using the right key they can get the original content. In this approach they use RSA (OAEP padding) for asymmetric algorithm and AES (CTR mode) for symmetric.

Phuong et al (PHUONG *et al.*, 2012) present a server-side model, where it requires a trusted third party responsible for the key management. In their model, data is encrypted by ElGamal asymmetric algorithm and could be shared with other users. Their focus is on retrieving encrypted data from the cloud and, as its name is also encrypted, they use a multi-user searchable encrypted data scheme, which allow a third party to look for encrypted data using encrypted keywords.

2.3.0.2 Data fragmentation

Another technique used to improve users' privacy in cloud data storage is through data fragmentation. Some benefits could be achieved by the use of this technique, such as the non-dependence on a single CSP (vendor lock-in), redundancy and greater data availability. Some authors say that splitting data into several small pieces could also achieve confidentiality, since the fragments would be small enough to reveal almost nothing about the original data. Also, to get the original data, attackers would need to recover all the fragments spread and rebuilt them in the correct way. There are several ways to split data, each one with its own benefits. However, this technique also presents some drawbacks, such as the ones related to confidentiality, since sensitive information could be obtained from the fragments, the complexity involved in using several clouds to store the data and also the sharing of data spread at multiple places.

Jaatun et al (JAATUN *et al.*, 2011a) present a solution using data fragmentation technique. In their proposal, there is a cloud service C&C (Control and Command) responsible for receiving users data and splitting them in several pieces to be stored in different CSPs. The authors highlighted that if the fragments were small enough and only its owner could recover all of them, data confidentiality could be achieved without cryptography. However, the fragments could still have some sensitive information and this is not handled by the authors. Also, data redundancy and data sharing are not a concern in their work, and are not dealt with.

Aiming to protect electronic medical records (EMR), Pao-Ching C.. et al (CHEN *et al.*, 2010) use data fragmentation in their solution, based on a model similar to RAID-3 technology (PATTERSON *et al.*, 1988). They use three different places to store the EMR: two CSPs and a local site. To ensure data confidentiality, they break each file at a byte level, where

a stream of data is stored byte by byte in a round-robin way in those places. However, they do not discuss the redundancy and data sharing in their solution.

RACS, another work using data fragmentation technique in the context of cloud data storage, is a system proposed by Abu-Libdeh et al (ABU-LIBDEH *et al.*, 2010). In their solution, a proxy interposed between users' application and a set of n CSPs, has to work as the RAID-5 technology, splitting users data into several fragments and storing them among the available providers, mostly to avoid vendor lock-in and reduce costs associated to cloud shifting. Erasure codes are used in their proposal to split data in an efficient way and also to allow redundancy in the system. However, data sharing it is not treated.

2.3.0.3 Cryptography and data fragmentation

The third way discussed in this work to protect users in cloud data storage environment is the combination of the two previous techniques: cryptography and data fragmentation. With the use of these two methods, users gain in terms of protection because of the advantages of the cryptography confidentiality and the redundancy and availability achieved by the fragmentation. However, as a drawback, some problems presented in these methods are added, as the key management in cryptography and data sharing in fragmentation.

The first discussed proposal using the two methods is due to Kumar M.S. and Kumar M. (KUMAR; KUMAR, 2013). They use a technique called Storage Efficient Secret Sharing (SESS) which uses the algorithm SSS (Shamir's Secret Sharing) to split data in several parts and requiring just some of them to recover the original data. The main purpose of this technique is to mitigate the computational complexity of SSS, reducing the high storage cost, which is a function of $S = n \cdot W$, where S is the total space required for all fragments, n the number of fragments resulting from the splitting process and W is the original data size. The SESS method reduces the storage overload to $S = W \cdot (1+n)/2$. Basically, this method splits data into two parts, where the first one is encrypted (RC4 algorithm - 160-bit key) using a key derived from the hash of the second one. The second part is the one which is applied to the SSS method to be split. All the parts are sent to different CSPs. There is no redundancy for the first part and data sharing is not allowed. Their approach use client-side encryption.

Ermakova and Fabian (ERMAKOVA; FABIAN, 2013) also describe an architecture using multiple clouds to store users data in order to improve the availability, confidentiality and integrity of medical records. The secret sharing is used for this end combined with an attribute-based encryption method (ABE), used to encrypt users data according to a certain policy. Data is also digitally signed. Firstly, it is encrypted and then broken in several pieces to be stored in different CSPs (client-side). To recover the original data, users only need part of the fragments. The method allows data sharing, but the details about the protocol are not provided.

Schnjakin et al (SCHNJAKIN *et al.*, 2013a) describe an architecture similar to RAID-5 technology. They split users data in several fragments and store them in different CSPs,

using Erasure Codes. They choose the providers which will store users data according to users' expectations, such as geographic location, quality of service, reputation and also budget preference. They also encrypt data before leaving users' computer (client-side), using symmetric encryption (AES algorithm).

2.3.0.4 Discussion

All proposals presented so far aims to mitigate some problems found in the cloud data storage. Table 2 presents the solutions along with the main techniques used by them to address each problem. Also, the proposals are classified as server-side or client-side. The later is the most privacy-friendly one and will be the only one considered as secure, as the operations and control of data are restricted only to users. The meaning of the letters is: *V* stands for problem solved; *X* indicates that the technique does not solve the particular problem and *P* shows that the technique provides a partial solution.

Table 2 – Comparison of literature solution in cloud data storage

	Threats									Features		
Papers	**Data leakage**	**Unknown data physical location**	**Lack of guarantee of total data destruction**	**Insecure App**	**Account/ service hijacking**	**Inability to access data**	**Data lock-in**	**Malicius insiders**	**Common shared infrastructure**	**Sha-ring data**	**Client-side encryp-tion**	**Algorithm**
(KALPANA; SINGARAJU, 2012)	V	V	V	X	X	X	X	V	V	X	V	RSA / Symmetric not mentioned
(PADMAJA; KODURU, 2013)	V	V	V	X	X	X	X	V	V	X	V	RSA / Symmetric not mentioned
(YIN *et al.*, 2014)	V	V	V	X	X	X	X	V	V	V	V	ECC
(KUMAR *et al.*, 2012)	V	V	V	X	X	X	X	V	V	V	V	ECC
(KAMARA; LAUTER, 2010)	V	V	V	P	X	X	X	V	V	V	V	AES / Attribute-based encryption / Searchable Encryption
(PHUONG *et al.*, 2012)	V	V	V	X	X	X	X	V	V	V	X	ElGamal
(XU *et al.*, 2012b)	V	V	V	X	X	X	X	V	V	V	V	Asymmetric / Symmetric / Re-encryption
(GASTI *et al.*, 2010)	V	V	V	X	X	X	X	V	V	V	V	RSA-OAEP / AES-CTR / Deniable encryption
(KUMAR; KUMAR, 2013)	V	V	V	X	X	V	V	V	V	X	V	RC4 / Secret Sharing
(ERMAKOVA; FABIAN, 2013)	V	V	V	X	X	V	V	V	V	V	V	Attribute-based encryption / Secret Sharing
(SCHNJAKIN *et al.*, 2013a)	V	V	V	X	X	V	V	V	V	X	V	AES / Erasure Codes
(JAATUN *et al.*, 2011b)	P	P	P	X	X	X	X	P	P	X	X	Not mentioned
(ABU-LIBDEH *et al.*, 2010)	P	P	P	P	X	V	V	P	P	X	X	Erasure Codes
(CHEN *et al.*, 2010)	V	V	V	X	X	X	X	V	V	X	X	Not mentioned

Some of these proposals are also not practical and still need some development until they can be used. According to the table, none of the solutions fulfil entirely all the threats discussed. Most of the work in this area is conceived aiming to address data leakage, malicious insiders, and others just with the adoption of cryptography. On the other hand, the insecure applications and account or service hijacking threats are not addressed by none of the proposals. The former it is addressed partially in only two cases. This indicates a necessity for further studies. The inability to access data and data lock-in are also addressed, but only by a few providers which use fragmentation techniques. These two threats and the data fragmentation technique will not be further studied in this work and will be left for future work.

2.4 Conclusions

In this chapter we presented and discussed some threats that users may face while using cloud data storage services. We also focused on users' privacy and detailed some concerns regarding it. Some techniques to mitigate these concerns were also presented and discussed. Finally, we presented solutions from the literature aimed to mitigate these threats, in order to improve the security and privacy in the cloud environment. These approaches were classified according to the technique used in their solutions. Cryptography, the first one discussed, improves users' confidentiality but brings problems with data sharing and with indexing and searching of encrypted files. Data fragmentation was the other technique presented to improve data availability, using methods that create data redundancy. The last technique discussed was a combination of the two previous ones to address most problems in data storage environments. However, in this combination the problems of both techniques are summed up, bringing new challenges.

Even though the academic solutions propose the most advanced solutions, they focus on specific problems and for this reason are insufficient for real world implementation. Moreover, some techniques used still require studies due to performance issues. Also, the proposals do not address some important threats to keep users protected in the cloud environment, requiring further studies.

3 Analysis of techniques to provide privacy in clouds

In this chapter we present a methodology to evaluate the techniques described in Chapter 2 to address users' privacy. In our analysis, we calculate the relative costs and benefits of each technique and their combinations, with the purpose of finding the best solutions to improve users' privacy.

3.1 Relative costs and benefits of the techniques

In this section, we will analyze the techniques introduced in chapter 2, evaluating their relative costs and benefits regarding to users' privacy. There are several ways to combine these techniques and it is important to highlight that these combinations will bring different costs and benefits to users. First, we will present a preliminary analysis of the techniques and discuss the initial results obtained. Then, we will present an improved analysis where we evaluate in detail each technique and some combinations. The main objective of both analysis is to find the combinations that produce the lowest cost/benefit ratios.

3.1.1 Preliminary analysis

For each technique or combination we establish a Relative Cost (*RC*) according to a subjective estimative of time and space costs in relation to the others. The necessary infrastructure for using each tool is considered to be already installed and ready for use, not incurring in extra costs. We also establish a subjective degree of Relative Privacy (*RP*) according to the estimated level obtained with each technique in relation to the others. We choose the simplest and less effective technique according to our point of view and expertise on the filed, and established a relative cost of one for it. For the other techniques, we assined weights to them according to how many times we consider they are more effective in protecting users' privacy. It is a completely subjective measure, but we consider it sufficient for our first comparative analysis, which is based not on the absolute values of each parameter, but on the relative values among them. The ratio *R* (cost/benefit ratio), is obtained from the division of *RC* by *RP*.

The techniques were named individually t_0, t_1, t_2, t_3, t_4 and t_5 as shown in Table 3, along with their *RC* and *RP* values. All of them can be combined except for TOR and VPN-Proxy since there is no reason for using them in the same solution because they have the same purpose (hide users' location) and this would only increase the costs with no further benefits for users. As a result, there are 48 different possible combinations, numbered from 1 to 59.

Table 3 – First analyses of the relative cost and privacy of the techniques used to provide privacy in the cloud

Symbol	Technique	RC	RP
t_5	Content encryption	4,0	3,5
t_4	Fragmentation with redundancy	10,0	2,5
t_3	Hide access via TOR	20,0	1,5
t_2	Hide access via VPN-Proxy	8,0	1,0
t_1	Hide data possession (Federated identity)	2,0	1,5
t_0	Metadata encryption	1,0	1,0

We considered files with the same size to estimate the costs, once some techniques (t_5, t_4, t_3, t_2) are directly affect by it. In order to identify all possible solutions, we proposed a code for each combination based on a binary weight given by each technique symbol. The 0 represents the absence of a given technique, while 1 indicates its use. Each one occupies a position in a 6 bits word, whose order is: $t_5, t_4, t_3, t_2, t_1, t_0$. Finally, this binary value is converted to a decimal one, which is used to represent the combination. An example of denomination for a solution that uses the Content Encryption (t_5), Fragmentation with redundancy (t_4) and TOR (t_3), would be $(111000)_2$ or $(56)_{10}$. It is considered that the RC_i and RP_i of a combination i are given by the sum of the *RCs* and *RPs* of component techniques.

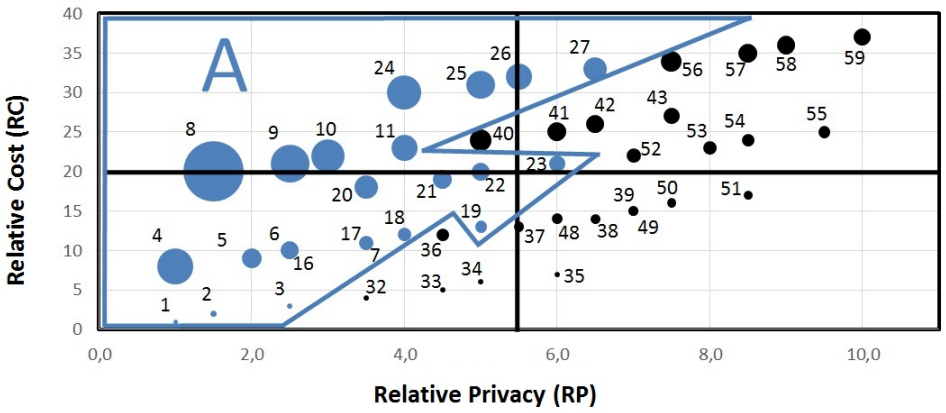


Figure 7 – Relative cost and privacy for different combinations

Fig. 7 illustrates all possible solutions with their respective relative costs and privacy ($RC_i x RP_i$) and also the value of $R_i = RC_i/RP_i$ of each combination i represented by the size of the circles. Notice that the most interesting combinations are the smallest ones located in the bottom (the numbers indicate the combinations).

Consider the combination number 59, having all techniques, except t_2. We can calculate R_{59} and then see how it changes when we remove each technique, one by one. This change gives us an idea about the weight of each in the formation of R_{59}. Then, removing t_0, t_1, t_3, t_4 and t_5 from combination 59 results in changes of $+8\%$, $+11\%$, -46%, -3% and $+37\%$,

respectively, in the value of R_{59}. We can see that techniques t_3 and t_4 are, individually, worsening R, as we get better values for it when they are removed individually. Although many other scenarios could be exercised here, the point is that the addition of another security technique may have an effect opposite to the desired one. Fig. 7 also indicates the impact of the adoption or not of t_5 (content encryption) in R. t_5 is inactive in A zone and active out of it (the bit related to t_5 is 0 on combinations 1 to 31). The inclusion of t_5 contributes significantly to the reduction of R. This kind of behaviour must be considered in the choice of this and other techniques. The preliminary analysis showed that the adoption of one or another security technique may have impacts in the overall cost-benefit ratio that are difficult to estimate. Although a precise value for the perceived privacy may be difficult to derive, we can do a more precise evaluation of the cost incurred by each technique.

3.1.2 Improved analysis of relative costs and benefits

In this subsection, we do an improved analysis of the techniques and their combinations to provide privacy in the clouds. The objectives are the same: finding the best solutions to improve users' privacy, and also point out the ones which are ineffective or even prohibited to use due to the elevated costs and low benefits. We will also analyse the *RC* and *RP* of each solution, but with a different approach. The *RC* is calculated based on the time needed to exchange messages between clients and CSPs (authentication phase), encryption/decryption times, data fragmentation time and so on. Then, it is divided by the shortest time (less costly technique) giving the *RC* value. As before, *RP* is estimated according to the level of privacy perceived from the use of each technique relative to the simplest one. The ratio R (cost/benefit ratio), is obtained in the same way as before, by the division of *RC* by *RP*.

In this improved analysis we also discuss the impact of file size in the results. In the next sections, we will show how *RC* and *RP* were calculated in detail, followed by a discussion about the new results found.

3.1.2.1 Calculating the Relative Cost

To calculate each technique cost, we considered a scenario where users want to download a file already stored in the cloud. Then, we analyzed all the messages exchanged between users' device and the CSPs using a given solution, and considered that the minimal number of messages are exchanged in order to complete the file transfer (i.e., there is no errors nor retransmissions). For all techniques and combinations we used the same context. The necessary infrastructure for using each tool is considered to be already installed and ready for use, not incurring in extra costs.

The first interaction between users and CSPs is related to the authentication process. Users send their credentials in the first message to the cloud in order to get access to their account and hence to their data. The CSP responds with an acceptance/refusal reply, which is

always considered to be acceptance in this calculation. The next step is a file request made by users, which is replied by the CSP with data, piece by piece until all the content is transferred (download phase).

Some techniques/solutions also need a secure communication channel, since private information may be sent on it. For this reason, extra messages are exchanged and after the process is finished, all the following messages are encrypted. In our analysis, we considered three types of standard messages: short,long and TOR messages. Fig. 8 illustrates the transmission of short messages, which can be sent through the network in plaintext or ciphertext. The same is valid for long messages, presented in Fig. 9. TOR messages are illustrated in Fig. 10. A short message (50 bytes) is characterized for being small and containing only the header and a little piece of information, used in the communication for exchanging parameters, credentials etc. The long messages (1500 bytes) contain the header and the largest possible chunk area, used mostly to send file pieces in the download phase. TOR uses special messages for communication within its network, with a fixed size of 512 bytes, containing specific fields (i.e. circuit identifiers, commands etc.).

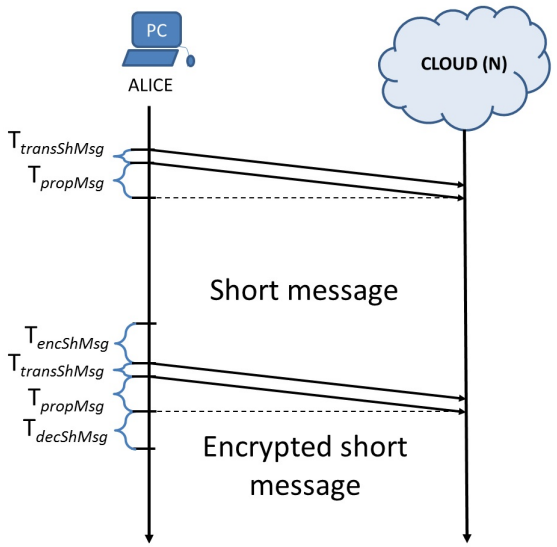


Figure 8 – Short messages sent during a communication process

As highlighted by Kurose J. F and Ross K. W. (KUROSE; ROSS, 2007), during the packets travel from one host to another there are some delays on the network along the path until they reach the destination. A packet passes through a series of routers during this journey, and the most important delays are the nodal processing, queuing, transmission, and propagation delays. Basically, they are defined as:

- Processing delay: Time to examine the packet's header. It may include factors as determining its next destination or error checking.

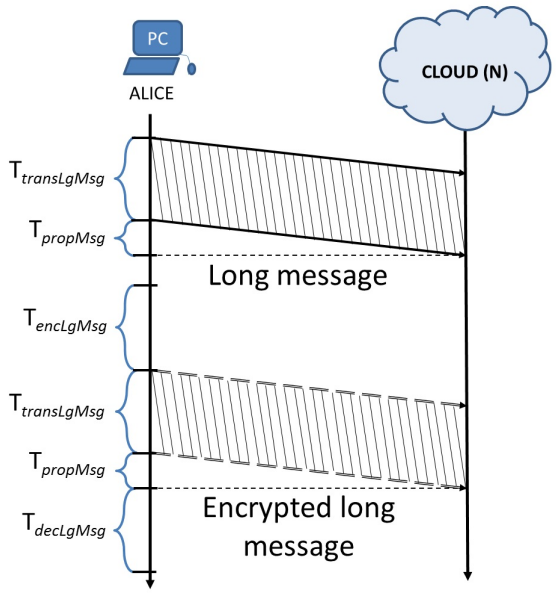


Figure 9 – Long messages used to transfer data between the peers

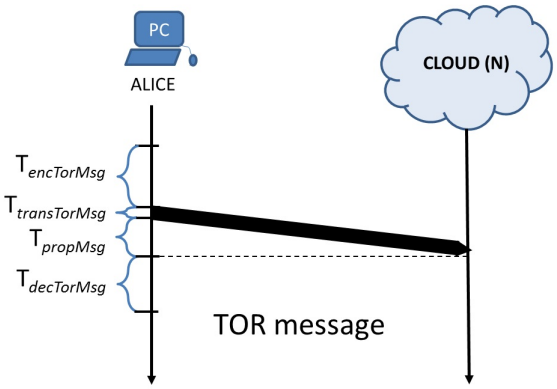


Figure 10 – Messages used in TOR between nodes

- Queueing delay: Time the packets have to wait to be transmitted. This time depends on the number of packets in the queue waiting to be transmitted, and it will be zero when there is no packets in the queue.
- Transmission delay: Time to put all the packet's bits into de link. This time is obtained from the division of the packet's length (l) by the transmission rate of the link ($trans_{rate}$).
- Propagation delay: Time a bit takes to propagates from the source to the destination. This time is obtained from the division of the distance between the nodes (d) by the propagation speed of the physical medium of the link ($prop_{speed}$).

In this work we will only consider the transmission and propagation delays in order to simplify the calculation. We also disregarded the presence of intermediary nodes along the network. In our approach, a packet leaves the source and goes directly to the destination. Also, we consider that there is no further packets to be processed in the queue and no error checking

or any other header processing. In some cases, it is necessary to consider the time to encrypt the packet as it leaves the source and the decryption time when it reaches the destination. For simplification purposes, the transmission and propagation times will be simplified in the next figures, when necessary, and will be refereed as T_{shMsg}, which is the sum of both. The physical media considered in this work is twisted-pair copper wire.

As each technique works in an unique way, they need to be analysed separately. Even when two or more techniques are combined in the same solution, their operation model may change and need to be constructed. For example, a solution using TOR and Federated Identity will not need to send authentication messages twice. In the approach used in this work, after the TOR circuit is created, users are authenticated with the federated identity technique but the necessary messages exchanged will not be short or long ones, but TOR messages. This special messages, explained below, will contain the data needed to authenticate users with their own IdPs. After the authentication phase, the file is downloaded using TOR infrastructure. As shown in this example, the cost of a combination will not be the sum of each technique isolated. The whole operation model needs to be constructed considering the scenario described. This was the method used in this work to calculate all combinations costs. The following items show the details of each technique that must be considered in possible combinations.

- Content encryption: The data encryption technique requires an additional step in its model. After data is downloaded to user's device, it needs to be decrypted. Fig 11 illustrates this process. The time for this operation is mostly influenced by the file size, but other factors as key size, algorithm and device's processor power will also affect it. When combining this technique with others, it is only necessary to add the decryption time at the end of the process.

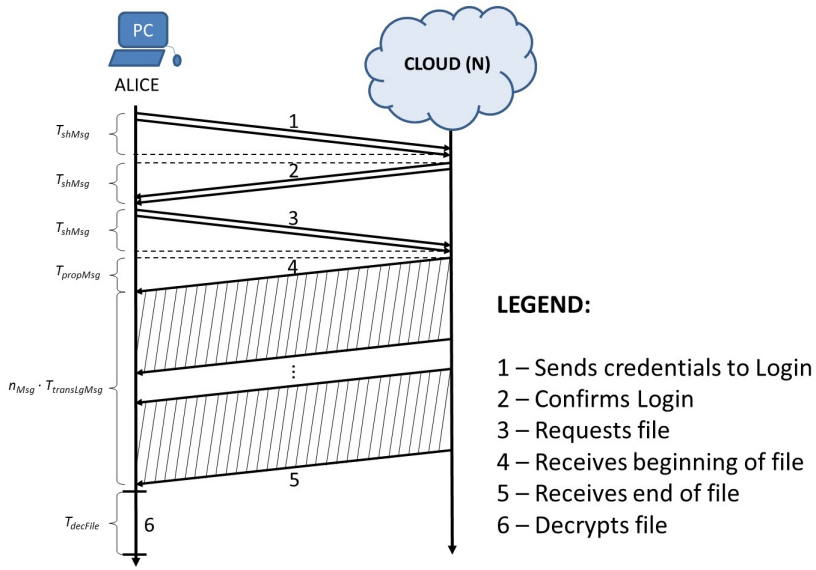


Figure 11 – Messages exchanged in content encryption technique

- Fragmentation with redundancy: Data fragmentation also requires an additional step when recovering files from the cloud, as shown in Fig. 12. Using this technique, data is broken into several fragments which are stored in different CSPs. In this analysis, we consider the use of parallelism in the recovery process, where several threads can be created to recover data pieces stored in different clouds simultaneously. This can decrease the necessary time for downloading the entire file, as the threads would download in parallel several file parts. However, this technique requires an additional step for rebuilding the original file. Also, the file size and the parameters chosen in the algorithm (K and M, as see in chapter 2) can increase/decrease this time. Combining this technique requires considering this extra time for reconstructing data. The download time will be that of a fragment instead of the entire data.

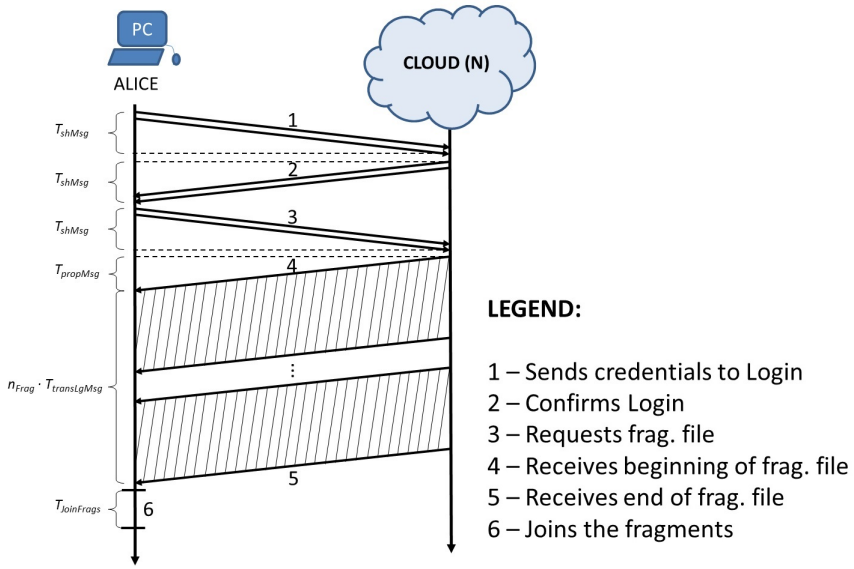


Figure 12 – Messages exchanged in the fragmentation technique

- Hide access via TOR: TOR is the most complex technique presented. In TOR, users first need to create a circuit, where random TOR nodes will be chosen to block others from knowing users' location. A secure communication channel must be established with each node in order to exchange messages. After the circuit creation, all communication with the cloud will be through it. These intermediary nodes and all encryption necessary in the communication increase the costs of using TOR, as showed in Figs. 13 to 17. This is due to the high number of messages exchanged in the entire process. The encryption/decryption times related to the secure communication channel are hidden in the figures, but the messages under this channel are the ones represented by the largest black arrows and are considered in the formulas. On the other hand, the times related to the encryption/decryption performed the the nodes are separately. In this figure, we represented the add/removal of an encryption layer by braces and powered by the number of layers in that message. When this technique is combined with others, usually TOR times overcome

other ones, which increases the costs since the messages will travel in an encrypted form and will go through extra nodes.

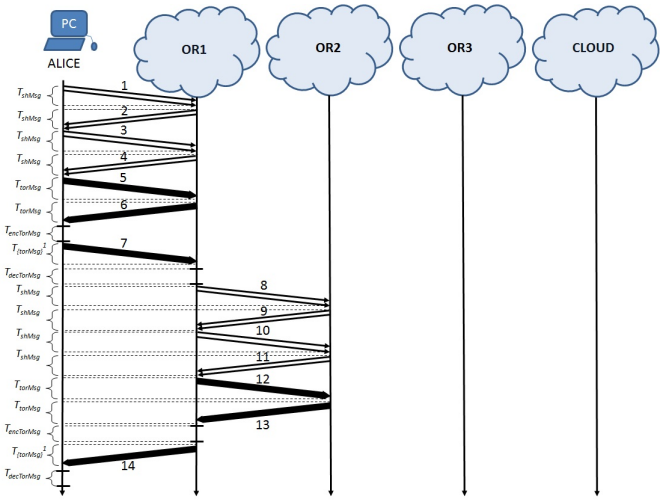


Figure 13 – Messages exchanged in TOR - Circuit creation

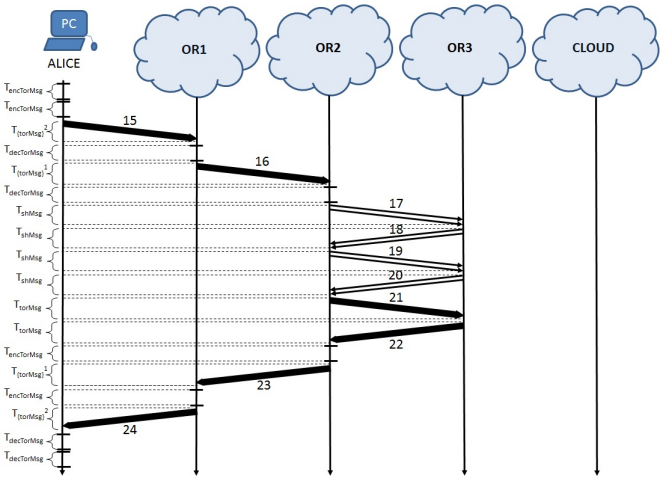


Figure 14 – Messages exchanged in TOR - Circuit creation (cont.)

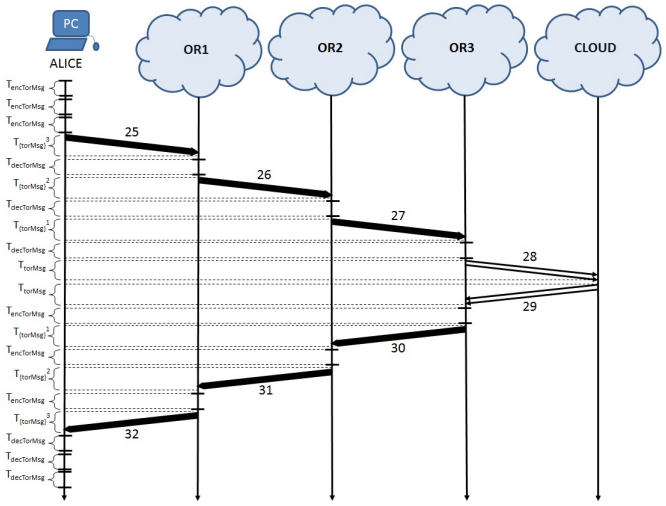


Figure 15 – Messages exchanged in TOR - Accessing time

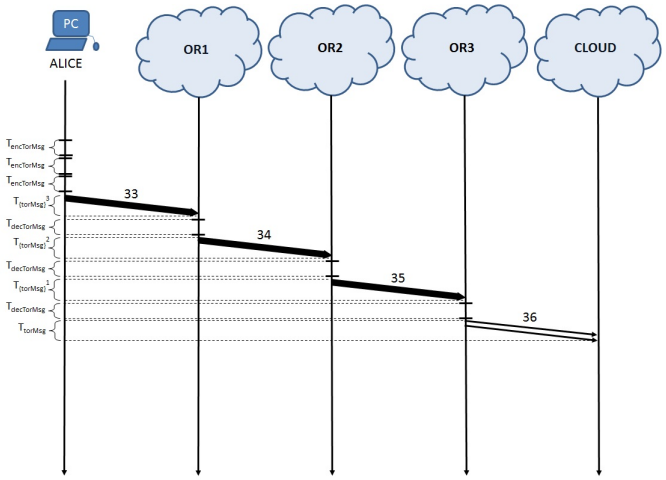


Figure 16 – Messages exchanged in TOR - Accessing time (cont.)

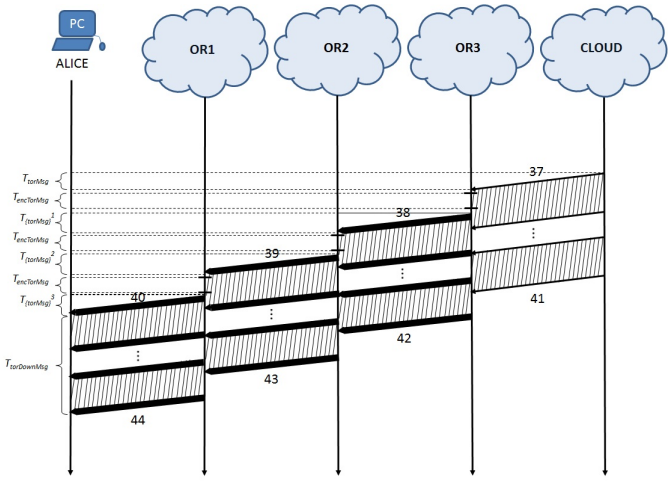


Figure 17 – Messages exchanged in TOR - Downloading

- Hide access via VPN-Proxy: Instead of a direct communication between users and cloud, there is a VPN-Proxy server in the middle which receives users' messages and requests, delivers them to the cloud and vice-versa. Besides the extra messages in the communication, there is also an extra cost related to the communication between users and the VPN-Proxy server, which occurs in an encrypted way. The whole process is showed in Fig. 18. The time related to the encryption/decryption is not included in the figures, but it is considered in the formulas. Combining VPN-Proxy server with other techniques requires considering the extra times in each encrypted message which is first sent to the proxy server, decrypted and then delivered to the cloud with a different IP. In the way back, similar steps will be necessary.

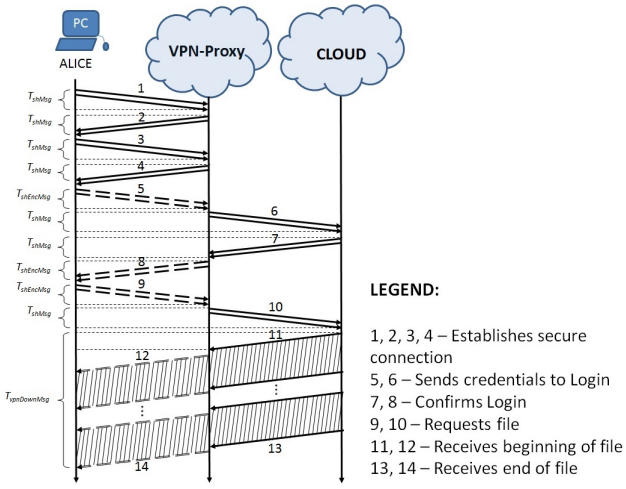


Figure 18 – Messages exchanged in access the cloud through VPN-Proxy technique

- Hide data possession (Federated identity): Using Federated identity scheme for authentication also incurs in additional costs, with an extra phase in the authentication process, as illustrated in Fig. 19. In order to use this technique, users first access their CSPs and choose to authenticate with a third party, the IdP (Identity Provider). They are redirected and start the authentication process with their own providers. Then, they are brought back to their CSPs already authenticated and able to access their data. The messages to request and download data are the same as in the normal operation process. For this reason, when combining this technique with others, only the extra messages related to the authentication process should be taken into consideration.

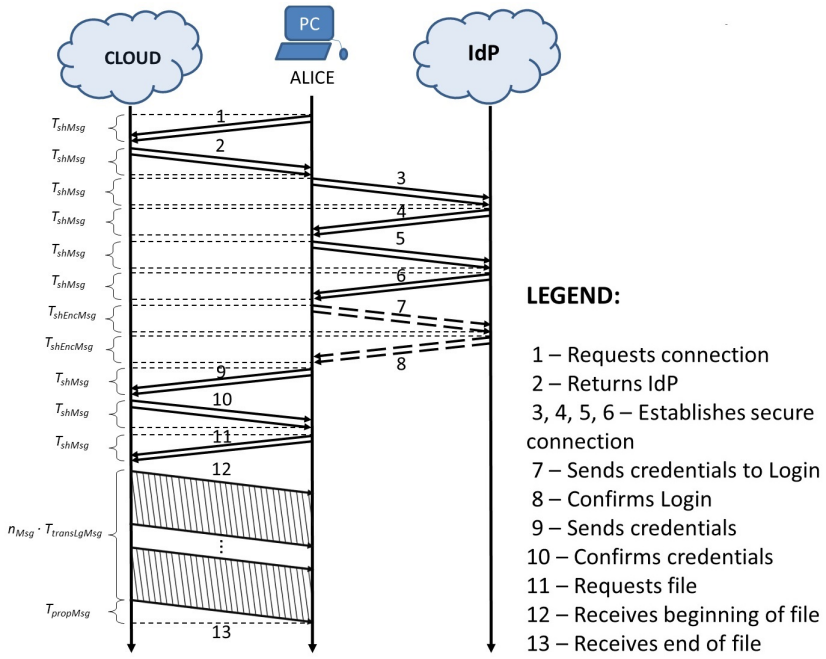


Figure 19 – Messages exchanged in the federated identity scheme

- Metadata encryption: The metadata encryption technique works similarly as the data encryption when considering the cost calculation process, but instead of having the content encrypted, its metadata, as name, size, ownership etc. that will be. This process must occur first, since the CSP will need the ciphertext filename in order to retrieve the right file. This process is shown in Fig. 20. Combining this technique with others only requires adding in the process the extra time for metadata encryption.

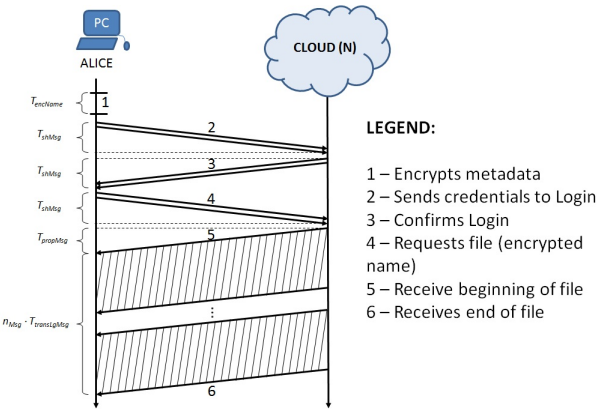


Figure 20 – Messages exchanged in Metadata encryption technique

Formulas to calculate the relative costs

For calculating the costs associated with each solution, we built expressions for each one based on the operation process, considering all details explained above. Then, we calculated the costs and normalized them dividing by the shortest time (less costly technique) to obtain the

RC value. Table 4 presents all 48 possible combinations and their formulas. In this table, besides the formula of each solution, we also show the binary and decimal codes used to represent each solution. The parameters used in the formulas are defined in table 5. Next, we present some details about the values used in the formulas to derive the costs and some considerations during the calculus.

Table 4 – Time costs for all combinations of techniques

Techniques	**Binary**	**Decimal**	**Formula**
t_0	000001	1	$T = T_{encName} + (3 * T_{shMsg}) + T_{propMsg} + (n_{msg} * T_{transLgMsg})$
t_1	000010	2	$T = (9 * T_{shMsg}) + (2 * T_{shEncMsg}) + T_{propMsg} + (n_{msg} * T_{transLgMsg})$
$t_1 - t_0$	000011	3	$T = T_{t_1} + T_{encName}$
t_2	000100	4	$T = (7 * T_{shMsg}) + (3 * T_{shEncMsg}) + T_{vpnDownMsg}$
$t_2 - t_0$	000101	5	$T = T_{t_2} + T_{encName}$
$t_2 - t_1$	000110	6	$T = (13 * T_{shMsg}) + (13 * T_{shEncMsg}) + (2 * T_{propMsg}) + T_{transLgMsg} + T_{encLgMsg} + (n_{msg} * T_{transLgMsg}) + T_{decLgMsg}$
$t_2 - t_1 - t_0$	000111	7	$T = T_{encName} + T_{t_2 - t_1}$
t_3	001000	8	$T = T_{circuit} + T_{authMsgTor} + T_{downFileTor}$
$t_3 - t_0$	001001	9	$T = T_{t_3} + T_{encName}$
$t_3 - t_1$	001010	10	$T = T_{t_3} + (3 * (T_{sendTorMsg} + T_{recTorMsg})) + (2 * (4 * (T_{propMsg} + T_{transTorMsg})) + (7 * (T_{encTorMsg} + T_{decTorMsg})))$
$t_3 - t_1 - t_0$	001011	11	$T = T_{encName} + T_{t_3 - t_1}$
t_4	010000	16	$T = (3 * T_{shMsg}) + T_{propMsg} + (n_{frag} * T_{transLgMsg}) + T_{joinFrags}$
$t_4 - t_0$	010001	17	$T = T_{t_4} + T_{encName}$
$t_4 - t_1$	010010	18	$T = (9 * T_{shMsg}) + (2 * T_{shEncMsg}) + (T_{propMsg}) + (n_{frag} * T_{transLgMsg}) + T_{joinFrags}$
$t_4 - t_1 - t_0$	010011	19	$T = T_{encName} + T_{t_4 - t_1}$
$t_4 - t_2$	010100	20	$T = (7 * T_{shMsg}) + (3 * T_{shEncMsg}) + (2 * T_{propMsg}) + T_{transLgMsg} + T_{encLgMsg} + (n_{frag} * T_{transLgMsg}) + T_{decLgMsg} + T_{joinFrags}$
$t_4 - t_2 - t_0$	010101	21	$T = T_{encName} + T_{t_4 - t_2}$
$t_4 - t_2 - t_1$	010110	22	$T = (13 * T_{shMsg}) + (13 * T_{shEncMsg}) + (2 * T_{propMsg}) + T_{transLgMsg} + T_{encLgMsg} + (n_{frag} * T_{transLgMsg}) + T_{decLgMsg} + T_{joinFrags}$
$t_4 - t_2 - t_1 - t_0$	010111	23	$T = T_{encName} + T_{t_4 - t_2 - t_1}$
$t_4 - t_3$	011000	24	$T = T_{circuit} + T_{authMsgTor} + (4 * (T_{propMsg} + T_{transTorMsg})) + (6 * T_{encTorMsg}) + (2 * T_{decTorMsg}) + (4 * n_{frag} * T_{decTorMsg}) + T_{joinFrags}$
$t_4 - t_3 - t_0$	011001	25	$T = T_{encName} + T_{t_4 - t_3}$
$t_4 - t_3 - t_1$	011010	26	$T = T_{circuit} + (3 * (T_{sendTorMsg} + T_{recTorMsg})) + (2 * (4 * (T_{propMsg} + T_{transTorMsg}) + (7 * (T_{encTorMsg} + T_{decTorMsg}))) + T_{authMsgTor} + (4 * (T_{propMsg} + T_{transTorMsg})) + (6 * T_{encTorMsg}) + (2 * T_{decTorMsg}) + (4 * n_{frag} * T_{decTorMsg})) + T_{joinFrags}$
$t_4 - t_3 - t_1 - t_0$	011011	27	$T = T_{encName} + T_{t_4 - t_3 - t_1}$
t_5	100000	32	$T = (3 * T_{shMsg}) + T_{propMsg} + (n * T_{transLgMsg}) + T_{decFile}$

Continued on next page

Table 4 – Time costs for all combinations of techniques (continued)

Techniques	**Binary**	**Decimal**	**Formula**
$t_5 - t_0$	100001	33	$T = T_{t_5} + T_{encName}$
$t_5 - t_1$	100010	34	$T = T_{t_1} + T_{decFile}$
$t_5 - t_1 - t_0$	100011	35	$T = T_{encName} + T_{t_1} + T_{decFile}$
$t_5 - t_2$	100100	36	$T = T_{t_2} + T_{decFile}$
$t_5 - t_2 - t_0$	100101	37	$T = T_{encName} + T_{t_2} + T_{decFile}$
$t_5 - t_2 - t_1$	100110	38	$T = T_{t_1 - t_2} + T_{decFile}$
$t_5 - t_2 - t_1 - t_0$	100111	39	$T = T_{encName} + T_{t_2 - t_1} + T_{decFile}$
$t_5 - t_3$	101000	40	$T = T_{t_3} + T_{decFile}$
$t_5 - t_3 - t_0$	101001	41	$T = T_{encName} + T_{t_3} + T_{decFile}$
$t_5 - t_3 - t_1$	101010	42	$T = T_{t_3 - t_1} + T_{decFile}$
$t_5 - t_3 - t_1 - t_0$	101011	43	$T = T_{encName} + T_{t_3 - t_1} + T_{decFile}$
$t_5 - t_4$	110000	48	$T = T_{t_4} + T_{decFile}$
$t_5 - t_4 - t_0$	110001	49	$T = T_{encName} + T_{t_4} + T_{decFile}$
$t_5 - t_4 - t_1$	110010	50	$T = T_{t_4 - t_1} + T_{decFile}$
$t_5 - t_4 - t_1 - t_0$	110011	51	$T = T_{encName} + T_{t_4 - t_1} + T_{decFile}$
$t_5 - t_4 - t_2$	110100	52	$T = T_{t_4 - t_2} + T_{decFile}$
$t_5 - t_4 - t_2 - t_0$	110101	53	$T = T_{encName} + T_{t_4 - t_2} + T_{decFile}$
$t_5 - t_4 - t_2 - t_1$	110110	54	$T = T_{t_4 - t_2 - t_1} + T_{decFile}$
$t_5 - t_4 - t_2 - t_1 - t_0$	110111	55	$T = T_{encName} + T_{t_4 - t_2 - t_1} + T_{decFile}$
$t_5 - t_4 - t_3$	111000	56	$T = T_{t_4 - t_3} + T_{decFile}$
$t_5 - t_4 - t_3 - t_0$	111001	57	$T = T_{encName} + T_{t_4 - t_3} + T_{decFile}$
$t_5 - t_4 - t_3 - t_1$	111010	58	$T = T_{t_4 - t_3 - t_1} + T_{decFile}$
$t_5 - t_4 - t_3 - t_1 - t_0$	111011	59	$T = T_{encName} + T_{t_4 - t_3 - t_1} + T_{decFile}$

Table 5 – Parameters to calculate the time costs

Parameters	**Description**
n_{msg}	Number of messages necessary to transmit the entire file
n_{frag}	Number of messages necessary to transmit a file fragment
n_{tor}	Number of messages necessary to transmit a file using TOR
$T_{encName}$	Time to encrypt the filename
$T_{decFile}$	Time to decrypt the file content

Continued on next page

Table 5 – Parameters to calculate the time costs (continued)

Parameters	**Description**
$T_{propMsg}$	Time to propagate a message
$T_{transShMsg}$	Time to transmit a short message
T_{shMsg}	Time to propagate and transmit a short message ($T_{propMsg} + T_{transShMsg}$)
$T_{shEncMsg}$	Time to encrypt, propagate, transmit and decrypt a short message
$T_{transLgMsg}$	Time to transmit a long message
$T_{encLgMsg}$	Time to encrypt a long message
$T_{decLgMsg}$	Time to decrypt a long message
T_{lgMsg}	Time to propagate and transmit a long message ($T_{propMsg} + T_{transLgMsg}$)
$T_{joinFrags}$	Time to reconstruct the original file based on the fragments
T_{t_N}	Time of the technique/solution N
$T_{transTorMsg}$	Time to transmit a TOR message
$T_{encTorMsg}$	Time to encrypt a TOR message
$T_{decTorMsg}$	Time to decrypt a TOR message
T_{torMsg}	Time to propagate and transmit a tor message ($T_{propMsg} + T_{transTorMsg}$)
T_{torMsg^n}	Time to propagate and transmit a tor message with n encryption layers ($T_{propMsg} + T_{transTorMsg}$)
$T_{sendTorMsg}$	Time to send a message using a TOR connection: $T = (4 * (T_{propMsg} + T_{transTorMsg})) + (6 * (T_{encTorMsg} + T_{decTorMsg}))$
$T_{recTorMsg}$	Time to receive a message using a TOR connection: $T = (4 * (T_{propMsg} + T_{transTorMsg})) + (6 * (T_{encTorMsg} + T_{decTorMsg}))$
$T_{circuit}$	Time to create a TOR circuit, necessary to initiate any communication: $T = (12 * (T_{shMsg} + T_{propMsg} + T_{transTorMsg})) + (18 * (T_{encTorMsg} + T_{decTorMsg}))$
$T_{authMsgTor}$	Time to authenticate users and request files using TOR: $T = (2 * T_{sendTorMsg}) + T_{recTorMsg}$
$T_{torDownMsg}$	Time to transmit the file from the first node (OR1) to the source (Alice): $T = (2 * T_{decTorMsg}) + (4 * n_{tor} * T_{decTorMsg})$
$T_{downFileTor}$	Time to download the file required using a TOR connection: $T = (4 * (T_{propMsg} + T_{transTorMsg})) + (6 * T_{encTorMsg}) + T_{torDownMsg}$

Continued on next page

Table 5 – Parameters to calculate the time costs (continued)

Parameters	**Description**
$T_{vpnDownMsg}$	Time to transmit the file from the cloud to the source (Alice): $T = (2 * T_{propMsg}) + T_{transLgMsg} + T_{encLgMsg} + (n_{msg} * T_{transLgMsg}) + T_{decLgMsg}$

Table 6 presents the values of the parameters used in the formulas to calculate the relative costs of the solutions. In this table, we present the parameters, a brief description of them and the values adopted.

Table 6 – Relative costs: parameters and values

Parameter	**Description**	**Value**
s	File size	1 KiB /10 KiB /100 KiB /1 MiB /10 MiB /100 MiB /1 GiB /10 GiB /100 GiB
$l_{msgHead}$	Message header length	40 bytes
$l_{shMsgPl}$	Short message payload length	10 bytes
l_{shMsg}	Short message length	$l_{shMsgPl} + l_{msgHead}$
$l_{lgMsgPl}$	Long message payload length	1460 bytes
l_{lgMsg}	Long message length	$l_{lgMsgPl} + l_{msgHead}$
$l_{torMsgHead}$	TOR message header length	14 bytes
$l_{torMsgPl}$	TOR message payload length	498 bytes
l_{torMsg}	TOR message length	$l_{torMsgPl} + l_{torMsgHead}$
k	Necessary parts to recover the original file	3 units (defined)
m	Redundant parts	1 unit (defined)
n	Total number of data fragments	$k + m$
s_{frag}	File fragmented size	$(s \times (1 + (m/k)))/n$
n_{msg}	Number of messages necessary to transmit the entire file	$s/l_{lgMsgPl}$
n_{frag}	Number of messages necessary to transmit a file fragment	$s_{frag}/l_{lgMsgPl}$

Continued on next page

Table 6 – Relative costs: parameters and values

Parameter	**Description**	**Value**
n_{tor}	Number of messages necessary to transmit a file using TOR	$s/l_{torMsgPl}$
enc_{rate}	Encryption rate	23923444,98 Bytes/s (measured)
$trans_{rate}$	Transmission rate	1×10^8 bits/s (defined)
d	Distance between peers	2000 meters (defined)
$prop_{speed}$	Propagation speed (Twisted-pair copper wire)	2×10^8 meters/sec (from ref. (KUROSE; ROSS, 2007)
$T_{propMsg}$	Time to propagate a message (Propagation delay)	$d/prop_{speed}$
$T_{transShMsg}$	Time to transmit a short message	$(l_{shMsg} \times 8)/trans_{rate}$
$T_{encShMsg}$	Time to encrypt a short message	l_{shMsg}/enc_{rate}
$T_{decShMsg}$	Time to decrypt a short message	$T_{encShMsg}$
T_{shMsg}	Time to propagate and transmit a short message	$T_{propMsg} + T_{transShMsg}$
$T_{shEncMsg}$	Time to encrypt, propagate, transmit and decrypt a short message	$T_{encShMsg} + T_{propMsg} + T_{transShMsg} + T_{decShMsg}$
$T_{transLgMsg}$	Time to transmit a long message	$(l_{lgMsg} \times 8)/trans_{rate}$
$T_{encLgMsg}$	Time to encrypt a long message	l_{lgMsg}/enc_{rate}
$T_{decLgMsg}$	Time to decrypt a long message	$T_{encLgMsg}$
T_{lgMsg}	Time to propagate and transmit a long message	$T_{propMsg} + T_{transLgMsg}$
$T_{lgMsgEnc}$	Time to encrypt, transmit and decrypt a long message	$T_{encLgMsg} + T_{propMsg} + T_{transLgMsg} + T_{decLgMsg}$
$T_{transTorMsg}$	Time to transmit a TOR message	$(l_{torMsg} \times 8)/trans_{rate}$
$T_{encTorMsg}$	Time to encrypt a TOR message	l_{torMsg}/enc_{rate}

Continued on next page

Table 6 – Relative costs: parameters and values

Parameter	**Description**	**Value**
$T_{decTorMsg}$	Time to decrypt a TOR message	$T_{encTorMsg}$
T_{torMsg}	Time to propagate and transmit a TOR message	$T_{propMsg} + T_{transTorMsg}$
T_{torMsg^x}	Time to propagate and transmit a TOR message with x encryption layers	$T_{propMsg} + T_{transTorMsg} + (x \times (T_{encTorMsg} + T_{decTorMsg}))$
$T_{encName}$	Time to encrypt the filename	$T_{encShMsg}$
$T_{decFile}$	Time to decrypt the file content	s/enc_{rate}
$T_{joinFrags}$	Time to reconstruct the original file based on the fragments	$T_{decFile}/12$

Observations:

- The times to encrypt and decrypt are considered equal for every kind of message.
- $T_{joinFrags}$: This time was defined according to some tests performed using a *Reed-Solomon* code (from Erasure Codes family) with the parameters defined in Table 6 and comparing it to the time needed for decrypting data in AES algorithm. In the tests performed, the time to reconstruct the fragments was generally twelve times shorter than the one to decrypt data.
- $T_{encName}$: We consider the time for encrypting filenames the same as for short messages, since most names considering extensions do not have more than 50 bytes.
- 1 GiB = 1024 MiB = 1024 $\times$ 1024 KiB and 1 KiB = 1024 Bytes.
- To determine the encryption rate (enc_{rate}) and the time to reconstruct the original file based on its fragments ($T_{joinFrags}$), we conducted experiments in a machine with the following configuration: Windows 7 Professional (32 bits) system with an Intel Core(TM) i3-330M with two cores of @2,13GHz, 3,00 GB DDR3 RAM and a 5400 rpm hard drive.

3.1.2.2 Calculating the Relative Privacy

The relative privacy was estimated with a different approach. First, we established a subjective degree of Relative Privacy (*RP*) based on a survey conducted with average users

where they had to estimate the level of privacy obtained for each technique, in a scale of 0 (no privacy at all) to 10 (highest level of privacy). This was an on-line survey where we expected that the participants expressed their perception of privacy using the techniques, considering their preferences, priorities and experiences. We provided a brief description for each technique with its characteristics, as well as how it can be used to preserve users' privacy. They should have considered the entire process of storing and recovering files in public clouds and all possible issues they could see related to the exposure of sensitive information, especially to the cloud administrators. The people invited to take part in the survey was expected to have some expertise in computer, in any of the related fields. Among the participants, we had professors, master, doctor, undergraduate students and some professionals of the field. The way the survey was conducted and the results found are presented and discussed in appendix A.

However, the results were not satisfactory according to our perception. Analyzing the values established by the participants, we noticed that according to a general point of view, the techniques give similar benefits for users. We believe that this is not the case. Some methods, as the content encryption, can be more efficient in protecting users' privacy than others, as the metadata encryption for example. We believe that new methods to evaluate the *RP* are necessary. The way we constructed the survey could have influenced the results, since we provide short explanations in order to not annoying the participants. On the other hand, if a more detailed description was made, a lot of them could have ignored the survey or choose random values just to finish it as soon as possible. Also, we believe that a detailed threat modelling description could help users understand the issues and give more precise and well informed opinions.

Thus, we decided to stick with the preliminary analysis method adopted. We choose the less effective technique according to our point of view and expertise on the field, and established a relative cost of one for it. The other techniques were compared to this one and to the others, and assigned with weights according to how many times they are more effective in offering privacy to users. We took into consideration the context being used, the cloud data storage, and the benefits related only to the privacy they can offer to users. For example, due to the benefits achieved by the content encryption, as the information secrecy, we believe this technique is 10 times more effective than metadata encryption in a privacy point of view. TOR is believed to be five times, and so on. For each technique, we briefly explain the main reasons for the adopted values, followed below.

- Metadata encryption: This technique received the smallest grade and it is the reference for normalizing the others in the relative privacy estimation. We believe that using only this technique for privacy purposes does not give large advantages to users with respect to their privacy, since anyone is still able to read the data content and get information from it.

- Federated Identity: This technique received a grade that is twice the value of metadata

encryption since it gives some extra protection to users' privacy when used alone.

- Hide access via TOR: TOR is used to hide users' location and for most users this is a useful privacy tool. Considering the context of cloud data storage, we understand that it offers five times more privacy than metadata encryption.

- Hide access via VPN-Proxy: VPN-Proxy technique has the same goal as TOR does. However, this is a less effective way to hide users' location, and for this reason the grade granted is lower than TOR.

- Content encryption: The content encryption technique receives the highest grade. This is due to the secrecy achieved when data is encrypted. Since the context of the present study is about cloud data storage, we believe that users will prefer this technique to the others for protecting their privacy.

- Fragmentation with redundancy: This technique is another form to protect users' privacy, but less effective than cryptography since some methods used to break data in several fragments do not offer properly secrecy properties. Data chunks could revel some useful information, which is not desirable from a privacy point of view. However, this technique introduces some extra benefits to users, as redundancy and performance, which are not counted as they are not related to privacy. In this approach, we do not evaluate the consequences of geopolitical factors in the CSPs choice and let this issue for future work.

3.1.2.3 Discussion

After the calculation performed using the formulas and methods described above, we construct some tables and graphics to show the results. The first analysis were made for 1 MiB files, showed in table 7, where all techniques are presented along with their new *RC* and *RP* values. Note that the scale of *RP* was normalized in the range from one to ten.

Table 7 – Relative cost and privacy of techniques used to provide privacy in the cloud with 1 MiB files

Symbol	**Technique**	**RC**	**RP**
t_5	Content encryption	3,97	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	5,32	5,00
t_2	Hide access via VPN-Proxy	2,68	3,00
t_1	Hide data possession (Federated identity)	2,68	2,00
t_0	Metadata encryption	2,67	1,00

Fig 21 illustrates all the possible solutions and their respective relative costs and privacy ($RC_i \times RP_i$), and also the cost/benefit ratio $R_i = RC_i/RP_i$ of each combination i represented by the size of the circles. In this figure, we considered the transfer of 1 MiB files.

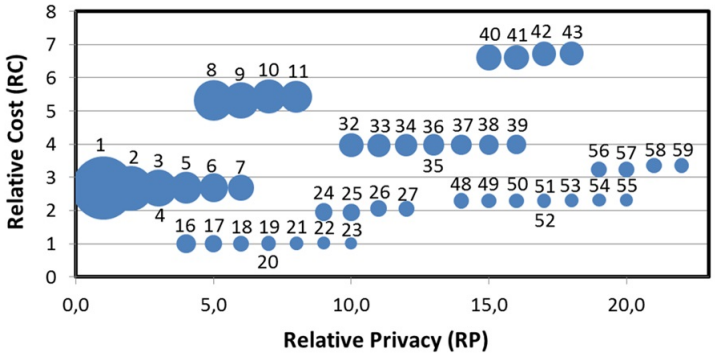


Figure 21 – Relative cost and privacy for each possible combination (1 MiB file storage). The circle area is proportional to the cost/benefit ratio.

As an example of how we calculated the ratio R, we demonstrate the calculation for the combination $(110001)_2$ or $(49)_{10}$. This solution uses the content encryption, fragmentation with redundancy and metadata encryption techniques. As t_5 and t_0 only add extra times related to the encryption of the content and metadata, respectively, we will show how to calculate the relative cost of t_4 and then add this two extra times to the result. In table 4 we can get the formulas to calculate the costs. The general formula for this combination is $T_{49} = T_{encName} + T_{t_4} + T_{decFile}$, where $T_{encName}$ represents the necessary time to encrypt the filename and $T_{decFile}$ the time to decrypt the file content. T_{t_4} is the time for calculating the whole process using only the fragmentation technique, and is obtained by the formula: $t_4 = (3 * T_{shMsg}) + T_{propMsg} + (n_{frag} * T_{transLgMsg}) + T_{joinFrags}$. After calculating these formulas using approximated values obtained in practical tests, we get the final value of the cost for this solution. Then, we divide it by the lowest costly technique, which is t_4, getting the relative cost RC. Now, we calculate the relative privacy of this solution. This is done by getting the relative privacy established for each technique and adding them all. The cost/benefit ratio R is got by dividing RC by RP.

In this new analysis, the individual removal of techniques t_0, t_1, t_3, t_4 and t_5 from combination 59 caused a variation in R_i $(i = 59)$ of $+4,8\%$, $+6,3\%$, $-11,3\%$, $+145,1\%$ and $+12,5\%$, respectively for 1 MiB files. Based on this results, we concluded that the adoption of t_3 technique must be done carefully, since its costs may not compensate the benefits. On the other hand, t_4 is highly recommended as its absence increased R significantly. This way, this technique can be used to balance some solutions. Using t_2 instead t_3 had a different impact, causing a variation in R_i $(i = 55)$ of $+5,3\%$, $+10,7\%$, $+16,9\%$, $+115,3\%$ and $-11,9\%$, respective. In this mode, the absence of t_5 provided the most positive impact, while the non use of t_4 the most negative impact.

In this analysis, the introduction of content cryptography do not contribute to the reduction of R as in the first one. Another difference is related to the variations seen in R_{59} in both analysis, which presented different results. However, both analysis pointed out to t_3 (TOR) as a bad choice since it worsens R. t_0, t_1 and t_5 are similar in both results, while t_4 presented an

opposite result from the first analysis. Indeed, the latter technique had the best positive effect when added individually to this combination.

When using t_2 (VPN-Proxy) instead of t_3 (TOR), the combination R_{55} will have a variation of $+31,7\%$ in relation of R_{59}. If we compare all the combinations using t_2 with the corresponding ones using t_3, we will have a variation of $68,4\%(R_{27}/R_{23})$, $49,9\%(R_{43}/R_{39})$, $26,5\%(R_{57}/R_{53})$ and $31,1\%(R_{58}/R_{54})$. Based on these results, we can conclude that using t_2 is more effective that t_3 in all cases. Moreover, it is worth using this technique than let it out of a combination, as showed in the analysis where adding it improves the ratio R in $16,9\%$.

In tables 8 and 9 we considered 10 MiB and 100 MiB files respectively. The values found did not change significantly with the file size variation, as observed in the graphics 22 and 23 related to the tables. The reason is that neither *RC* nor *RP* changed. The *RP* values were based on our perception of privacy and so they should remain the same despite of changes in file sizes. We also did the calculations for other file sizes. For larger files, as 1 GiB, 10 GiB and 100 GiB, the results were practically the same. However, for files lower than 1 MiB, as 1 Kib (table 10 and Fig. 24), 10 kiB (table 11 and Fig. 25) and 100 kiB (table 12 and Fig. 26), the changes were significant. In these cases, the download time becomes small compared to the other times involved, as those for authentication, secure channel establishment and so on, altering the relative costs and the graphics. The first conclusion we can draw from these results is that the use of privacy preserving techniques with files smaller than 1 MiB may have a cost/benefit ratio too high to be practical, depending on the combination of techniques adopted. Nonetheless, users normally store in clouds files that are steadily increasing in size (high resolution pictures, videos, documents etc) and small files are mostly used by the operating system (configuration files etc). Therefore, for storing user data in clouds securely, several techniques and their combinations can be used in a cost effective way.

Table 8 – Relative cost and privacy of the techniques used to provide privacy in the cloud with 10 MiB files

Symbol	**Technique**	**RC**	**RP**
t_5	Content encryption	3,97	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	5,24	5,00
t_2	Hide access via VPN-Proxy	2,68	3,00
t_1	Hide data possession (Federated identity)	2,68	2,00
t_0	Metadata encryption	2,68	1,00

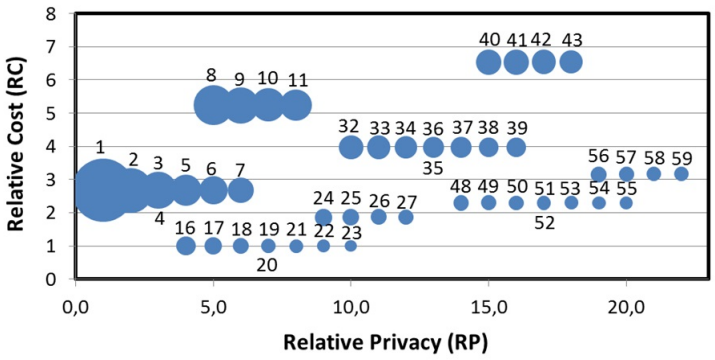


Figure 22 – Relative cost and privacy for each possible combination (10 MiB file storage). The circle area is proportional to the cost/benefit ratio.

Table 9 – Relative cost and privacy of the techniques used to provide privacy in the cloud with 100 MiB files

Symbol	**Technique**	**RC**	**RP**
t_5	Content encryption	3,97	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	5,23	5,00
t_2	Hide access via VPN-Proxy	2,68	3,00
t_1	Hide data possession (Federated identity)	2,68	2,00
t_0	Metadata encryption	2,68	1,00

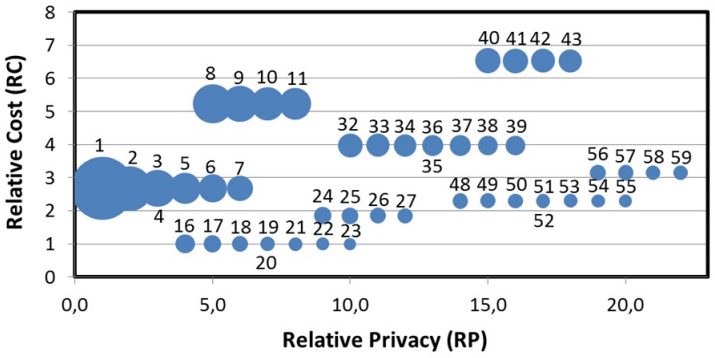


Figure 23 – Relative cost and privacy for each possible combination (100 MiB file storage). The circle area is proportional to the cost/benefit ratio.

Table 10 – Relative cost and privacy of the techniques used to provide privacy in the cloud with 1 kiB files

Symbol	Technique	RC	RP
t_5	Content encryption	2,12	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	40,25	5,00
t_2	Hide access via VPN-Proxy	5,95	3,00
t_1	Hide data possession (Federated identity)	3,07	2,00
t_0	Metadata encryption	1,66	1,00

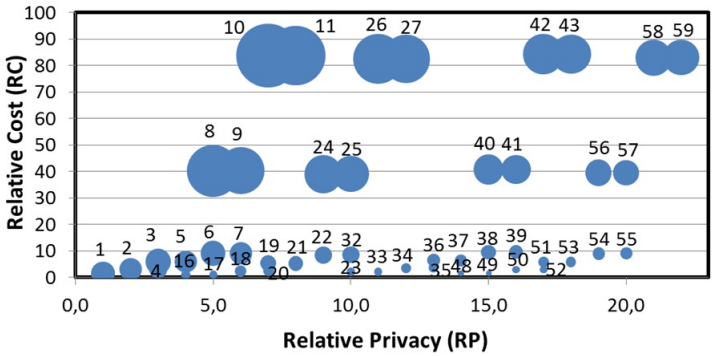


Figure 24 – Relative cost and privacy for each possible combination (1 kiB file storage).

Table 11 – Relative cost and privacy of the techniques used to provide privacy in the cloud with 10 kiB files

Symbol	Technique	RC	RP
t_5	Content encryption	3,55	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	13,20	5,00
t_2	Hide access via VPN-Proxy	3,42	3,00
t_1	Hide data possession (Federated identity)	2,77	2,00
t_0	Metadata encryption	2,44	1,00

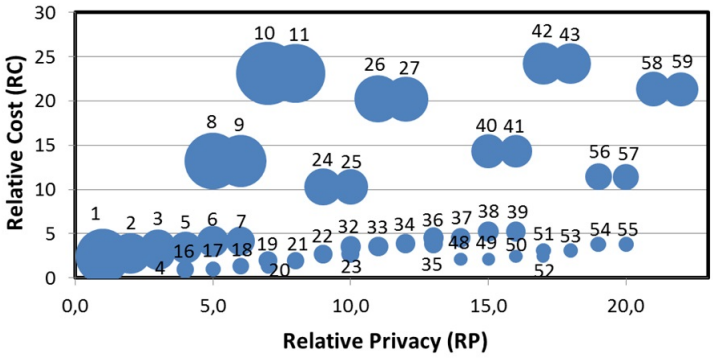


Figure 25 – Relative cost and privacy for each possible combination (10 kiB file storage).

Table 12 – Relative cost and privacy of the techniques used to provide privacy in the cloud with 100 kiB files

Symbol	Technique	RC	RP
t_5	Content encryption	3,93	10,00
t_4	Fragmentation with redundancy	1,00	4,00
t_3	Hide access via TOR	6,14	5,00
t_2	Hide access via VPN-Proxy	2,76	3,00
t_1	Hide data possession (Federated identity)	2,69	2,00
t_0	Metadata encryption	2,65	1,00

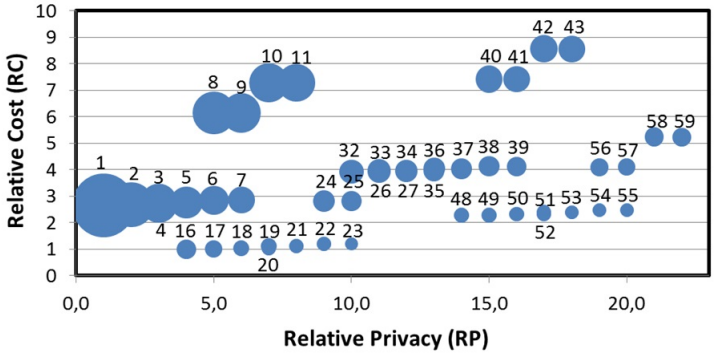


Figure 26 – Relative cost and privacy for each possible combination (100 kiB file storage).

We also analyzed the top ten solutions considering files of 1 MiB, which are, respectively: 23, 22, 55, 54, 21, 53, 51, 52, 19 and 50, in increasing order or ration R. All of them use t_4 (data fragmentation), since it helps in the performance of the combination, decreasing file download time and, consequently, the relative cost. Besides, it has a good relative privacy. The combination that use t_5 (content encryption) appear from the third position on the list, even having t_5 the highest RP value among all the techniques individually. This is due to the high costs associated to it. However, for those more concerned about privacy and willing to adopt a

combination that uses at least t_5, solutions as 55, 54, 53 and 51 are good ones and offer a high privacy with reasonable cost/benefit ratios.

This top ten list remains similar for file sizes from 100 kiB to 100 GiB. However, for smaller files it changes. For 10 kiB files, the ten best solutions are: 49, 51, 48, 50, 53, 52, 19, 55, 54 and 17. For 1 kiB they are: 49, 48, 51, 50, 33, 17, 32, 16, 35 and 34. In these new ranks, the best combinations mostly use t_5 (content encryption) with t_4 (fragmentation with redundancy), t_2 (VPN-Proxy), t_1 (federated identity) and t_0 (metadata encryption) showing that t_3 (TOR) is expensive and prohibitive when the files are small. We also noticed that the best solutions use at least three or more techniques in a combination, showing us that these techniques are complementary and can give good results when working together.

On the other hand, the bottom ten solutions considered cost ineffective or even prohibitive due to their elevated ratio R, are: 1, 2, 8, 4, 3, 9, 10, 11, 5 and 6, for file sizes varying from 1 MiB to 100 GiB. In these combinations, the most predominant techniques are t_0 (Metadata encryption) and t_1 (Federated identity), which are known for the few benefits provided. t_3 (TOR) also appears in the list as the third most predominant one, since it presents high costs. A similar rank is observed for 100 kiB files: 1, 2, 8, 10, 9, 4, 11, 3, 5 and 6. For 10 kiB files, the rank is: 10, 11, 8, 1, 9, 26, 27, 42, 2 and 43. Considering 1 kiB files, we have: 10, 11, 8, 26, 27, 9, 42, 43, 24 and 58. In all worst ranks, it is clear the high occurrence of t_3 and its negative impact on ratio R due to the elevated costs associated to the extra procedures, as the circuit creation, for example. Therefore, we can see that t_3 should be used only in very specific scenarios where the need to hide users' location justifies its high cost, especially when small files are considered. We also noticed that the worst combinations have less than two techniques in a combination (except in 11, using t_3, t_1 and t_0).

From our analysis, we concluded that t_3 (TOR) is a bad choice since it worsens R while t_4 gives an opposite result: it has the best positive effect when added individually to a combination. Also, it is clear that using combinations is more effective than using techniques alone, considering their cost/benefit relation.

3.2 Conclusions

In this chapter we presented an analysis of the relative costs and benefits of some techniques and their combinations used to improve users' privacy on cloud data storage, with the purpose of finding the solutions with the best cost/benefit ratios. We presented two analysis; a first one based on a estimative about the costs and benefits provided by the techniques and a second and improved version of the first one, where a more detailed evaluation was performed. We also showed how we calculated the relative costs, explaining the details taken into consideration. The relative privacy was also discussed and we showed how the values were defined. Based on the analysis, users can choose the solutions that best fit their needs. Moreover, they can

see which combinations are worth and which ones are ineffective due to their high costs and lower benefits. We also discussed the impact caused by each technique individually in some combinations and showed how file sizes may impact the cost/benefit ratio (i.e. cost/privacy ratio) for each possible combination of privacy preserving techniques. Besides, we found that it is more effective to use combinations than techniques isolated to preserve users' privacy.

4 Requirements for secure cloud data storage

This chapter aims to present some requirements for a secure, reliable and user-friendly application for data storage services on public clouds. These requirements will be separated into two groups, the first one related to security and the other related to usability. From now on, we will only focus on concerns and threats related to data content and attributes confidentiality, while others related to the data access and possession confidentiality will be covered in future works. We will present and discuss some systems aimed to improve the security and privacy in cloud data storage, with the purpose of understanding the subject's state of art and finding opportunities for further studies.

4.1 Requirements

The requirements are separated into two groups: security and usability. In the first group, there is a set of requirements which aims to protect users from several attacks and threats in cloud data storage environment. The latter group deals with requirements that are essential in making the user experience as simple and productive as possible, avoiding the complexity normally present in security applications.

4.1.1 Security requirements

In this subsection we propose a small set of security requirements for data storage services on public clouds. For each requirement we will present its characteristics and discuss how to make a good use of it from a security point of view. These requirements were chosen because they are the most important ones as they have direct impact on security. They can mitigate common attacks available nowadays, like those related to unauthorized disclosure of users data, insecure applications, insider attacks, account hijacking, among others.

4.1.1.1 Cryptographic keys security

NIST (National Institute of Standards and Technology) provides recommendations about cryptographic keys that should be adopted by all systems related to security that use cryptography. These recommendations are important for system security in a long term period, and are related to the management of cryptographic keys, cryptoperiods and the states that a key undergoes during its lifetime. The right management of keys is very important, since the whole security of a cryptographic system stands on them.

Cryptoperiod is defined by NIST (BARKER W. BARKER, 2006) as the time span during which the use of a key is allowed, and it is based on factors like the estimated effective lifetime of the key algorithm, type and purpose of a key. The states of cryptographic keys are formed by six possible states that a key undergoes during its life. These states are: pre-activation, active, deactivated, destroyed, compromised and destroyed compromised. All the states and their possible transitions are described in Fig. 27.

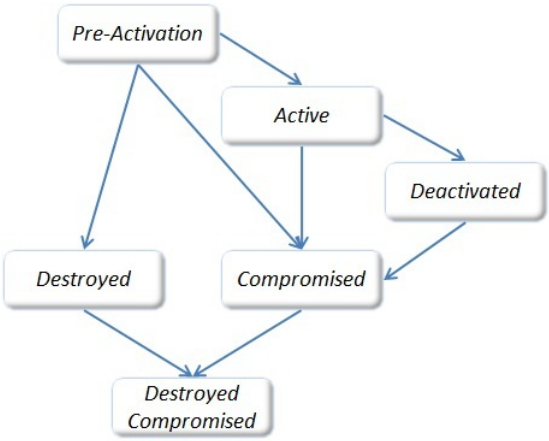


Figure 27 – Key states and transitions (Adaptation from (BARKER W. BARKER, 2006)

These states change according to specific events, such as the expiration of a cryptoperiod or the detection of a compromised key. More details about cryptoperiods, the states and all the possible transitions on a key in its life cycle can be seen in Ref. (BARKER W. BARKER, 2006).

These recommendations may not seem so important in the beginning, but some time later a system that did not followed them could be compromised and have users data leaked.

4.1.1.2 Secure deduplication

Deduplication is a technique used to eliminate redundant data in a storage system, in order to save disk space. Instead of storing multiple copies of a file, it creates links of such file and stores only a single copy (HARNIK; SHULMAN-PELEG, 2010). The deduplication process could be done considering the whole file or just pieces (or fragments) of it, which is more effective since parts of the file are more likely to have similar ones than the whole file. When an entire file is considered, even small changes will make the process consider it as different one. However, if this file is split in pieces, it is more likely to found equal pieces and deduplicate them, as some fragments could have the same content. The efficiency of the split process depends of the technique being used, and it is a difficult task to perform it in order to be as efficient as possible (ESHGHI; TANG, 2005).

Some CSPs adopt this technology in their servers; however there are some concerns in the way they implement it, especially because it could generate vulnerabilities and not only

economic advantages. There are two ways to perform the deduplication process: target-based or source-based. The target-based approach is characterized by the unawareness of the client about the deduplication process, since it occurs only on the CSP (server side). The main objective of this approach is to save disk space, and the process only starts after users data are stored in the cloud.

In the source-based approach, the deduplication process occurs on the client side, before data is sent to the cloud. The client application is responsible for verifying the existence of files in cloud servers before sending them. They usually send just a small piece of information related to that file (a file's hash, for example) and compares it to what is stored. The advantage of this approach is that, besides space, it also saves bandwidth, as data is never sent to the cloud if a copy of it already exists there. In that case, the CSP just creates a link of that data in the user's area. Data is only sent if there is no copy of it already stored.

Another characteristic of deduplication process is related to the search space used to look for equal data. There are two ways to make this search: single-user and cross-user. The single-user searches only in the space belonging to a particular user, while the cross-user searches in the entire storage infrastructure, considering all users' accounts. When an user makes a request to the CSP for storing some data on their servers and the CSP searches only in this user's account for redundant data, it is a single-user deduplication. However, if the CSP looks for redundant data in all users' accounts, it is a cross-user deduplication. The latter method is more efficient considering the amount of disk space saved, since a group of users are more likely to have the same file. As a drawback, the cross-user mode is more vulnerable to a security attack, as pointed out by Harnik et al. (HARNIK; SHULMAN-PELEG, 2010).

This attack is known as the identifying files attack. Providers that implement the source-based approach combined with cross-user deduplication are the ones vulnerable to it. To understand the attack, consider an attacker who wants to know if some user in a chosen cloud has a given file. First, he tries to send the intended file to the cloud and keeps monitoring the network to check if the file will be uploaded to the cloud. If so, it means that no one has this file stored in the cloud. Otherwise, the file will not be sent and the attacker can conclude that there is an equal file already stored. If the attacker is a government agency, it may force the CSP to release the file owner's name.

In order to avoid attacks as the one described and to obtain some benefits from deduplication, the cross-user approach should be combined with target-based approach only. This would save disk space and protect users' privacy.

Besides that, there is one possible scenario where the deduplication process becomes less useful. This happens when cryptography is used to protect users data. Two equal files encrypted with different keys will be completely different due to the cryptography algorithm properties (STALLINGS, 2008). As each user will choose its own cryptographic key to encrypt his data, the results from the encryption function will be totally different from each

other, even when two users have the same file, making the deduplication process useless.

One possible solution to this problem is to encrypt the data using its own hash as the key (WANG *et al.*, 2010). This way, two equal files will be encrypted with the same key and will produce the same encrypted file, enabling the deduplication process. However, it is important to notice that the attack described above will still be possible if this solution is adopted.

4.1.1.3 High level of data secrecy

The levels of secrecy are related to the way cryptography is used to protect users data. Here we propose a way to classify such levels according to the amount of privacy they provide.

- **Level 0 - No encryption**: This level represents the use of no encryption at all. Users send data in plaintext to the CSP which stores it in plaintext as well. Data is vulnerable during transmission to attacks like tampering and sniffing, and also while stored to threats like those described previously. Nowadays, it is unlikely to find CSPs using this level of secrecy, since the HTTPS protocol (IETF, 2000a) is being broadly adopted, offering a secure communication channel, which elevates the security to Level 1, as described next.

- **Level 1 - Communication channel encryption(CCE):** This level is characterized by the use of cryptography only in the communication channel between the user and the CSP. Technologies like TLS (Transport Layer Security) (IETF, 2008) are usually used to provide security. It creates a tunnel between user's application and the CSP, and all data transmitted between them go through this tunnel encrypted. Attacks like tampering and sniffing are no longer possible. However, this level only provides secrecy in the communication channel. Data will be encrypted as it enters the tunnel to be sent to the CSP and will be decrypted as it leaves the tunnel in the CSP, being stored in plaintext file. Fig. 28 and 29 illustrate the process of storing and recovering data from the cloud, respectively. Data stored in plaintext file gives the CSP access to users data to improve search mechanisms or for advertising purposes. Besides, the deduplication process could be done easily. In this level, the process of sending and receiving data requires four cryptographic operations (client encryption, server decryption, server encryption and client decryption, respectively). However, there is no overhead for users related to key management and the entire process is transparent to the user.

- **Level 2 - CCE and server-side encryption:** This level corresponds to the use of cryptographic protocols by the CSPs to protect users data while in transit and at rest. CSPs are responsible for performing all cryptographic operations on users data and also for taking care of the key management. The protection of the communication channel is required,

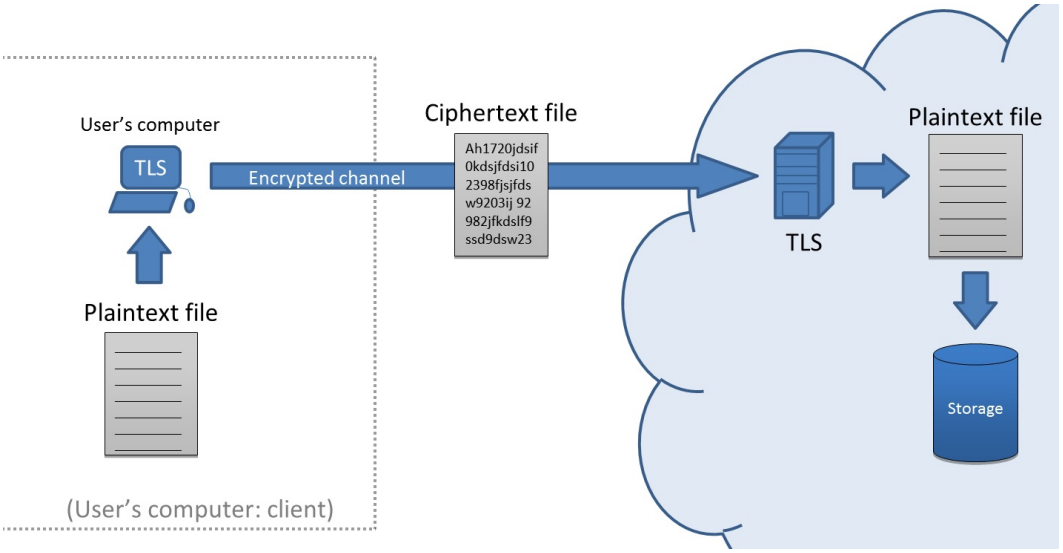


Figure 28 – Data is sent to the cloud through an encrypted channel

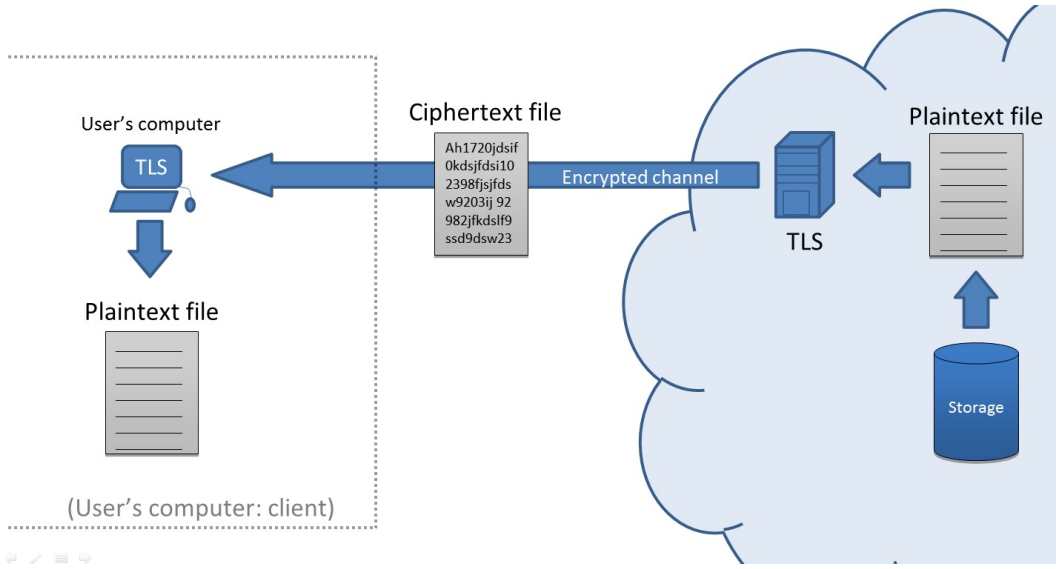


Figure 29 – Data is recovered from the cloud through an encrypted channel

since there is no meaning in sending data from user's device to the CSP in plaintext file to encrypt it later.

In a basic scenario (Fig. 30) data is sent in a secure way (using protocols like TLS) to the cloud, which is responsible for generating a secret key to encrypt it, before storing it on the servers. This secret key is wrapped by a public key and stored along with the data it relates to. In the recovery process (Fig. 31), the CSP uses the corresponding private key to decrypt the secret key and then opens the data. After the decryption process is finished, data is sent back to the user through a secure channel. As in Level 1, the CSPs also have access to users data, since they manage all cryptographic keys. They are still able to improve search mechanisms or do advertising, and also to use deduplication techniques to save disk space. Sending and receiving data require six cryptographic op-

erations: Four, as in Level 1, plus two additional ones required by the encryption and decryption of data when it enters and leaves the CSP. In this level there is also some cryptographic operations related to the encryption/decryption of the symmetric keys, which will not be counted since they have a much smaller costs than the ones done in data content. As a way to keep all keys safe, some CSPs encrypt the secret key (or private key) using a KEK (Key Encryption Key) derived from user's password using algorithms like PBKDF2 (Password Based Key Derivation Function) (IETF, 2000b), which makes a key recovery process easier for users as they only need to remember a secret and also harder for attackers than usual in brute force attacks. Level 2 also takes all the necessary overhead related to key management, eliminating this burden from users.

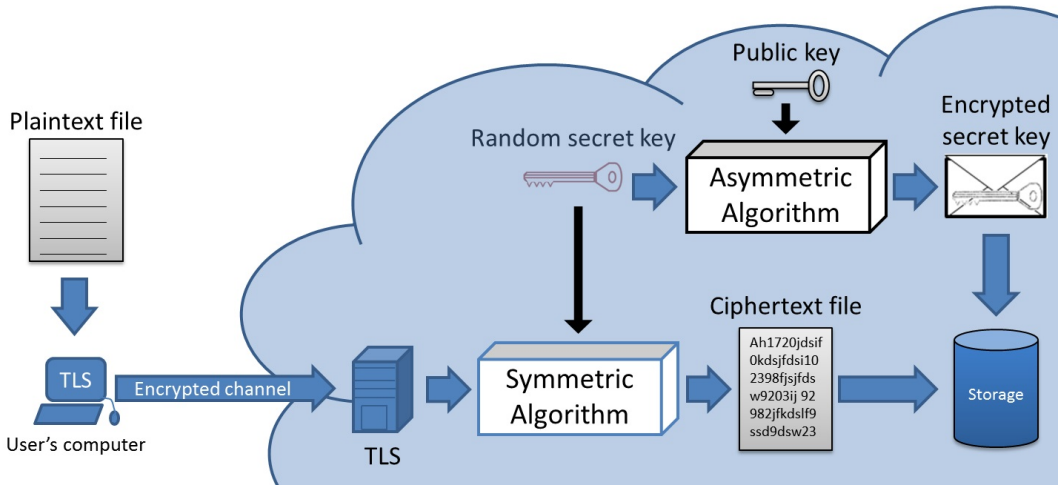


Figure 30 – Storing data in the cloud

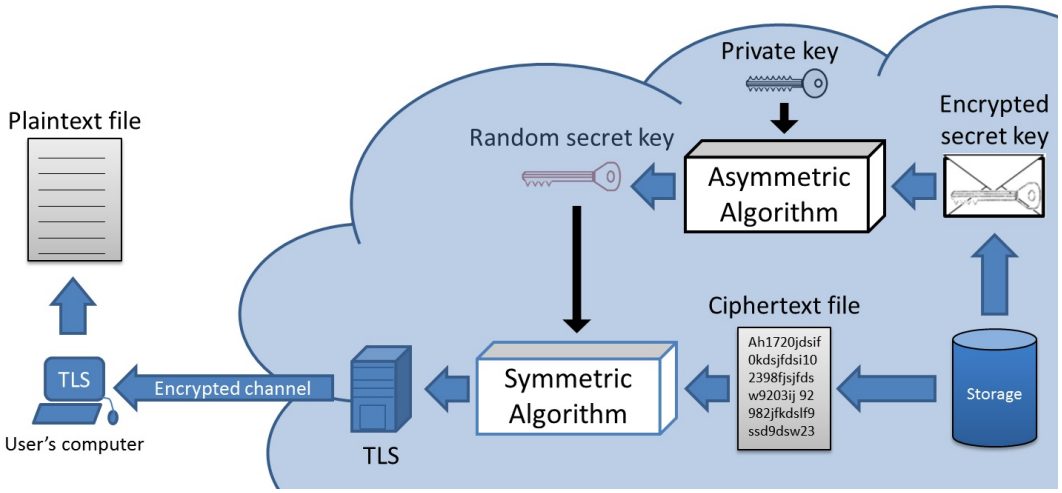


Figure 31 – Recovering data from the cloud

- **Level 3 - Client-side encryption:** This level corresponds to the cryptographic operations being performed on client-side. Before sending data to the cloud, users encrypt and send

it to be stored in the cloud. There are applications provided by some CSPs to this end, but some users may prefer to use outsourced and more trusted applications, like PGP (Pretty Good Privacy), GPG (Gnu Privacy Guard) or even the CPG (Cloud Privacy Guard) proposed in this work. In this level, users no longer need a secure communication channel to transmit their data to the cloud, as data leave users' devices encrypted. Also, the CSPs are not capable to do any processing in the data and deduplication can no longer be done, unless some specific measures are taken, like encrypting the data using its own hash as a key. In this case, identical files will remain identical after encryption, since the keys used in the process will be the same (the files hash). The users' password can also be used to encrypt his secret key (or private key), but it is important to use a different password from the one used in the authentication process, as the CSP could know the latter somehow (storing in plaintext, for example). This level is safer for users but it will be no longer possible to recover a lost password, as the user will be the only one knowing it. Losing his password means losing all encrypted data, because no one can recover it, and this might be a burden for some users. In this level, sending and receiving data requires less cryptographic operations than in the other levels. Only two operations are required: data encryption (on client side) before sending it, and decryption (also on client side) after downloading it. There is also the costs related to the encryption/decryption of the symmetric keys, but will not be counted as before. However, the main obstacle in the adoption of this level is the overhead that is placed on users due to the cryptographic key management. This is an important concern, since the whole security stands on the key. Fig. 32 illustrates the basic scenario of Level 3 while storing data on the cloud. For each file, a secret key is generated and used to encrypt it. After this process, this key is wrapped by user's public key, and sent along with the file to the cloud. Fig. 33 covers the recovery process. First, the file and the secret key, both encrypted, are downloaded. The encrypted secret key is decrypted using user's private key, followed by the decryption of the file using the resulting key. After this process, users can access their files.

- **Level 4 - Client-side encryption and manipulation of encrypted data by the server:** In this level, the cryptography is also performed by users, with the difference that it is no longer the classic cryptography approach. Instead, the homomorphic cryptography is used, allowing encrypted data to be processed by third parties without learning anything about it. As an example, users can store encrypted data in the cloud and later request their CSPs to perform some operation on it. The CSPs will do it without having the keys to decrypt the data and without getting any information about it. Even the operation results will not be revealed, and will be available only for users with possession of the cryptographic keys (LI *et al.*, 2012). One example of scenario where this technique can be applied is in database systems stored in the cloud (SHATILOV *et al.*, 2014). In this example, the database and each users' request are encrypted, in a way that the server can perform op-

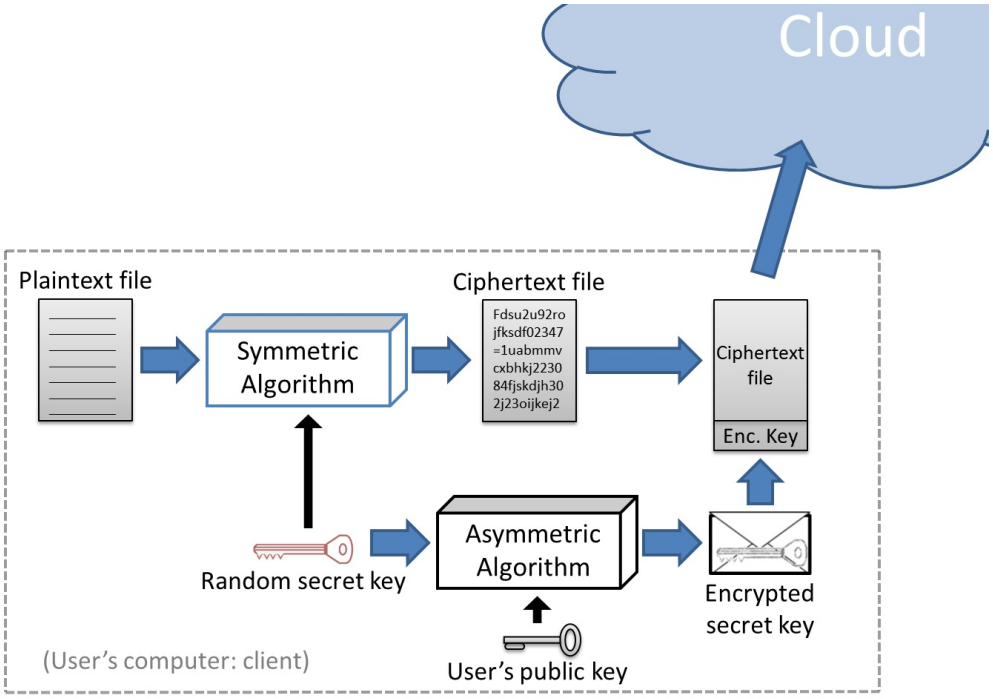


Figure 32 – Storing encrypted data in the cloud

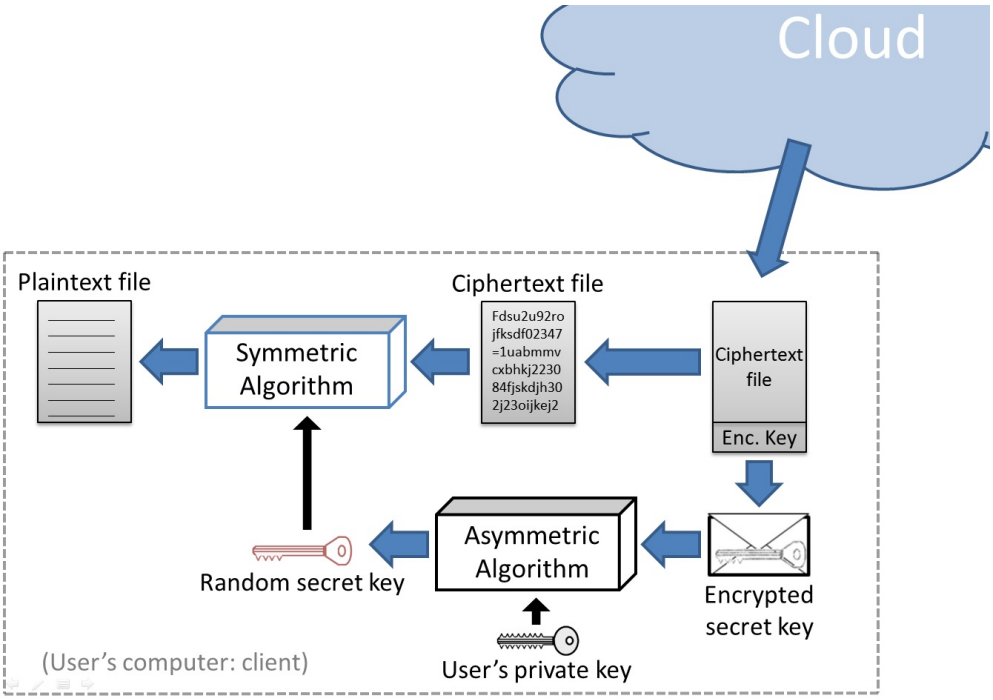


Figure 33 – Recovering and decrypting data from the cloud

erations and produce the results without knowing the request or even the results in clear text. The user will receive the results in encrypted form and with his key, he will open them. However, new advances are still necessary in this area before homomorphic encryption can be effective and widely adopted for storage in clouds.

In levels 3 and 4, there are two ways to use cryptography: Via browser or via application. Next, we will explain these two modes along with a brief evaluation of them.

- Via browser: This mode consists on using a browser to perform all the cryptographic operations. Some libraries already installed and configured in the browsers by default are used, which makes this scheme simple since users do not need to worry about installation and updating processes. Another major benefit for using a browser for this purpose is the usability, as users do not need to learn how to use a new tool to encrypt their data. Basically, in this model the server sends a script to the browser every time it requests a service, which will be interpreted and executed locally. However, there are several vulnerabilities in this model, as users' device has to execute an external code. A possible vulnerability is the existence of trapdoors in the code. As an example, before data encryption, the key could be sent to an attacker who broke into the script and was able to modify it. Users will hardly suspect about this attack, once the system is fulfilling its main function. Other forms of malicious code could be inserted into the script sent to users (PTACEK, 2011). However, if a secure channel is used, this attack can be more difficult to execute. Another way to minimize this problem is through the use of browser extensions. However, this is equivalent to the second mode described next.

- Via application: In this mode, an independent software is used and needs to be installed and configured properly in users' device. This software is usually supplied by the CSP itself, but it could also be from a third party, as PGP or any other, including the CPG proposed in this work. These tools are responsible for performing cryptographic operations locally on users data, before sending them to the cloud. They usually provide means to deal with users' cryptographic keys in a safe, but not necessarily, easy way. The major advantage of this model is that it allows users to audit and/or verify the signature of these applications, increasing the reliability of the services. If any malicious code is inserted in the application, its signature verification will fail and the user will be advised to stop using it.

4.1.1.4 Trust no one

When users lose or forget their passwords, they can use a common feature provided by most Internet services that offer a process to recover their passwords. With this feature, users must prove to the server that they are who they claim to be, and then the server sends them their password or allows the creation of another one. By choosing the password recovery, the server can replace the old one. It is a good practice to store users' password in coded form (a hash, for instance) and calculate a new code during the authentication phase to compare with the stored value every time users try to login into a system.

However, in cloud storage environment, when users look for services that provide security and privacy (mostly by the use of cryptography), the recover feature is not a good solution. In this scenario, if the CSP is able to restore user's password, it means that the CSP can find a way to access users' data, since they usually encrypt data using keys based on their owners' passwords. However, not having this feature means that if users forget or lose their passwords, it will not be possible to decrypt their files any more and all their data will be lost. This is the price that users have to pay in order to increase their privacy level.

The trust no one requirement aims to separate user's authentication password from the one used to encrypt user's keys. In this way, users will have two passwords, one used on the authentication process with the CSPs, and another, unknown by the CSPs, used to encrypt user's keys, which will not be stored anywhere and it will be used only on user's devices.

4.1.1.5 Confidentiality of file attributes

Another important security service that CSPs should provide in their solutions is the file attributes confidentiality. Unfortunately, this issue receives almost no attention from users who most of the time, are not aware of problems related to it. Moreover, CSPs normally do not implement this service due to the complex management required. This service is about hiding some attributes, as filename, size, ownership, last modification date, creation date, among others, that could be related to the file content and help attackers to choose their targets, if stored in plaintext form. Applying cryptography techniques over the file attributes (Fig. 34) could improve the security and privacy and add a barrier to attackers, who will have a hard time figuring out which files are important and worth stealing.

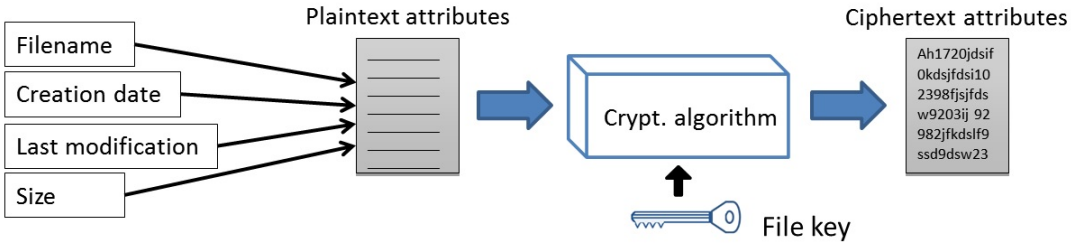


Figure 34 – Encrypting file attributes

4.1.1.6 Open Source

An important issue in developing a secure software is to make it available for community analysis. Most CSPs that provide cryptographic services do not have their codes available for review, which could raise suspicions about their software. This fact may decrease users' trust on the CSP since they have to believe the application provided to them is really secure. It is not enough for the community to have the CSPs word; they need to see the codes and how they

really works to trust on them. The security community has an important role checking if a software really does what its creator claims to. Besides, it is a good way to improve the software, through the detection of potential vulnerabilities in the project.

4.1.1.7 Software authenticity

This is a complementary requirement to the open source, discussed above. Sometimes the running code is not the same one available for review. For this reason, it is necessary to have ways to prove that the running code is indeed the same one generated by the source code available. Code signing is a technique that could be used to verify the software before installing it, making sure that it is the version published and revised by the community, which is probably the safest one. Fig. 35 illustrates the code signing process, which involves the code being hashed and the result encrypted with the developers (or company) private key. The result is normally attached to the software. Before the installation process, users can verify it using the developers (or company) public key (certificate). They need to calculate a hash from the software and compare it to the decryption of the code signature block, using the developers public key (Fig. 36). If both results are equal, the software is indeed the one published by its developer, and users can install and use it. Otherwise, the software suffered a modification since it was signed.

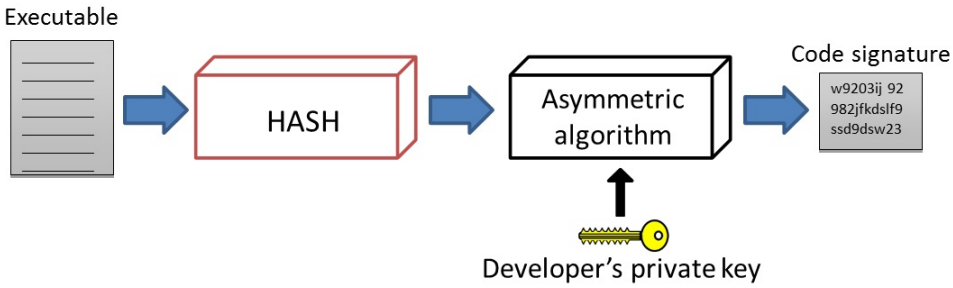


Figure 35 – Code signing process

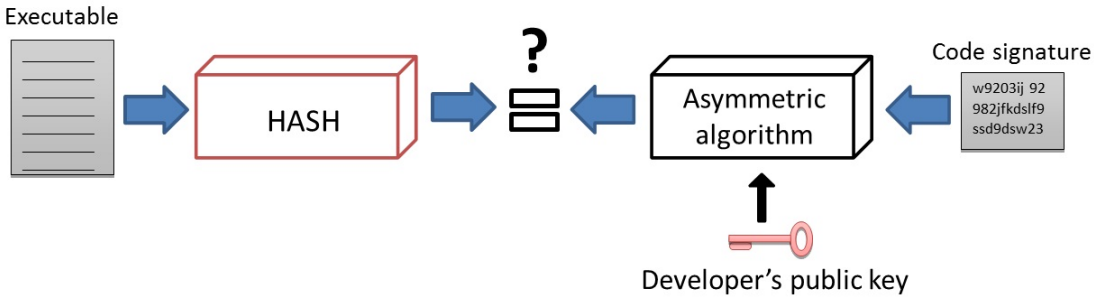


Figure 36 – Code signing verification process

As users will need the developers public key to verify the software, it is important that the developer uses a certificate belonging to a trusted PKI (Public Key Infrastructure). This

would prevent users from being misled by attackers that could change the developers public key by theirs and the software signature block from one created by them with a modified version of the software. In this case, when users download the software and the public key to perform the verification process, they would download the attackers public key and the modified software. Then, they would do the whole process and the results would be valid. A trusted PKI would allow users to check the ownership of a public key and avoid being fooled by attackers using false keys.

4.1.1.8 Multi-factor authentication

A possible solution to achieve a high level of trust in authentication could be by the use of multi-factor authentication. This technique consists in combining more than one authentication method to verify someone's identity. The methods used with this purpose are usually based on something the user knows (password), something he has (cryptographic device) or even something he is (biometric characteristics, like fingerprints, iris, voice etc.). Lee et al. (LEE I. ONG, 2010) describe a two factor authentication system using the usual login/password method combined with users' mobile phones. The login/password is the most well-known and accepted method to authenticate users, since it does not require extra equipment and people are already familiar with . Therefore, as mobile phones become popular, they are being used intensively as a second factor to improve the authentication process.

The multi-factor authentication can also benefit the cloud storage environment, since it will add an extra authentication phase to the system and make it more secure. Current CSPs that offer cryptographic systems derive a symmetric key from users' password and use such key to encrypt/decrypt their private key (or master key). This makes users' credentials the most sensitive point on the system. Even if the best cryptographic algorithms and the biggest possible keys are applied, the security of the system could fall down if attackers get users' password. Besides, getting their credentials is not a hard task nowadays, when users have a lot of different identities on the Internet. Every single service that requires an identity (a new login/password) makes them create and manage a new one. With so many pairs of login/password, users started using the same password in several places or even reducing their complexity, in order to make them easier to remember. However, this creates vulnerabilities and makes attacks easier to succeed. Malware and social engineering are other ways to get users' credentials successfully, but it will not be discussed here since it is out of the scope of this work.

Even the United of States government recognizes the fragility of the login/password model and started demanding the adoption of the two factor authentication to improve the security of its agencies in order to avoid certain attacks (STERNSTEIN, 2015). This action represents the vulnerability presented nowadays and that the adoption of multi-factor authentication technique by CSPs must be considered, in order to preserve users' privacy and the security of their data.

4.1.2 Usability

Another requirement indispensable in popular applications is usability. The dictionary defines it "as something available or convenient for use; capable of being used". According to the ISO 9241-11: Guidance on Usability (STANDARDIZATION., 1998), usability is "the extent to which a product can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction in a specific context of use". The focus of the discussion here will be on cryptographic applications. These applications are usually difficult to use and demand a lot of time for users to learn how to manage them. This usually involves some procedures and knowledge in the field. This factor could make users give up on using applications with low levels of usability, or at least avoiding them whenever possible. In order to get users' attention and enable them to use secure applications in their daily lives, these applications should be intuitive and easy to use. For the purpose of accomplishing this goal, a new requirement should be attended: Usability.

Usability in cryptographic applications

When cryptography techniques are introduced in a system, they usually make it more complex to use and manage by lay users (and even by experts in the field), due to the extra procedures needed. As a result, many users believe in the concept that a secure system is complicated and hard to use. In fact, this concept is almost always true, once security and usability are conflicting requirements.

For a software to be popular and adopted by many users, it needs to be easy to use and intuitive, everything that a cryptographic system usually is not. As an example, it could be mentioned the PGP (ALLIANCE, 2001). This is a well-known and established cryptography software which the security community reviewed and considered secure. However, its use is quite complex to lay users in cryptography, as they need to create and manage their cryptographic keys, distribute them to other users, among other procedures. Even some security experts have a hard time using this software daily because of the complexity involved. It is due to this and other reasons that the PGP is mainly used only by those familiarized with the concepts of data security.

Some CSPs have noticed this problem and started creating applications easier to use in order to attract more customers, but this is not a trivial task. A CSP that has stood out because of the simplicity of its application is the ProtonMail (PROTONMAIL, 2014). This is an email service that uses cryptography to protect users' emails. They focus on building an easy to use interface through users' browser, performing all the cryptographic processes on client side (Level of secrecy 3 - Via Browser). ProtonMail users normally do not perceive the introduction of cryptography, which works in background. Even though this service is criticized for working with the browser for performing all cryptographic operations, it is an evident effort by software developers to break the existing paradigm between security and usability.

An ideal software is the one looking at simplifying the cryptographic processes, in order to be easy to use even by lay users in security and cryptography. All the necessary procedures as key management and other ones, must be absorbed by the application and cause the least impact on the usability as possible. All of this is desired without compromising security. In this way, most users would enjoy the benefits of a higher privacy and safety, without the need to bear the high costs associated with the learning processes of the security tools available nowadays.

4.2 Related work

Solutions to improve security and privacy in cloud data storage could be found in the literature as discussed in chapter 2, but they also can be found in commercial products and even in specific applications built for other purposes. Most of commercial solutions focus on satisfying some of the requirements presented previously. Some of these solutions are also concerned with usability issues. In this section, we will cover some cryptographic solutions that were developed for, or can be adopted in, the cloud environment.

4.2.1 Commercial solutions

In this subsection we analyze some commercial cloud service providers available nowadays. All the providers discussed use cryptographic techniques in their solutions somehow, either in the communication channel or in data at rest. All of these CSPs will be analyzed according to the security requirements presented, with the goal of identifying possible gaps in their solutions. After the analysis we compare all solutions and point out the needs for further studies. For a detailed description of the cryptographic operations of these solutions, see appendix B.

4.2.1.1 Cloud service providers with cryptography protection

Onedrive (Microsoft) (ONEDRIVE, 2015) and Google Drive (DRIVE, 2015) are cloud service providers that offer data storage service for its customers for free, charging only for customized services, being classified as Level 1. Both providers use TLS (Transport Layer Security) to protect the communication between the servers and users' device. Users can recover their password whenever necessary (there is a specific process for that) and can share files. These CSPs also allow users to enable the multi-factor authentication, using a second method (users' mobile phone) to give them access to their accounts. Box (BOX, 2015) is also a CSP classified as Level 1, using only the TLS protocol to protect users data. However, Box does not have a free option for their services and only supports multi-factor authentication when users choose to authenticate with their Google accounts.

ownCloud (SCHIESSLE, 2013) is a free and open source project that allows users to build their own private cloud. There is an optional application called Encryption App that handles all the cryptographic services. If users choose to enable this module, all their data is encrypted on server-side, making ownCloud be classified as Level 2. Users are allowed to choose if they want to enable the recovery password process or not. If so, all users data is also encrypted with the administrator's keys, and in case the password is lost, the administrator can recover all users data and create a new password for them. It is also possible to use a secure communication channel (TLS) and to share data with other users. The ownCloud service does not allow data deduplication on its servers (OWNCLOUD, 2013).

arXshare (GMBH, 2015) is a secure and open-source solution for file synchronization and cloud storage, with a similar concept to ownCloud. Users can build their own servers based on PHP and collaborate with others in a secure way. All data is encrypted before transmission to protect users' privacy, and this CSP is classified as Level 3. Each user in the system has a unique pair of RSA keys (2048 bits key-size). When files are sent to be stored securely, they are split into chunks of one megabyte each and are zipped and encrypted, using a unique DEK (AES-256) and IV (Initialization Vector). Every directory has its own AES credentials, used to encrypt the DEK, wrapped by a KEK (Key Encryption Key) derived from users' password using a PBKDF2 (Password Based Key Derivation Function) algorithm (IETF, 2000b). Every time users share folders with a recipient, the credentials of that folder are encrypted using that recipient's public key. Also, filenames and all metadata, even file size, are encrypted. arXshare has a free version, but it is limited to just three users. It has its server component published as open source, but the client source code is not.

SpiderOak (SPIDEROAK, 2015) is a cryptographic CSP solution classified as Level 3, since all operations occur on client's side. In this solution, each folder have a key (KEK) used to wrap all the DEK employed to encrypt users data. This KEK is then encrypted with another KEK, derived from users' password using PBKDF2 algorithm. The system also encrypts users' filenames. Users are not allowed to recover their passwords when they lose it. If this happen, they lose access to all their data. However, they can share their data with others (paid version). SpiderOak claims to use deduplication on single-user mode and target-based approach.

Mega (MEGA, 2015) is a CSP similar to, and meets the same security requirements as, SpiderOak. However, Mega does not allow deduplication at any mode, since all the cryptographic keys are randomly generated. Mega provides its services in two ways: directly through users' browsers or an application installed on users' device. The latter mode is the one considered here. Another difference from SpiderOak, is that Mega allows data sharing, through asymmetric encryption. The symmetric keys used to encrypt users' files in Mega are smaller (128-bit key) than those in SpiderOak.

Wuala (GROLIMUND *et al.*, 2006) (MEISSER, 2011) is also similar to SpiderOak. The main differences are the use of a pair of cryptographic keys (asymmetric cryptography)

instead of a folder key (symmetric cryptography) for sharing data and the deduplication process is cross-user mode. There is no free version of Wuala. However, this CSP has discontinued its services. An announcement was made on August 17th to allow its customers to take back all their data. The remaining information still stored was deleted on November 25th, 2015.

Cyphertite (CYPHERTITE, 2015) also uses similar cryptographic schemes to those of SpiderOak. The main differences between them are: Cyphertite does not encrypt filenames, it is an open source project and users can share files.

Bazil (BAZIL, 2015) is a file synchronization implementation targeted to Unix-like systems. It aims to be an open source and free to use software. By the time of this writing, the project was still in development. It could be classified as Level 3, since all cryptographic operations are done on client side. Bazil divides each file in chunks, hashes it in a Merkle tree, and stores it into a Content-Addressed Store (CAS). All chunks are encrypted and authenticated with a NaCl secret box (BERNSTEIN *et al.*, 2012) (which is a library that provides all the core operations needed in a higher-level cryptographic tool). The CAS is mostly used to store content more than once without requiring extra space, while the Merkle tree is used to ensure data integrity and also to avoid that small changes to the file requires the uploaded of all chunks; instead, just the changed chunks are uploaded. Bazil allow the deduplication service, and for this reason, adopts the convergent encryption (WANG *et al.*, 2010). Users are not allowed to share data.

Carbonite (CARBONITE, 2015) is a CSP with focus on backup services only. The security provided in their solution is offered in two ways and could be chosen by users: auto encryption and private key encryption. The first one is automatic and takes place on server-side, using 128-bit AES and SSL protocol. The second mode, considered in this work, uses 256-bit AES and executes on users' device. All data in this mode are encrypted with a unique key, which is wrapped using a KEK generated from users' password. Carbonite does not allow users to recover their password in case it gets lost and does not allow data sharing.

BackBlaze (NUFIRE, 2008) is another CSP intended for backup services. As Carbonite (second mode) it does not allow users to share data. However, it uses two-factor authentication to give users access to their data and also combine symmetric and asymmetric encryption. Each user has a 2048-bit RSA key pair. For each backup session, a new random 128-bit AES key is generated to encrypt data and it is wrapped by users' public key. The private key used to decrypt all users data is protected by users' password, which is unknown by BackBlaze and not stored anywhere. As a drawback, when users restore files from Backblaze, they need to transfer their private encryption key to the server. This means that there is a small period of time when users' private key are with the CSP, an approach that is not secure and could lead to the leakage of users data.

Tresorit (TRESORIT, 2015b) is a CSP classified as a Level 3, where all users data are encrypted with a random symmetric 256 bit key (AES, CFB mode). This DEK key

is wrapped by the root directory key. Data integrity is protected with HMAC (HMAC-SHA-512), and all transactions (file uploads, modifications etc.) are authenticated with RSA-2048 signatures applied on SHA-512 hashes. The communication channel is protected by the TLS protocol. To share data, users need to send invitations to the recipients in order to establish a key agreement protocol. A suite of group key management protocols is used allowing a group of users to agree on a shared group key (LAM *et al.*, 2012). Recipient's data access in a directory structure is provided by Tresorit's Agreement Module, which can be RSA-based or Tree-based Group Diffie-Hellman (TGDH). Using a RSA-based module, there is a set of pre-master secrets for each user who is sharing the file, encrypted with the user's public key (RSA). To decrypt a file, users must supply their private key to the module, which will decrypt the pre-master with the provided private key and derives the symmetric key of the root directory. With this key, users can decrypt the root directory and access files and folders. The TGDH-based Agreement Module works similarly but instead of storing encrypted pre-master secrets, it stores users' Diffie-Hellman certificates. In Tresorit, users have their own X.509 certificate which contains personal data, as name and an e-mail address. Key revocation is also treated by this protocol, which uses the lazy re-encryption approach, where data is only re-encrypted when there are changes. Also, users' keys are protected by a KEK derived through their passwords using a PBKDF2 function (with 160-bit random salt and 10.000 iterations) (TRESORIT, 2015a).

Credeon (CREDEON, 2015) and Boxcryptor (BOXCRYPTOR, 2015) are cryptographic modules intended to be integrated with an existing CSP, like Google Drive, Microsoft OneDrive and so on, and they are responsible only to provide cryptographic services. They can be classified as Level 3. The main differences between these two services are: Credeon uses only symmetric encryption, does not allow password recovery or filenames encryption and is not an open source project, while Boxcryptor uses asymmetric encryption, allows password recovery and filenames encryption and is an open source project. PBKDF2 technology is used in both applications to derive a KEK from users' password to encrypt users' keys. However, none of them allow deduplication, since all the keys used to encrypt users' files are random and unique.

ProtonMail (PROTONMAIL, 2014) is another CSP but with a purpose different from the ones presented so far. This is an email service that offers cryptography in their solution to protect users' privacy. It can be classified as Level 3. Scripts are sent to the browsers which are responsible to perform all necessary operations, using the already existent libraries on users' machine. ProtonMail uses two passwords to grant users' access to their emails. The first one is used along with an username to give users access to their account, but with all their content encrypted. Then, with the second one, they can decrypt it. This latter password is intended to never leave users' machine. Also, ProtonMail does not allow users to recover their passwords in case they are lost (at least the second password related to the decryption of the cryptographic keys), and despite some libraries used in their project being open source, their core application are not and remains unknown by the community. To protect data, each user has a RSA key

pair, used to wrap the symmetric keys (AES-256) that encrypt/decrypt users' emails. However, the header of emails remains in plaintext to enable the servers make decisions regarding to the forwarding of messages.

Storj (WILKINSON *et al.*, 2014) (STORJ, 2016) adopts a different approach for storing data securely in the cloud. This is an alternative for a decentralized application using the concepts of the cloud storage services based on P2P (peer-to-peer) (BABAOGLU *et al.*, 2012), (BABAOGLU; MARZOLLA, 2014), (XU *et al.*, 2012a). This way, there is no third parties for controlling and handling users data. Users store data in the peers available on the network. Besides the store service, they can also benefit from contributing to the network on Storj. They can share hard drive space and/or internet bandwidth, and doing so are rewarded with cryptocurrencies as a payment mode. The Storjcoin X (SJCX) is a cryptocurrency developed to be used inside the Storj network to purchase bandwidth and storage spaces among users. This is an incentive to attract more users to the network. The more nodes present and contributing, the more copies of files will be distributed and available for downloading at the same time, increasing the speed of the network. Storj is divided into two application: MetaDisk and DriveShare. The first one is intended to upload files to the network, while the latter is to share hard drive space with the network. In MetaDisk (WILKINSON; LOWRY, 2014), when users upload data to the cloud, it is split into pieces, or shards, using Erasure Code techniques to create redundancy according to users' need; each shard is encrypted using a different key, along with its attributes. Then, all shards are transferred to different farmers (nodes that provide storage space and bandwidth to the network). This approach is classified as Level 3 and adopts a convergent encryption key, 128-bit AES-CTR, using Pycrypto (The Python Cryptography Toolkit), a library implementing a collection of cryptographic modules designed for python. Storj also implements other technologies, as blockchain, used as a distributed consensus mechanism to be able to locate files and check for their integrity, and to keep metadata files (file hash, network location of the shards, and Merkle roots).

4.2.1.2 Comparison of CSPs

We analyze the CSPs according to the set of security requirements defined in the previous section. In Table 13 we show the CSPs that meet the security requirements, represented by a green V, the ones that do not meet represented by a red X and the ones that partially meet, indicated by a blue P. The question mark in this table represents the absence of information, where we could not determine if the corresponding CSP meets or not the requirement.

All the features analyzed were extracted from the documentation available on the providers' website, forums, emails exchanged with the developers and also by trying some of them. However, even with the lack of documentation it was possible to build a model of how most CSPs handle the cryptographic services. As it is shown in the table, none of the CSPs meet all the security requirements. There are some requirements that are not fulfilled

Table 13 – Comparison of main CSPs features

CSPs	Cryptographic keys security	High Level of data secrecy	Trust no one	Confidentiality of file attributes	Secure deduplication	Open Source	Software authenticity	Multi-factor authentication
Onedrive	X	X	X	X	X	X	X	V
Drive	X	X	X	X	X	X	X	V
Box	X	X	X	X	X	X	X	V
ownCloud	P	X	P	X	V	V	X	X
Bazil	P	V	V	X	X	V	X	X
Cypherite	P	V	V	X	V	V	X	X
BackBlaze	P	V	V	X	V	X	X	V
Credeon	P	V	V	X	V	X	X	X
ProtonMail	P	V	V	X	V	X	X	X
Carbonite	P	V	V	X	?	X	X	X
Mega	P	V	V	P	V	X	X	X
Wuala	P	V	V	P	X	X	X	X
SpiderOak	P	V	V	P	V	X	X	X
Boxcryptor	P	V	V	P	V	V	X	X
arXShare	P	V	V	V	V	V	X	X
Storj	P	V	V	V	V	V	X	X
Tresorit	V	V	V	X	V	X	X	X

entirely by neither of them and others that just few ones did, like the cryptographic key life cycle control, file attributes encryption, source and executable code signatures and multi-factor authentication. Most CSPs meet only partially the requirement regarding the control of the key life cycle, while only one CSP is concerned with the encryption of file attributes besides the filename. Some of the providers have their codes available, but do not offer ways to prove that the application running in their servers is indeed the result of compilation of the available code. Only the CSPs classified as Level 1 of secrecy and only one in Level 3 offer more than one method to authenticate users (two factor authentication).

Despite all the security features adopted by them, some problems still remain. Most CSPs do not offer the necessary level of security to guarantee users' privacy, having a lot of space for improvements with respect to the users' privacy point of view. Further studies are required in order to improve the privacy in the cloud data storage environment.

4.2.2 Other solutions

In this subsection we present an analysis of other forms to achieve security and privacy in cloud data storage. We introduce two well-known applications and analyze how they work to protect users' privacy, followed by a set of possible improvements.

4.2.2.1 PGP

PGP stands for Pretty Good Privacy. It is a security software package developed by Philip Zimmermann that supports strong symmetric and asymmetric encryption, being used to provide privacy, confidentiality and authentication of data, mostly in network communication. There have been many versions of PGP: OpenPGP (ALLIANCE, 2001) (standard specified by RFC 4880 (CALLAS *et al.*, 2007)) and GNU Privacy Guard (GnuPG or GPG) with its variants GPG4Win and FireGPG for Firefox. Also, there are some versions to be integrated with browser to encrypt emails. Commercial versions are also available. PGP provides several services for its users, as data encryption, digital signatures, data compression, among others. In order to use these services, users need to install the PGP software through a Wizard (GPG4Win). At some point, they are required to create the encryption keys (there are different cryptosystems and sizes for choice). They also can limit the key lifetime. After that, they are asked to create a password to protect their keys, which is expected to be long (at least eight symbols) and complex, but users have to remember it since a recover is impossible in case it gets lost. PGP provides an interface for its users manage their cryptographic keys and also the public keys of their friends, co-workers or business partners, which can be imported into the system. The export option is also available as they have to share their public key with others (KOŚCIELNY *et al.*, 2013).

According to the RFC 4880 specification, the PGP combines symmetric and asymmetric encryption. When encrypting a message, email, file etc., firstly the object is encrypted by a unique and random symmetric key (DEK) which is later wrapped by the receiver's public key (Fig. 37). The object is usually compressed before the encryption process. Besides the confidentiality achieved by the encryption, data could also be signed digitally. In this case, the signature is attached to the object and then both are encrypted. In the decryption process, seen in Fig. 38, users first need to provide their password to the system, which will convert it in a symmetric key (KEK) to unwrap the DEK. After that, the object is decrypted. If the object was signed, the verification process using the sender's public key is performed, followed by a decompression process whether necessary.

PGP also adopts the concept of Web of Trust, a decentralized alternative to the centralized trust model of a public key infrastructure (PKI). Instead of relying on a certificate authority (CA) to endorse someone's public key, users can trust on others to do it, avoiding all the expenses of a PKI. PGP generates certificates for its users, which contain their public key and also some personal information. The certificates are signed by a trusted person. Users will only use the keys that have been signed by trusted participants of them. A key could be signed several times, in order to create a large mutual trust group (KOŚCIELNY *et al.*, 2013).

4.2.2.2 TrueCrypt

According to Koscielny C. et al. (KOŚCIELNY *et al.*, 2013), TrueCrypt is a free and open-source security software mostly used for disk encryption, available for multiple platforms,

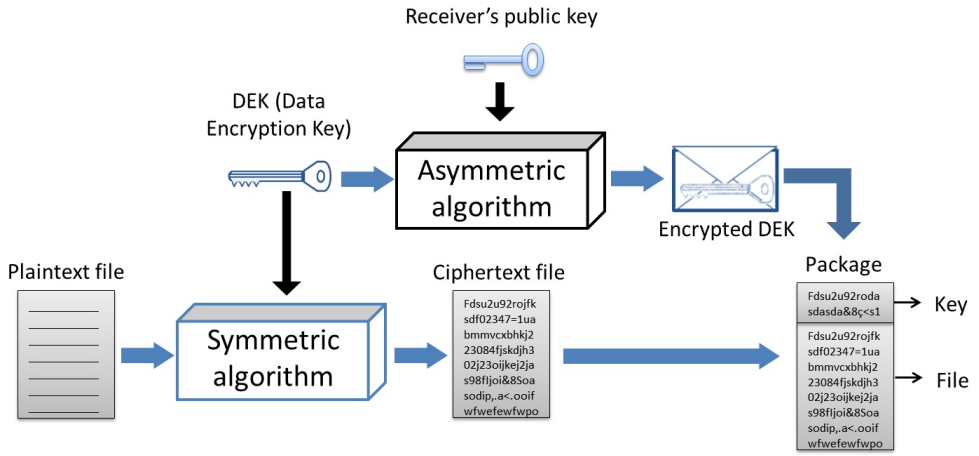


Figure 37 – Encryption process on PGP

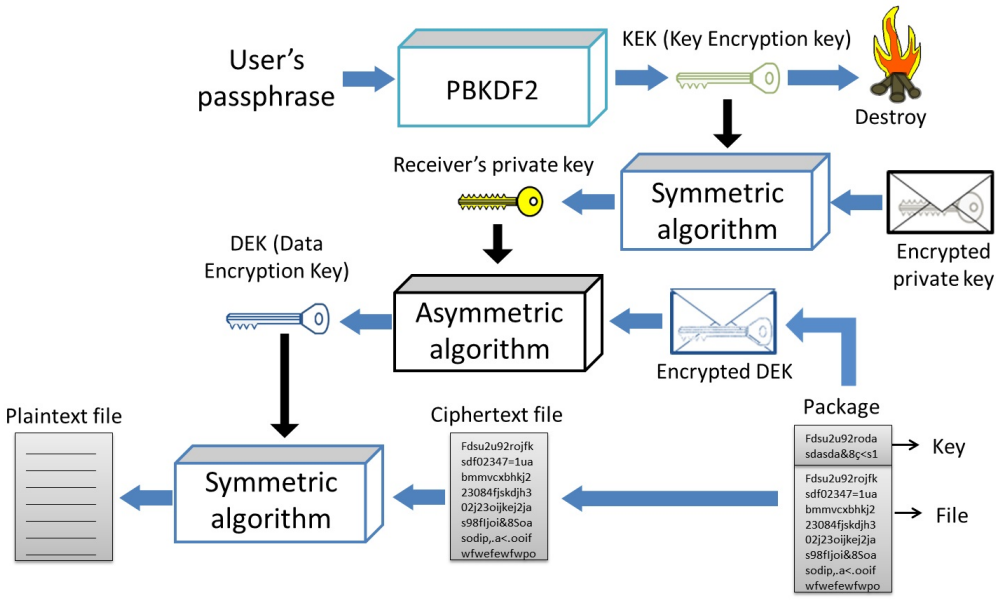


Figure 38 – Decryption process on PGP

as Windows, Mac OS, and also Linux. It could be installed through an easy Wizard guide (standard installation) or be used in a portable version of it. Users are able to encrypt single container-files or an entire disk partition. The root partition, where the operating system is installed, can also be encrypted using TrueCrypt. The encryption process is automatic, real-time (on-the-fly) and transparent.

When users encrypt an entire partition and want to move or copy data in it, data will be encrypted during the copying process. To move or copy data to another partition, the process goes the other way. If data is moved from an encrypted partition to another, it is first decrypted and then re-encrypted using different keys. Before encrypting data, users need to choose a password, which is recommended to be long and strong enough for the purpose of avoiding brute force or dictionary attacks. In order to read encrypted data, they have to provide the correct password to the system.

TrueCrypt also allows its users to create hidden volumes, which should consist of hidden and also some neutral data, chosen in the creation of the volume. Users can create an encrypted file system and protect it with a password. Inside it, they can create a hidden area to protect some files, which will be accessed by a second password. Unless users reveal the existence of the hidden area (and also the password) to an adversary, it should be impossible to determine its existence. However, some applications can leak significant information outside of the hidden area (CZESKIS *et al.*, 2008) that could allow attackers to guess it.

In order to enhance its security, TrueCrypt allows users to adopt Keyfiles. This is an optional form that enables two-factor authentication and adds a second line of security to users data. This could be achieved by any file on users' computer or one stored in an external medium (pen drive). TrueCrypt reads part of the chosen file and uses it as a password in the encryption process. Accessing encrypted data without the password and the Keyfile will be impossible. There can be many Keyfiles for an object, but in this case all of them will be necessary in the decryption process. If one of them is lost, damaged or modified, the decryption will fail (KOŚCIELNY *et al.*, 2013).

Broz and Matyas (BROZ; MATYAS, 2014) highlight that TrueCrypt uses different symmetric algorithms to encrypt data. Users can choose among AES, Serpent, and Twofish. The mode used in the encryption process is the XTS. When encrypting data, TrueCrypt creates a container, consisting of an encrypted header with metadata (key and other relevant information) and the encrypted data in the remaining space. Header and data are encrypted using the same algorithm, but with different keys. Data is encrypted using a symmetric key which is stored in the header, while the header is encrypted using a key derived from a salt (random data stored in the header) and user's password using the PBKDF2 algorithm. The simplified encryption and decryption process are illustrated in Figs. 39 and 40.

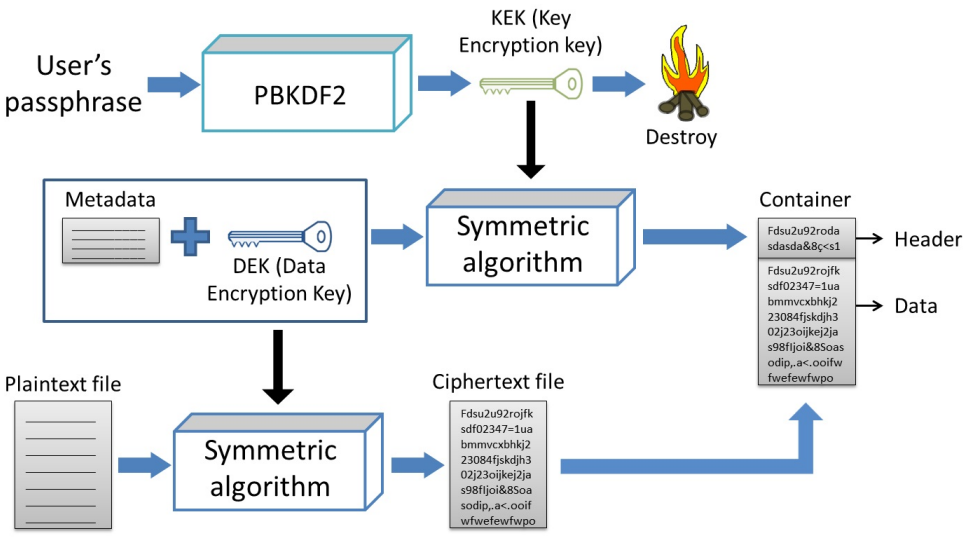


Figure 39 – Encrypting data with TrueCrypt

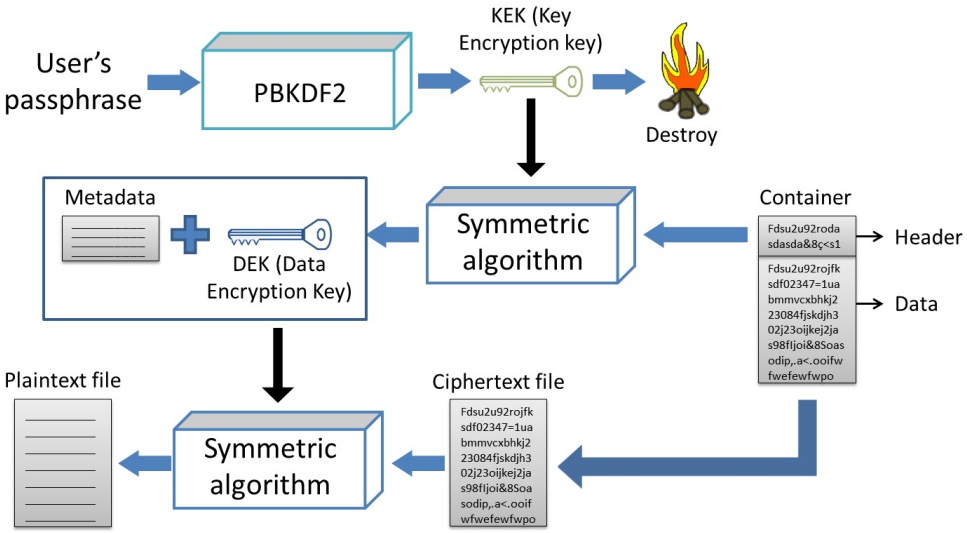


Figure 40 – Decrypting data with TrueCrypt

Despite being a good tool for protecting users data, TrueCrypt project was ended in May, 2014, and said to be insecure (TRUECRYPT, 2014). However, there is an initiative dedicated to the development of the next TrueCrypt called TCnext ((TCNEXT, 2015)). Also, there are some projects derived from it, as VeraCrypt and CipherShed. The first one is a free and open source project aimed to disk encryption, based on TrueCrypt version 7.1a (VERACRYPT, 2015). The authors claim to have fixed most security issues found on TrueCrypt (JUNESTAM; GUIGO, 2014). The latter one is also a free and open source project for disk encryption, but at this time it is still in development (CIPHERSHED, 2015).

4.2.2.3 Operating System's Built-in Encryption

Besides PGP and TrueCrypt, users also have the possibility of using cryptographic applications built for specific operating systems. In Windows, users can enable BitLocker (MICROSOFT, 2015) to get full-disk encryption. Mac OS X offers for its users the FileVault disk encryption (FILEVAULT, 2015), while in Linux there is a variety of encryption technologies, but modern distributions often enable full-disk encryption though LUKS (Linux Unified Key Setup) (LUKS, 2015). Other solutions include AES Crypt, AxCrypt, DiskCryptot, EncFS, Secrecy, among others, which are beyond the scope of this work.

4.2.2.4 Discussion

PGP, TrueCrypt and other similar solutions could be an alternative for cloud data storage, even though they were not built for this specific purpose. They require a lot of interaction and manual steps for users to store their files in the cloud. Users have to encrypt their files and upload them to the cloud. In the decryption process, the file must be downloaded and then decrypted. All of this process could be too hard for lay users to handle, especially when

they are responsible for the cryptographic key management. Another complication is about encrypted folders. Accessing a specific file inside a folder would require the download and decryption of the entire folder, increasing the processing time according to the folder size, as it happens in TrueCrypt. The volume created would be stored in the cloud and in order to access any file inside it, users would have to download the whole content and decrypt it.

Some users, especially lay ones, would rather not store data in encrypted form in the cloud due to the complexity involved using the tools. An application made specifically for cloud data storage in a way that makes all the processes transparent to users, would facilitate its adoption. Also, this would be even a motivation for starting using this kind of technology. Such application should remove all the burden caused by the cryptography and require less interaction from users, in a way that they would not even realize that cryptography is being used.

4.3 Conclusions

In this chapter we presented and discussed some requirements for a secure, reliable and user-friendly cloud data storage application. These requirements were divided into two groups: the ones related to the application security and the one associated to the application usability. The first group of requirements can protect users' privacy and provide security for their data from attacks and threats they may face while using cloud data services. The latter aims to make the application more easy to use, requiring less effort and specific knowledge from users. All the complexity is absorbed by the software. The main goal is to leverage the protection offered by security applications to as many users as possible, even for those with minimum knowledge of cryptography.

We also presented in this chapter some solutions to mitigate common risks and improve users' privacy in cloud environment. We divided and presented the solutions in two lines: commercial and others. The first one is related to commercial solutions available nowadays to protect users' privacy. The CSPs discussed in this chapter saw an opportunity regarding data protection in cloud and started offering easy-to-use solutions to store data in the cloud securely. However, some improvements are still necessary. None of the CSPs fulfil all the security requirements, as showed in the analysis done where the CSPs' features were contrasted with requirements.

The second line was related to other solutions that could be integrated in the cloud environment to protect users' privacy. The solutions presented and analysed were not built specifically for cloud data storage use. For this reason, they require much more effort from users than the available commercial solutions presented before. However, even though they have low levels of usability for this environment, they still are an option and some are used by experts in cryptography due to their reliability and flexibility.

5 Cloud Privacy Guard (CPG)

In this chapter we propose a cryptographic software called CPG (Cloud Privacy Guard) to overcome the most common security risks in cloud data storage. CPG satisfies some important security requirements in clouds and is based on good usability principles. Moreover, we present a comparison of CPG with other applications with similar purposes and a discussion about its limitations and its future.

5.1 Introduction

There are many systems and tools created to improve users' privacy in the cloud environment. However, most of them still lack features to be a secure and reliable applications, as showed in previous chapters. Aiming to fulfil the gaps found in the available solutions and also taking into consideration the techniques proposed in the literature, we propose a new application to mitigate some security problems on cloud data storage and also to address some relevant security requirements. The application, called Cloud Privacy Guard (CPG), was designed with the purpose of being secure and reliable, but also easy to use. It reduces the overhead caused by the introduction of the cryptography due to the extra procedures necessary, and aims to be adopted by most users in daily activities. We will explain how this application works, how it manages the cryptographic keys and how it addresses the security requirements.

5.2 Objectives

With the purpose of getting a higher security and better privacy in cloud computing, solutions that mitigate common risks and threats in this technology are necessary. CPG is a cryptographic solution aiming to bring the advantages of cryptography to all kind of users, even the ones with no knowledge in this field. This is achieved by a user-friendly interface that requires less effort from users when storing their files in the cloud securely. CPG is an automatic solution that employs cryptographic techniques in a way that users do not even realize they are using them, since all the processes are transparent and run in background. They also do not need to worry about key management or other issues incurred by cryptography in order to protect their data and privacy. The solution was conceived using interface techniques already understood by most users, as the drag and drop model.

One of the main purposes of CPG is to give users exclusive access to their data. It adopts symmetric and asymmetric cryptography to mitigate some of the confidentiality problems on cloud data storage. It also allows users to share their private data with others, through certificates. In order to improve users' privacy and become a secure and reliable cryptographic

system, CPG tries to fulfil all the security requirements discussed previously. Another objective of CPG is to be used with any kind of CSP that works with a specific folder to synchronize data, where users only need to put their files in this folder to have it sent to the cloud. CPG does not provide the storage infrastructure for users storing their data, but it is a software that can be integrated with most CSPs available. Examples of CSPs supported by CPG are Google Drive, Dropbox, Microsoft OneDrive, and ownCloud.

CPG allows its users to access their files from different devices. In order to do that, they just need to configure the application in each device. If the right passphrase is given, the system will automatically find and open users' key pairs obtained from the cloud, where they are stored in encrypted form. Also, the CSP client needs to be installed and configured in the intended device.

5.3 Requirements to be met by CPG

This section will explain how CPG intends to meet the requirements presented and discussed in Chapter 4 to be a secure, reliable and easy to use application.

5.3.1 Cryptographic keys security

The cryptographic keys security is an important requirement and must be addressed carefully, since all the security of cryptography software relies on the protection of its keys. CPG adopts the key life cycle control in order to fulfil this requirement. CPG uses two types of cryptographic keys: a symmetric key for encrypting data (DEK) and an asymmetric key pair for each user to protect the DEK and also to enable data sharing. The first is dealt by adhering to the key states recommended by NIST. For each new file, a new key is generated. Also, every time files are modified, new file keys are generated to re-encrypt them, avoiding access to new versions of it by users that are no longer permitted and also as a measure to mitigate the risks of attacks. The older keys are destroyed. However, the pre-activation and deactivated states are not adopted, since there is no need for performing proof of possession or key confirmation (process used in the pre-activation state). The keys are also destroyed when files are modified or destroyed and are not used again even for further decryption, not being necessary the deactivate state. On the other hand, a different treatment is done with respect to the asymmetric keys. These keys are outsourced to a PKI responsible for issuing certificates. This PKI will control and limit the validity period of each key pair and also the key stages, besides other related issues.

Another security measure adopted by CPG is related to the time user's private key remains in memory. Despite CPG does not propose a security alternative to address this problem, CPG mitigates it enforcing that users' cryptographic keys do not remain in memory for a long period of time but just during the period they are needed.

5.3.2 Secure deduplication

As seen before, deduplication is useful for CSPs, but can be insecure for users. Only a secure deduplication process should be allowed, but CPG has no control over CSP internal procedures. Because CPG uses unique and random keys to encrypt files, the deduplication at CSP servers will not be possible and its problems will be avoided.

5.3.3 High level of data secrecy

Based on the description about the levels of secrecy, it can be seen that Level 3 is the practical one with the best privacy preserving features when compared to the other ones, since all the cryptographic operations are performed on users' device. In this mode, users are in control of the whole process and are the only ones to have access to their data. However, this could be a drawback since it is a burden too heavy for some users to handle, as they will be the ones responsible for the cryptographic key management and other issues. Based on this, the proposed CPG adopts Level 3 concepts in its design, but builds a model where most of the key management procedures are done by the application in order to release users from this burden. The only thing they will need to do (after initial configuration) is to remember a passphrase, which will be used to give them access to their keys and hence to their data in a transparent form.

5.3.4 Trust no one

The CPG application uses the concept called trust no one since it will improve data security and users' privacy. Users are the only ones who know the passphrase needed to give them access to the securely stored data and thus another passphrase (or password) is needed to identify them to the CSP. One important consequence of this requirement adoption is that the CSP will not be able to help users in case they loose the passphrase used to encrypt files.

5.3.5 Confidentiality of file attributes

This requirement is also addressed by CPG through the encryption of the file attributes, like timestamps, group, ownership, filenames etc., in order to protect user's privacy. The same DEK used to encrypt the file content is used to protect its attributes, which are collect into another file that is stored together with the corresponding data file.

5.3.6 Open Source

CPG has its source code opened to the community for reviews. This practice helps the detection of possible flaws and opens space for improvements, coming from community reviews and suggestions.

5.3.7 Software authenticity

CPG adopts the source and executable code signatures in order to meet this requirement and allow users to verify the authenticity of the application installed in their devices. The signatures are based on certificates of well known and reliable Certificate Authorities.

5.3.8 Two-factor authentication

CPG adopts the multi-factor authentication using something users know (a passphrase) and something they have (mobile phone). When users choose the two factor authentication (2FA), the system will add an additional step requesting the second factor in the authentication process. In the activation process of this new authentication factor, it is demanded that users go through a registration phase, where they will follow some steps to configure their device and also the application.

Google Authenticator, which uses the TOTP protocol (*Time-Based One-Time Password Algorithm* (IETF, 2011)), was the choice for the initial stage of tests with CPG with 2FA because, despite its vulnerabilities (DMITRIENKO *et al.*, 2014), it is easy to install, configure and use. In the first step of the registration phase, CPG will generate a sequence of 16 characters that need to be inserted in users' mobile phone, which is supposed to have the *Google Authenticator App* installed. The generated code can be inserted into Google's app in two ways: manually or with a QR code.

In the authentication process, the *Google Authenticator* will generate a small code (6 characters) every 30 seconds that needs to be inserted in CPG application when requested (there is an additional tolerance time to type the code). If the code provided is the same as the one generated by the application, users have their access granted to CPG; otherwise, users are rejected and, after three wrong attempts, the process is ended. To try again, users must initiate CPG and start the whole process all over again. The entire authentication process using the second factor is summarized in Fig. 41.

5.3.9 Usability in cryptography applications

CPG aims to simplify the cryptographic processes to its users in order to be adopted even by lay ones. It proposes to use concepts already adopted by most users, as drag and drop objects to a specific folder, for example. This is the same model as the one adopted by popular CSPs available nowadays, like Google Drive, OneDrive, Dropbox, etc., with the purpose of making CPG's use experience simpler, requiring less interactions between users and application.

Most of the necessary security procedures as key management and other ones are absorbed by CPG to cause as little impact on the usability as possible. All cryptographic keys, for example, are protected by just one passphrase, the only one users have to remember to use

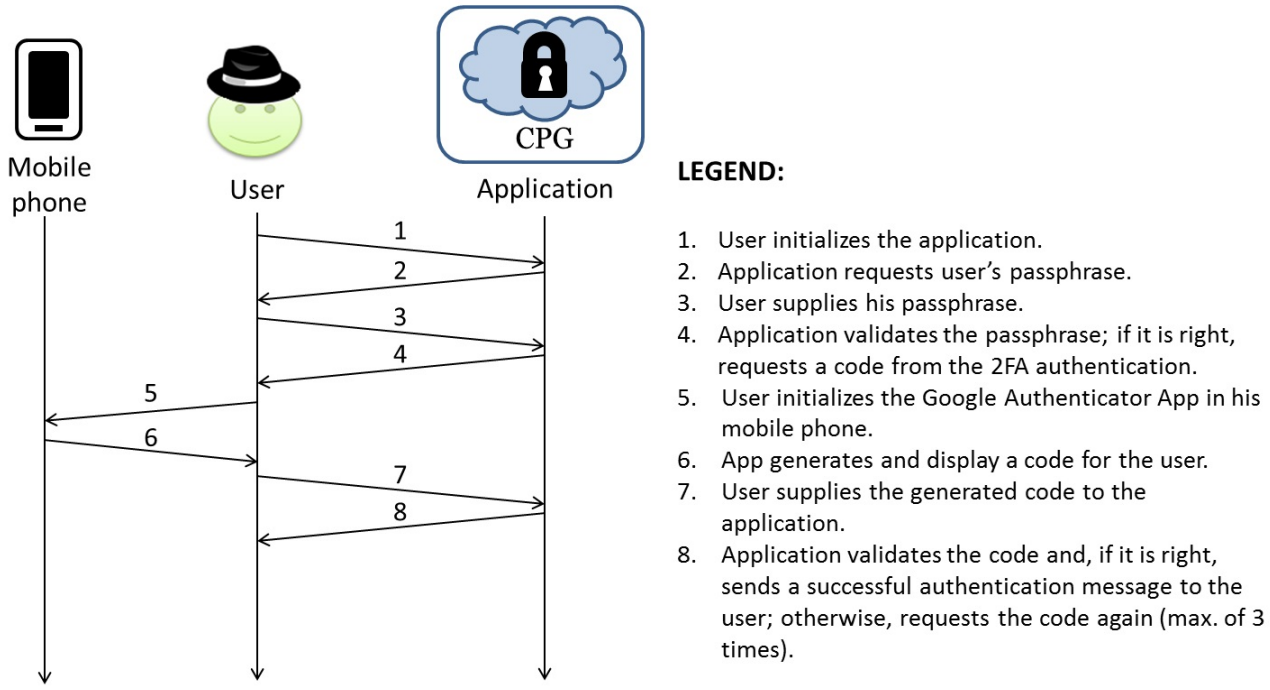


Figure 41 – Authentication 2FA

the application. In this way, most users would enjoy the benefits of a higher privacy and safety for their data, without having to submit themselves to high costs associated to the learning and adoption of traditional security techniques.

5.4 How CPG works

This section gives an introduction about CPG operation. It covers topics such as first use, configuration settings and authentication, among others. For more information about CPG, as its main use cases, see Appendix C.

5.4.1 First use

Before users start CPG for the first time, they need to create and configure an account in any CSP that can be integrated with CPG. They also need to download and install the client application responsible for synchronizing data from their devices with the cloud servers. After that, they need to install and configure CPG in their device. When it is initialized, it requires the creation of a passphrase used to protect users' cryptographic keys. Then, the application creates a pair of keys (or allows users to select one already existent) and requests users to fill out the configuration settings.

Every time the application performs a secure task, it requests users' passphrase, unless it was supplied some moments ago (there is a *time to live* parameter for a keyed passphrase).

The CPG application starts every time users' device initializes. Users can see it running through the system tray, represented by an icon that allows them to interact with it (Fig. 42). They can close the application or even change the configuration settings through this icon.

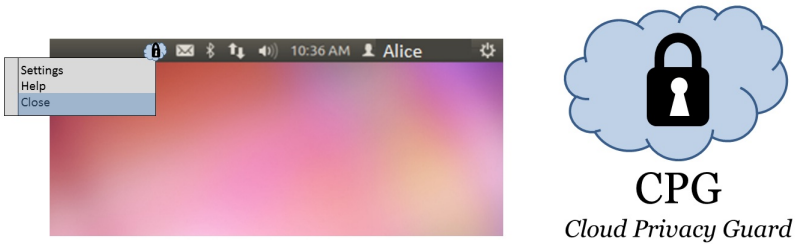


Figure 42 – CPG icon on the system tray

5.4.2 Configuration settings

Users need to configure CPG before using it. This step is done after the first time the system is initialized. Users can also change the application setting at any time by accessing the configuration options while CPG is running, through the icon available on the system tray. The configuration process is short and simple. Users only need to choose the folders used by the application, which are the *Encrypted Files* folder, used to synchronize data between users' computer and the cloud servers, and the *Secret Files* folder, used to define the data to be encrypted. There is also a path indicating where the cryptography keys are placed, which is defined automatically when the keys are created (users can modify it if necessary). It is important to notice that the *Encrypted Files* folder can be the root of the cloud or just a specific folder inside of it with the purpose of storing encrypted files. CPG will only handle encrypted files in this folder and ignore other ones. Fig. C.3 illustrates the CPG configuration screen.

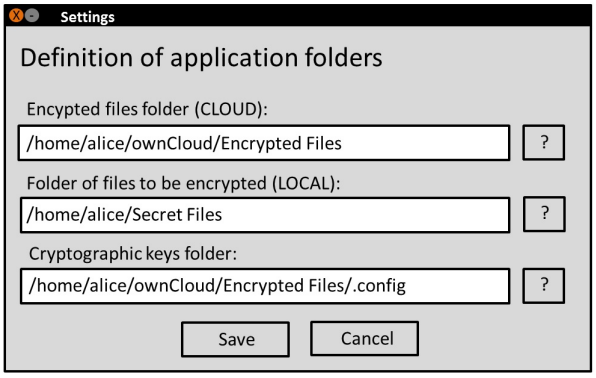


Figure 43 – CPG configuration window

5.4.3 Drag and Drop model

CPG adopts the well understood drag and drop model, used by most CSPs available nowadays. This is a choice to create an easy to use interface, requiring less effort and interaction

from users when using CPG. To send encrypted files to the cloud, users just need do drag (or copy) them into the *"Secret Files"* folder, as shown in Fig. 44. CPG will be monitoring this folder and any modification on it will be detected and the appropriate procedures started. The CSP client will be responsible for synchronizing the files with the cloud servers.

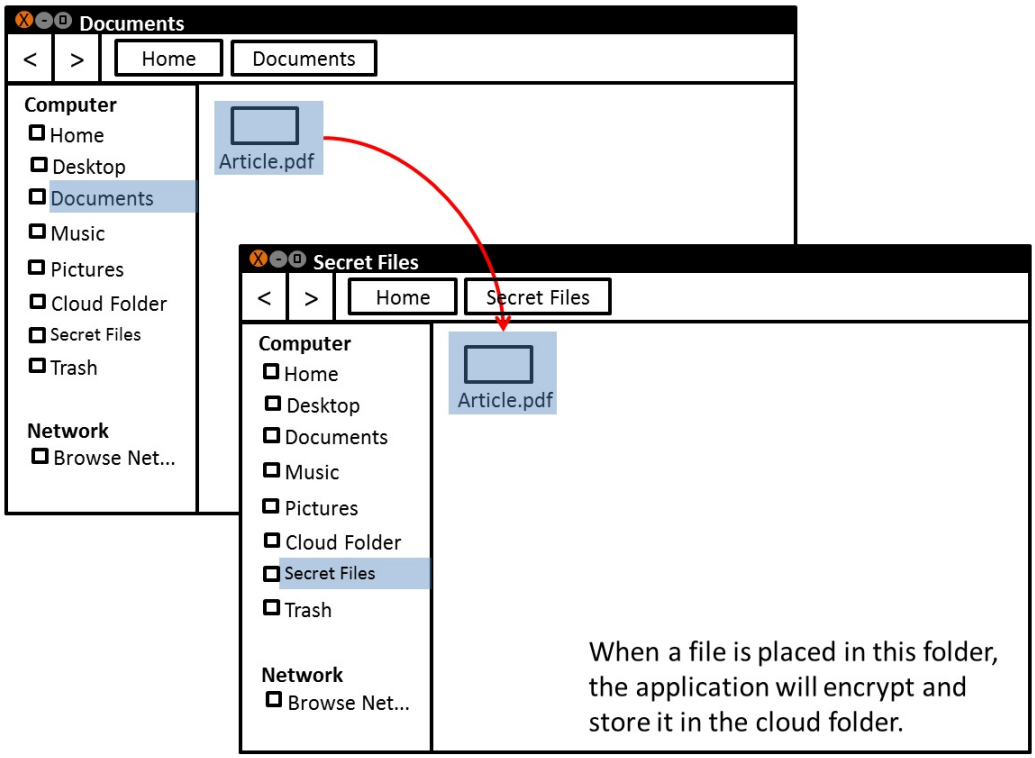


Figure 44 – Drag and Drop model - How it works.

5.4.4 How to send encrypted files to the cloud

CPG was conceived to require as little interaction with users as possible to send encrypted files to the cloud. The CPG working process could be better explained using Fig. 45, followed by an example of how sending files securely to the cloud.

In this figure, user Bob wants to store his files in the cloud securely. He first initializes CPG (step 1), which requests his passphrase. After supplying it (step 2), the application goes to a validation process. If Bob's passphrase is correct, CPG starts; otherwise, Bob have three attempts to supply the correct password before the application is ended. If the access is granted, CPG starts and synchronizes data in the *Secret Files* folder with the *Encrypted Files* folder. Besides, it also starts monitoring these two folders. Any modification on them, as file creation, deletion or modification will activate CPG. When Bob stores a new file (step 3), CPG will be activated and starts encrypting it using Bob's certificate (step 4). In step 5, the encrypted file is stored in the *Encrypted Files* folder. Step 6 is performed by the CSP client, responsible for the synchronization process between user's device and the cloud servers. Step 7 corresponds to

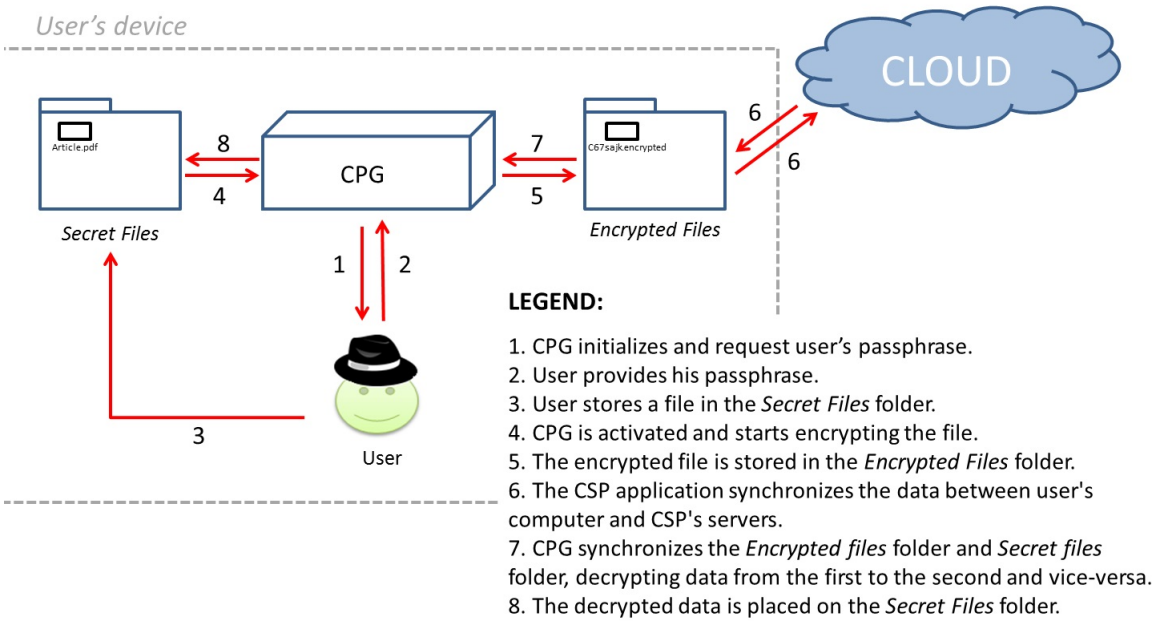


Figure 45 – CPG working process

CPG synchronizing the *Encrypted Files* and *Secret Files* folders, decrypting data from the first to the second. The decrypted data is then placed on the *Secret Files* folder (step 8).

5.4.5 How to decrypt files

The decryption process is not performed very often by CPG. Although all data in *Secret Files* folder is encrypted before it is sent to the cloud, a version in plaintext will remain in this folder on users' device. He will use this version and every change will be noticed by CPG, which will encrypt it and replace the encrypted version in the cloud by this new one. The decryption process will be needed only in two cases. The first one is when users access their data through a different device. When they install and initiate CPG in a new device, all existing encrypted data will be decrypted and placed in the *Secret Files* folder of this device. The second case of decrypting data is done when a user shares data with other ones. In this case, the shared information will be available for use by the recipients and, in the moment it appears in their space, CPG will detect, decrypt and store a plaintext version of it in their *Secret Files* folder.

5.4.6 How to share files

Public key infrastructure is a well-established technology used to solve some security problems with asymmetric cryptography; however it comes with a high cost. In Brazil, an alternative for minimizing this cost in educational and research institutions was the creation of an educational public key infrastructure called ICPEdu (RNP, 2015). Students, teachers, researchers and the staff people that are part of educational institutions affiliated to RNP (Brazilian national research and education network) can get freely ICPEdu certificates. CPG makes use of

this kind of certificates to encrypt files and to share them among users. As ICPEdu certificates are technically based on X.509 format, any other certificate in such format will be compatible with CPG.

In order to share encrypted files, users need to share the folder containing the intended files through the conventional CSP interface, and then put in such folder the certificate of every person that shall be able to open those files. For security reasons, such certificates must be encrypted by the file owner private key, avoiding that someone else put undesired certificates on the same shared folder. The application will encrypt the data with the certificates present in the folder. Removing access to a shared file from a particular user, requires the removal of that user's certificate from the folder. When new data is stored in the folder or a file is modified, it will be encrypted only for those users who still have their certificates presented in the folder, denying access to this new information to the ones whose certificates were removed. This technique is referred as lazy revocation (KALLAHALLA *et al.*, 2003).

5.5 Cryptographic process

The proposed CPG application adopts both symmetric and asymmetric cryptography, taking advantage of the best properties of each one. The symmetric algorithm chosen was the 256-bit AES due to strength, reliability and good performance in the encryption/decryption of large amounts of data. Indeed this is one of the most used algorithms in the solutions studied. CPG creates a random and unique symmetric key for each file (DEK), which is used to encrypt users data. To protect the DEK, the asymmetric RSA algorithm was chosen. Despite some authors preferring to use the Elliptic Curve Cryptography (ECC) (YIN *et al.*, 2014) (KUMAR *et al.*, 2012) due to its better performance, we preferred to stick with classical RSA solution in the first version of CPG. Other alternatives that could present better performance than RSA will be studied for future versions of CPG.

The public key cryptography was chosen to be implemented in CPG since it facilitates data sharing and could rely on a PKI (Public Key Infrastructure), which is a way to control the key life cycle of one of the most valuable keys belonging to users: the private key.

Each CPG user has a 2048-bit RSA key pair (public and private keys), used to wrap the DEKs. The public key is stored in plaintext form in a certificate and the private key stored encrypted with a symmetric key (KEK) derived from user's passphrase by the PBKDF2 algorithm. When a file needs to be encrypted, a new and random symmetric key is generated and used to encrypt it along with its metadata. After this process, the encrypted file key is wrapped by user's public key. The encrypted file, metadata and key file are put together in a container, which is stored in the cloud (Figs. 46 and 47). To decrypt the file, the user needs to provide his passphrase to derive the symmetric KEK intended to decrypt his private key, needed to unwrap the encrypted key file and finally open the encrypted file and metadata. The latter is used to

restore the file attributes (Figs. 48 and 49).

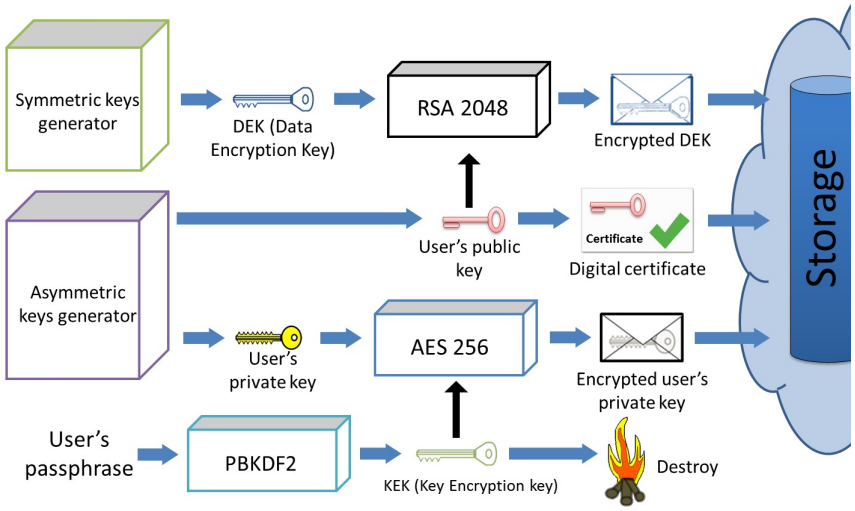


Figure 46 – Keys encryption process on CPG

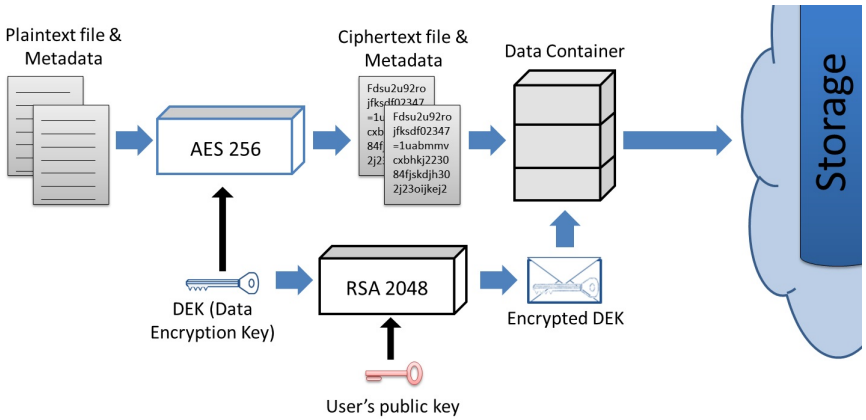


Figure 47 – Files encryption process on CPG

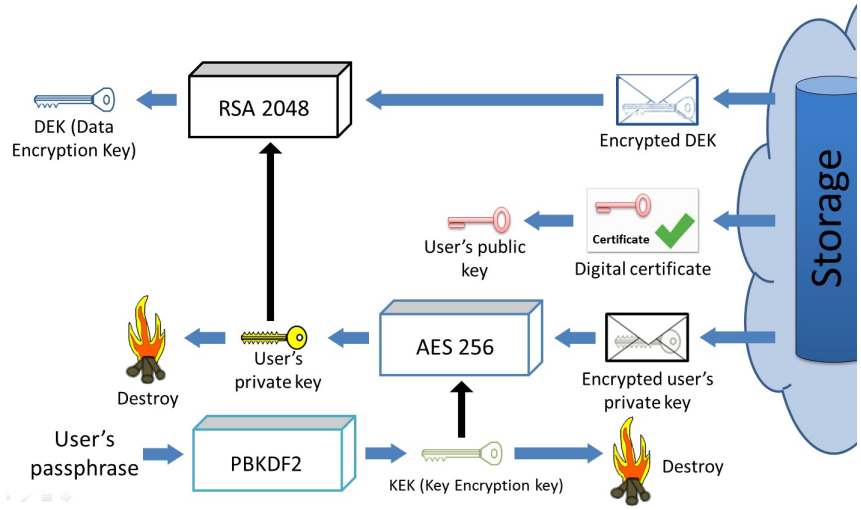


Figure 48 – Keys decryption process on CPG

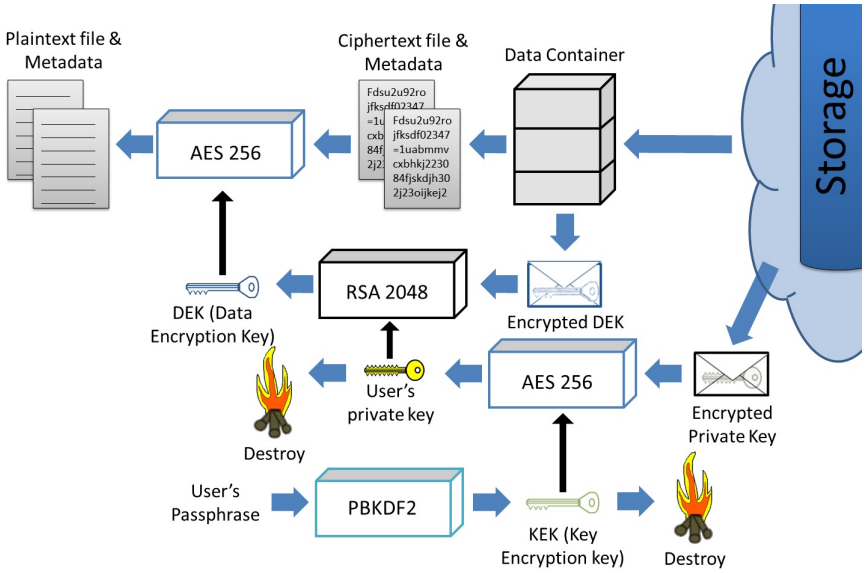


Figure 49 – Files decryption process on CPG

When sharing files, users need the recipient's certificate (public key) in order to give them access to the data. The DEK is wrapped using the owner's public key and also the recipient's public key. The encrypted DEKs are stored in the same data container. When the recipient user gets access to the encrypted file, he will decrypt the corresponding key using his private keys, and then open the encrypted file.

5.6 Comparison

CPG aims to meet all the requirements discussed in Chapter 4 in order to be a secure, reliable and easy to use application, and also to fulfil the gaps found in most solutions available nowadays. For this reason, we performed a comparison of CPG with other solutions that propose to store data securely in the cloud. Table 14 shows how well the commercial solutions and CPG satisfies the security requirements. The CSPs that meet the security requirements are represented by a green *V*, the ones that do not meet represented by a red *X* and the ones that partially meet, indicated by a blue *P*. The question mark in this table represents the absence of information, where we could not determine if the corresponding CSP meets or not the requirement.

Among all of these CSPs, Credeon and Boxcryptor are the ones similar to CPG in the way they work (an application to be integrated with an existing cloud storage infrastructure). These CPSs also use the drag and drop model in their interface to be simpler to users when storing encrypted files in clouds. As shown in the table, Boxcryptor is more close to a secure and reliable model than Credeon. However, none of them neither others solutions presented in the table fulfil all the requirements as CPG does. Most CSPs do not fulfil the requirement related to the high trusted authentication, and the ones which do it, do not address other important requirements. Also, the cryptographic keys security requirement is partially meet by most

Table 14 – Comparison of CPG and other solutions for storing data securely in the cloud

CSPs	Cryptographic keys security	High Level of data secrecy	Trust no one	Confidentiality of file attributes	Secure deduplication	Open Source	Software authenticity	Multi-factor authentication
Onedrive	X	X	X	X	X	X	X	V
Drive	X	X	X	X	X	X	X	V
Box	X	X	X	X	X	X	X	V
ownCloud	P	X	P	X	V	V	X	X
Bazil	P	V	V	X	X	V	X	X
Cypherite	P	V	V	X	V	V	X	X
BackBlaze	P	V	V	X	V	X	X	V
Credeon	P	V	V	X	V	X	X	X
ProtonMail	P	V	V	X	V	X	X	X
Carbonite	P	V	V	X	?	X	X	X
Mega	P	V	V	P	V	X	X	X
Wuala	P	V	V	P	X	X	X	X
SpiderOak	P	V	V	P	V	X	X	X
Boxcryptor	P	V	V	P	V	V	X	X
arXShare	P	V	V	V	V	V	X	X
Storj	P	V	V	V	V	V	X	X
Tresorit	V	V	V	X	V	X	X	X
CPG	V	V	V	V	V	V	V	V

CSPs, and just a few ones are concerned with the confidentiality of file attributes. The software reliability is not addressed entirely by none of them.

5.7 Limitations

CPG model considers users' computer a safe place to store files and metadata. Data is only encrypted while it is outside users' device. Malware and other threats capable of getting access to users' computer were not in the scope of this work and require different procedures in order to be mitigated. Other attacks capable of stealing users' private key while in memory are not covered, but they are mitigated since the users' cryptographic keys do not remain in memory for a long period of time but just during the period they are needed. The client device evaluation will be covered in future work, as it is an important issue that has direct impact on users' privacy. Also, a device compromised can make users feel a false sense of security, once they have tools to protect them (CPG, for example) but they are ineffective against the attacker controlling that device.

Other limitation is related to data modifications performed at the same time in two devices. If this happens, there is no way to guarantee which modification will persist after synchronization. CPG is not responsible for the synchronization in the cloud level, only locally,

and for this reason it can not control which change will occur first. There is also a problem related to the accuracy of the device clocks. It is recommended to use just one device at a time to perform any modification and let the files be synchronized before accessing them from another device.

5.8 Proof of concept

A proof of concept of CPG was developed and a first version of the application was released for tests (https://github.com/regras/cpg). The programming language chosen was Java, due to its attractive features, especially the interoperability. Besides, it has standard cryptography frameworks that simplifies the buiilding of cryptographic applications, as the JCA (Java Cryptography Architecture). This is a framework for working with cryptography that is part of the Java SE Security, a large set of APIs, tools, and implementations of security algorithms, mechanisms, and protocols. Also, JCA includes the JCE (Java Cryptography Extension) in its framework, which is responsible for providing a uniform implementation of security features in Java, supporting several applications in digital security, such as symmetric, asymmetric, stream and block ciphers, key generation, storage and retrieval, digital signatures, among others. (ORACLE, 2015a).

5.8.1 Implementation model

In order to allow users having an automatic encryption service using a simple way to store encrypted data in the cloud (through the use of the drag and drop model), CPG combines a set of technologies. The package java.nio.file is one of them, which defines interfaces and classes for the Java virtual machine, allowing access to files, file attributes, and file systems. A watch service is used to monitor the directories related to CPG (*Secret Files* and *Encrypted Files* folders) in order to catch all the changes on them, as file creation, modification or deletion (ORACLE, 2015b). The watch service is created as a thread responsible for reporting to CPG the events that may occur in the monitored folders. It gets the folder path and the event kind so CPG can know where and what to do. This information is passed to other CPG threads responsible for handling them. If any file is inserted or modified in the *Secret Files* folder, it should be encrypted and replace its old version in the *Encrypted Files* folder when possible. In case of deletion, CPG must find the corresponding encrypted version and delete it too. If the changes occur in the *Encrypted Files* folder and are related to insertion or modification, CPG must decrypt the corresponding file and replace the old version in the *Secret Files* folder for the new one. If the deletion event occurs, the plaintext version of the file must be erased too.

CPG adopts metadata files used to control the files and identify the modifications. The basic structure of these metadata, called state files, can be seen in Fig. 50. Basically, the state files are XML files and for each data handled by CPG, a new structure similar to the

one described in the figure is created, containing information related to the filename, encrypted filename, last modification date, last modification date of the encrypted file, file path, among others. Each folder has its own state file, and it is stored locally in the same folder containing CPG application files. If a directory does not have any files, its state file will be created but empty. Each device will have its own state file, created when CPG starts for the first time. If someone is using the application in a device for the first time but already used it in a different device, CPG will noticed the presence of encrypted files in the cloud when configured. Then, the files will be decrypted and all information related to them will be filled out in the state file corresponding to that folder. CPG uses the information on these metadata files to detect which data needs to be encrypted, decrypted or erased.

When users start their devices, CPG will start along with it. The first task of CPG is to look for changes in both directories (*Secret Files* and *Encrypted Files* folders) using the state files. These metadata files are necessary to compare searched data from the directories with the inputs on it using a more efficient approach since we cannot compare data in plaintext form (*Secret Files* folder) with encrypted form (*Encrypted Files* folder) without decrypting them all the time. This would require users' password to decrypt their private key, which would be used to open the DEK and then decrypt the filenames. Only then the comparison would be possible because we would know which encrypted file corresponds to the plaintext file and vice-versa, at least when both are available since there could be a new file which does not have its own plaintext or decrypt version yet. This decryption process would be necessary for all files and every time the system synchronizes data. Also, users' private key would be used more frequently and remains in memory more than necessary, and most of times being just used for verification purposes which could return nothing.

Each file and folder in both directories are verified in order to keep the system synchronized. Those files which are in the *Secret Files* folder and do not have an input in the state files, need to be encrypted. Files in the state file which are not in *Secret Files*, need to be erased from the *Encrypted Files* folder, since users must have erased them while CPG was not running. In the *Encrypted Files* folder, the logic is similar: files in *Encrypted Files* which are not in the state file need to be decrypted and the ones in the opposite situation are erased from *Secret Files*. File modifications are detected through the last modification date got from the operating system and compared to the register in the state file. When both versions of a file are available, they are verified and the last modification date of each one is compared to the entries in the metadata file. If the plaintext version has a modification data more recent than the one in the state file, the file needs to be encrypted again and its encrypted version replaced. On the other hand, if the encrypted version is more recent than the entry in the state file, the file needs to be decrypted and its plaintext version replaced.

In the synchronization process using multiple devices, as each one of them has its own state file, the logic applied is the same as before. An example: If all devices are synchro-

```
<?xml version="1.0" encoding="UTF-8"?>
<Files pathFileEncrypt="/home/vitormoia/Project/Tests/EncryptedFiles/">
    <File id="0">
        <FileName>text.txt</FileName>
        <path>/home/vitormoia/Project/Tests/SecretFiles/text.txt</path>
        <lastModification>Wed Jul 29 17:43:20 BRT 2015</lastModification>
        <lastModificationEncryptedFile>Thu Jan 07 11:31:25 BRST 2016
        </lastModificationEncryptedFile>
        <directory>false</directory>
        <FileNameEncrypted>E3SWYPALVNKK7W3YYSEGK3QSQNYYYRDYCQU3MKIFRLCRTPPKOT5Q====
        </FileNameEncrypted>
        <pathEncryptedFile>
        /home/vitormoia/Project/Tests/EncryptedFiles/E3SWYPALVNKK7W3YYSEGK3QSQNYYY
        </pathEncryptedFile>
    </File>
    <File id="1">
        <FileName>Test-folder</FileName>
        <path>/home/vitormoia/Project/Tests/SecretFiles/Test-folder</path>
        <lastModification>Fri May 22 19:39:56 BRT 2015</lastModification>
        <lastModificationEncryptedFile>Wed Dec 31 21:00:00 BRT 1969
        </lastModificationEncryptedFile>
        <directory>True</directory>
        <FileNameEncrypted>ZJCMR7AOTDDN2DHMJFDHUYB5MNEER7WHIDJKXV7AQ5HEMLX7MCGA====
        </FileNameEncrypted>
        <pathEncryptedFile>
        /home/vitormoia/Project/Tests/EncryptedFiles/ZJCMR7AOTDDN2DHMJFDHUYB5MNEER
        </pathEncryptedFile>
    </File>
    <File id="2">
        <FileName>Paper.pdf</FileName>
        <path>/home/vitormoia/Project/Tests/SecretFiles/Paper.pdf</path>
        <lastModification>Fri May 22 19:39:56 BRT 2015</lastModification>
        <lastModificationEncryptedFile>Wed Dec 31 21:00:00 BRT 1969
        </lastModificationEncryptedFile>
        <directory>false</directory>
        <FileNameEncrypted>24OEY2Y5GPQWACVK6UI64AKUXLFCDJA7TF3WAPSAWWDEGEMLRP5Q====
        </FileNameEncrypted>
        <pathEncryptedFile>
        /home/vitormoia/Project/Tests/EncryptedFiles/24OEY2Y5GPQWACVK6UI64AKUXLFCD
        </pathEncryptedFile>
    </File>
</Files>
```

Figure 50 – CPG Metadata structure

nized and a file is erased from device 1, the input in the state file and the encrypted version of it in the *Encrypted Files* folder will be erased from this device. When other devices are used, the CSP client will synchronize the *Encrypted Files* folder in these devices with the cloud servers. The file erased from device 1 will have its encrypted version erased from the other devices, and when CPG starts the synchronization process, will notice an input the in state file and the plaintext file, but not the encrypted version of it. Following the logic above, the input and plaintext file will be erased, maintaining all devices with the same data.

In order to keep users data synchronized, CPG adopts a verification scheme performed in two ways: partial and full. The first one is performed through the operating system

events using the watch service described above. CPG will stay sleeping and waiting for any event occurrence to treat it. On the other hand, the full verification is performed every time CPG starts to execute and from time to time. In this scheme, CPG looks for changes in both directories (*Secret Files* and *Encrypted Files* folders) using the state files. Each file and folder in both directories is verified in order to keep the system synchronized.

The full verification is necessary mostly in systems which stay disconnected from the network or turnned off for a long period of time. If an event was lost for some reason and CPG could not handle it, the corresponding file would not be encrypted, decrypted or erased, and CPG directories would not be synchronized until the system be restarted. To avoid this situation, after a defined period of time, a complete verification takes place and every file and folder is verified again, synchronizing the folders.

5.8.2 CPG class diagram

Fig. 51 presents a simplified class diagram of CPG first version. In this figure, we illustrated only the main classes essential to the application operation. A complete class diagram can be found along with CPG source code (https://github.com/regras/cpg).

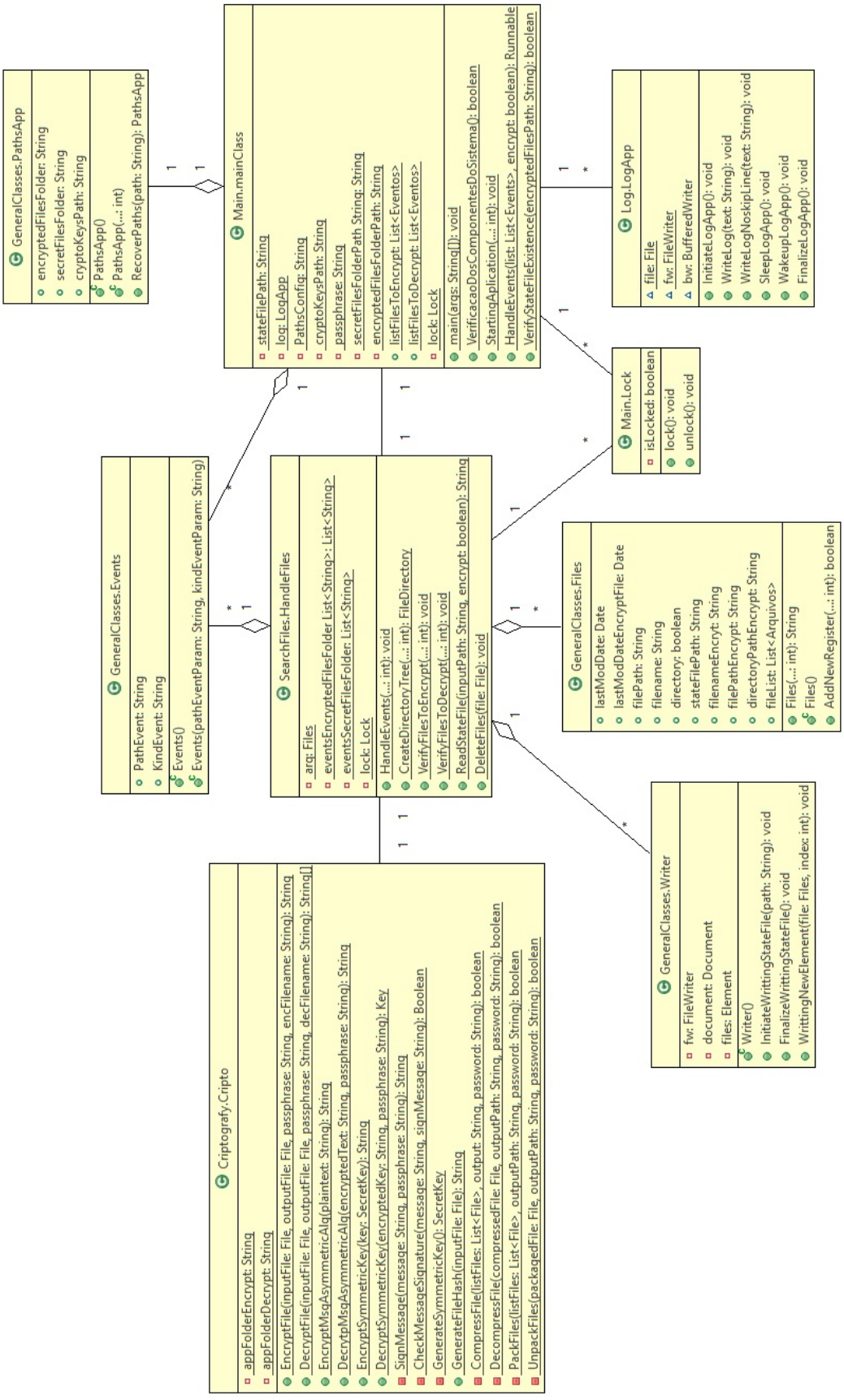


Figure 51 – CPG simplified class diagram

5.9 Future steps in CPG development

CPG tried to fulfil all the requirements discussed in Chapter 4, but in its first version improvements are still necessary in order increase the level of security and usability, as discussed in this section.

The first step is to create a new version of CPG that attends all the requirements entirely. The support for PKI is one of the features that still need to be implemented to allow data sharing and to fulfil the cryptographic keys security requirement.

A future improvement could be related to the high trusted authentication. CPG could replace the second factor authentication adopted in the current version (mobile phone using *Google Authentication App*) by a more secure model, similar to the one offered by Nitrokey (NITROKEY, 2015). They use an external device built specifically to protect users' cryptographic keys. In Nitrokey's solution, users' secret keys are stored in a tamper-resistant and PIN-protected device, which is secure against viruses and Trojan Horses, and even by loss or theft since users have a PIN to protect the keys stored on it. CPG could store users' private key in a device similar to this one in order to increase its level of security in the authentication process. However, this extra level of security comes with a price. This would require an extra hardware and users would have to carry it whenever they want to access their data. Also, the costs associated to this new device should be considered.

There are other possible changes in the CPG's authentication based on 2FA. In the current model, the second factor is required every time users perform the authentication. However, the 2FA could be used in different situations. In the first one the 2FA is only required when users need to access their most valuable files. They would choose some files to add an extra protection layer, and every time they would access these files the second factor would be required. Other scenario is using the continuous authentication method, where users must provide their credentials from time to time to prove they still own all of them. The use of more sophisticated forms of 2FA will be left for future work.

In order to improve CPG's performance, we can replace the RSA cryptography by ECC. The latter asymmetric algorithm uses keys with the same level of security but smaller in size. This would increase the performance of cryptographic operations. However, the implementation of ECC is not very simple since choosing trusted parameters is difficulty and requires a better understanding of all the implications.

5.10 Conclusions

In this chapter we presented CPG, an application proposed for mitigating some risks and threats on cloud data storage and also to fulfil a set of requirements indispensable in a secure, reliable and easy to use cloud application. Along this chapter, we described the

objectives of CPG, how it works and performs all the cryptographic operations, the requirements and how they were met by CPG, and also a comparison with other CSPs and applications with a similar purpose. Finally, we discussed CPG's limitations and also the next steps in its development.

6 Conclusions

Privacy in cloud computing is an important subject and ways to ensure it are needed towards a wider adoption of a more private computing model, mostly when it comes to the protection of sensitive data. In this work we presented some problems related to the security and privacy of users who store their data in the cloud. Moreover, we presented some concerns about users' privacy on data storage services, related to the data name, content, access and possession confidentiality, along with some techniques used to mitigate them. We also presented a review on academic solutions aiming to minimize some threats and concerns. We studied these solutions and identified the problems they tried to mitigate, the proposed techniques used to solve these problems and how they did it. However, these solutions are limited on solving only a set of problems, which could not be enough for some users. Also, some of them are still not practical for use and in most solutions the usability is not even considered.

Another subject approached in this work was the proposal of a set of requirements we consider essential in the design of a secure, reliable and easy to use cloud system for data storage. The focus of these requirements are on data content and name confidentiality due to the greater concern demonstrated by users about these topics. We also presented several related works from commercial solutions and other ones which use the cryptography to address confidentiality in general and could be integrated to the cloud storage environment in order to protect users data and privacy.

The current commercial solutions were analysed and compared, concluding that most of them do not offer the necessary level of security to guarantee users' privacy, since none of the available cloud service providers (CSP) meet all the requirements proposed in this work. Despite all security features adopted by them, some problems still remain and there is a lot of space for improvements with respect to the user's privacy point of view. With respect to the general solutions, we concluded that, even offering a good level of security, they do not meet the requirements and were not built for this purpose, being harder to use in this environment for most users. Extra procedures would be necessary to integrate them to cloud data storage, which could be a burden for some users.

In this work, we also made an analysis of the relative costs and benefits of several techniques to preserve users' privacy and all their possible combinations. A methodology to calculate these costs in order to find the best solutions was also proposed. Besides, this methodology revealed the solutions that are worth to be used and the ones which are ineffective and even prohibitive due to their elevated costs and low benefits.

In order to consolidate and prove our claims, we proposed (and built a proof of concept) a cryptography application, called Cloud Privacy Guard (CPG), aiming to mitigate some

of the problems discussed and also to fulfil the security requirements discussed. We showed how the application works, including the details of the cryptographic protocols adopted, and made a comparison of it with commercial solutions. CPG proposal is concerned about being transparent to users, through initiatives like signing open source code and executable files. It uses certificates to mitigate some security problems and to enable data sharing. Furthermore, the proposal is compromised to leverage the security and improve privacy for all kinds of users, being them experts or not in the filed of security.

6.1 Contributions

The main contributions of this study are summarized below.

- A methodology to calculate and analyze the relative costs and benefits of some techniques used to provide privacy in cloud computing and their combinations.
- Identification and grouping of a set of requirements for cloud data storage systems in order to be secure, reliable and easy to use.
- A classification of how cryptography can be applied in the cloud data storage (levels of secrecy).
- The proposal of an application model (called CPG) with the characteristics desired in a cloud data storage application.

All these contributions are also documented in the papers published in national and international scientific conferences, which are presented in appendix D.

6.2 Future work

As a future work we intend to study the concerns related to data access and possession confidentiality in more detail, with the purpose of deriving requirements for a more private cloud storage application. We also plan to study the threats related to inability to access data, data lock-in, and data fragmentation.

A more precise estimate of the costs and benefits from the techniques used to improve users' privacy in cloud computing is planned, which could help users making a better choice. It is also planned to look for more adequate ways to determine the privacy level according to users' vision, since the online survey conducted was not efficient. We realized that most users had difficulties in understanding the problems and the techniques from the description provided in the survey. We think that more realistic results can be obtained by exposing the participants to a better explanation where all details (threat model and the benefits obtained from each technique) are given.

We intend to study deeper the 4th level of secrecy, where Homomorphic Encryption is used, to better understand its potential. The appropriate case scenario needs to be understood as well as the advantages of this technique to users. Another requirement that will receive further attention is the security of cryptographic keys, where the adopted protocol could be improved to simplify the key life cycle management. Other ways to implement the high trust authentication requirement will also be studied, along with the use of different authentication factors. We also plan to study and evaluate new CSPs, according to the requirements presented.

Another future work is related to CPG. We want to work more in this application proposal in order to improve its efficiency, user-friendliness and security.

Bibliography

ABU-LIBDEH, H.; PRINCEHOUSE, L.; WEATHERSPOON, H. Racs: A case for cloud storage diversity. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*. New York, NY, USA: ACM, 2010. (SoCC '10), p. 229–240. ISBN 978-1-4503-0036-0.

ALLIANCE, C. The notorious nine: cloud computing top threats in 2013. *Cloud Security Alliance*, 2013.

ALLIANCE, O. *About OpenPGP*. 2001. <http://www.openpgp.org/about_openpgp/>. Accessed 2015 Dez 10.

ASHKTORAB, V.; TAGHIZADEH, S. R. Security threats and countermeasures in cloud computing. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, v. 1, n. 2, p. 234–245, 2012.

BABAOGLU, O.; MARZOLLA, M. The people's cloud. *Spectrum, IEEE*, IEEE, v. 51, n. 10, p. 50–55, 2014.

BABAOGLU, O.; MARZOLLA, M.; TAMBURINI, M. Design and implementation of a p2p cloud system. In: ACM. *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. [S.l.], 2012. p. 412–417.

BARKER W. BARKER, W. B. W. P. M. S. E. Recommendation for key management-part 1: General (revised). In: CITESEER. *NIST special publication*. [S.l.], 2006.

BAZIL. *Bazillion bytes*. 2015. <https://bazil.org/>. Accessed 2015 Dez 10.

BERNSTEIN, D. J.; LANGE, T.; SCHWABE, P. The security impact of a new cryptographic library. In: *Progress in Cryptology–LATINCRYPT 2012*. [S.l.]: Springer, 2012. p. 159–176.

BESSANI, A.; CORREIA, M.; QUARESMA, B.; ANDRÉ, F.; SOUSA, P. Depsky: Dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, ACM, v. 9, n. 4, p. 12, 2013.

BIRRELL, E.; SCHNEIDER, F. B. Federated identity management systems: A privacy-based characterization. 2012.

BOX. *Secure, effortless collaboration from any device*. 2015. <https://www.box.com/home/>. Accessed 2015 Dez 10.

BOXCRYPTOR. *Technical Overview*. 2015. <https://www.boxcryptor.com/en/technical-overview>. Accessed 2015 Dez 10.

BROZ, M.; MATYAS, V. The truecrypt on-disk format–an independent view. *Security & Privacy, IEEE*, IEEE, v. 12, n. 3, p. 74–77, 2014.

CALLAS, J.; DONNERHACKE, L.; FINNEY, H.; SHAW, D.; THAYER, R. RFC 4880-openpgp message format. *Informe técnico, Internet Engineering Task Force (IETF)*, 2007.

CARBONITE. *Encryption*. 2015. <http://support.carbonite.com/articles/Server-Windows-Encryption#auto_encrypt>. Accessed 2015 Dez 10.

CHADWICK, D. W. Federated identity management. In: *Foundations of security analysis and design V*. [S.l.]: Springer, 2009. p. 96–120.

CHEN, P.-C.; FREG, C.-P.; HOU, T.-W.; TENG, W. G. Implementing raid-3 on cloud storage for emr system. In: *Computer Symposium (ICS), 2010 International*. [S.l.: s.n.], 2010. p. 850–853.

CIPHERSHED. *CipherShed Technical Wiki*. 2015. <https://wiki.ciphershed.org/>. Accessed 2015 Dez 10.

CREDEON. *Cloud Data Protection*. 2015. <http://psg.hitachi-solutions.com/credeon/cloud-data-protection-overview>. Accessed 2015 Dez 10.

CYPHERTITE. *Cryptography*. 2015. <https://www.cyphertite.com/papers/WP_Crypto.pdf>. Accessed 2015 Jul 10.

CZESKIS, A.; HILAIRE, D. J. S.; KOSCHER, K.; GRIBBLE, S. D.; KOHNO, T.; SCHNEIER, B. Defeating encrypted and deniable file systems: Truecrypt v5. 1a and the case of the tattling os and applications. In: *HotSec*. [S.l.: s.n.], 2008.

DMITRIENKO, A.; LIEBCHEN, C.; ROSSOW, C.; SADEGHI, A.-R. Security analysis of mobile two-factor authentication schemes. *Intel® Technology Journal*, v. 18, n. 4, 2014.

DRIVE. *Visao geral das conexoes SSL*. 2015. <https://support.google.com/a/answer/100181?hl=pt-BR>. Accessed 2015 Dez 10.

DUFFIELD, N.; GREENBERG, A.; GOYAL, P.; MISHRA, P.; RAMAKRISHNAN, K.; MERWE, J. Van der. *Virtual private network*. [S.l.]: Google Patents, 2005. US Patent 6,912,232.

ERMAKOVA, T.; FABIAN, B. Secret sharing for health data in multi-provider clouds. In: *Business Informatics (CBI), 2013 IEEE 15th Conference on*. [S.l.: s.n.], 2013. p. 93–100.

ESHGHI, K.; TANG, H. K. A framework for analyzing and improving content-based chunking algorithms. *Hewlett-Packard Labs Technical Report TR*, v. 30, p. 2005, 2005.

FILEVAULT. Usar o filevault para criptografar o disco de inicialização no seu mac. 2015. Accessed 2015 Dez 14.

FU, Y.; SUN, B. A scheme of data confidentiality and fault-tolerance in cloud storage. In: *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. [S.l.: s.n.], 2012. v. 01, p. 228–233.

GASTI, P.; ATENIESE, G.; BLANTON, M. Deniable cloud storage: Sharing files via public-key deniability. In: *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*. New York, NY, USA: ACM, 2010. (WPES '10), p. 31–42. ISBN 978-1-4503-0096-4.

GMBH, A. S. *arXshare: End-to-end encrypted file storage*. 2015. <http://www.arxshare.com/faq/>. Accessed 2015 Dez 10.

GROLIMUND, D.; MEISSER, L.; SCHMID, S.; WATTENHOFER, R. Cryptree: A folder tree structure for cryptographic file systems. In: IEEE. *Reliable Distributed Systems, 2006. SRDS'06. 25th IEEE Symposium on*. [S.l.], 2006. p. 189–198.

HARNIK, B. P. D.; SHULMAN-PELEG, A. Side channels in cloud services: Deduplication in cloud storage. *Security & Privacy, IEEE*, IEEE, v. 8, n. 6, p. 40–47, 2010.

HUBBARD, D.; SUTTON, M. Top threats to cloud computing v1. 0. *Cloud Security Alliance*, 2010.

IETF. *HTTP Over TLS*. 2000. <https://tools.ietf.org/html/rfc2818>. Accessed 2016 Jan 10.

IETF. *Password-Based Cryptography Specification: Version 2.0*. 2000. <https://www.ietf.org/rfc/rfc2898.txt>. Accessed 2015 Dez 10.

IETF. *The Transport Layer Security (TLS) Protocol: Version 1.2*. 2008. <https://tools.ietf.org/html/rfc5246>. Accessed 2015 Dez 10.

IETF. *TOTP: Time-Based One-Time Password Algorithm*. 2011. <https://tools.ietf.org/html/rfc6238>. Accessed 2015 Dez 10.

JAATUN, M.; NYRE, A.; ALAPNES, S.; ZHAO, G. A farewell to trust: An approach to confidentiality control in the cloud. In: *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*. [S.l.: s.n.], 2011. p. 1–5.

JAATUN, M. G.; ZHAO, G.; ALAPNES, S. A cryptographic protocol for communication in a redundant array of independent net-storages. In: *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*. Washington, DC, USA: IEEE Computer Society, 2011. (CLOUDCOM '11), p. 172–179. ISBN 978-0-7695-4622-3.

JACOBSON, V.; SMETTERS, D. K.; THORNTON, J. D.; PLASS, M. F.; BRIGGS, N. H.; BRAYNARD, R. L. Networking named content. In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. New York, NY, USA: ACM, 2009. (CoNEXT '09), p. 1–12. ISBN 978-1-60558-636-6.

JUNESTAM, A.; GUIGO, N. *Open Crypto Audit Project: TrueCrypt*. 2014. <https://truecrypt.ch/wp-content/uploads/2015/04/TrueCrypt_Phase_II_NCC_OCAP_final.pdf>. Accessed 2015 Dez 10.

KALLAHALLA, M.; RIEDEL, E.; SWAMINATHAN, R.; WANG, Q.; FU, K. Plutus: Scalable secure file sharing on untrusted storage. In: *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2003. (FAST '03), p. 29–42.

KALPANA, P.; SINGARAJU, S. Data security in cloud computing using RSA algorithm. *IJRCCT*, v. 1, n. 4, p. 143–146, Sep 2012.

KAMARA, S.; LAUTER, K. Cryptographic cloud storage. In: *Proceedings of the 14th International Conference on Financial Cryptograpy and Data Security*. Berlin, Heidelberg: Springer-Verlag, 2010, (FC'10). p. 136–149. ISBN 3-642-14991-X, 978-3-642-14991-7.

KHALIL, I. M.; KHREISHAH, A.; AZEEM, M. Cloud computing security: A survey. *Computers*, Multidisciplinary Digital Publishing Institute, v. 3, n. 1, p. 1–35, 2014.

KOŚCIELNY, C.; KURKOWSKI, M.; SREBRNY, M. Pgp systems and truecrypt. In: *Modern Cryptography Primer*. [S.l.]: Springer, 2013. p. 147–173.

KUMAR, A.; LEE, B. G.; LEE, H.; KUMARI, A. Secure storage and access of data in cloud computing. In: *ICT Convergence (ICTC), 2012 International Conference on*. [S.l.: s.n.], 2012. p. 336–339.

KUMAR, M.; KUMAR, M. A secured cloud storage technique to improve security in cloud infrastructure. In: *Recent Trends in Information Technology (ICRTIT), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 97–102.

KUROSE, J. F.; ROSS, K. W. *Computer networking: a top-down approach*. Boston, MA, USA: Addison-Wesley, 2007.

LAM, I.; SZEBENI, S.; BUTTYAN, L. Invitation-oriented tgdh: Key management for dynamic groups in an asynchronous communication model. In: *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. [S.l.: s.n.], 2012. p. 269–276. ISSN 1530-2016.

LEE I. ONG, H. L. H. L. S. Two factor authentication for cloud computing. *Journal of information and communication convergence engineering*, v. 8, n. 4, p. 427–432, 2010.

LI, J.; SONG, D.; CHEN, S.; LU, X. A simple fully homomorphic encryption scheme available in cloud computing. In: *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. [S.l.: s.n.], 2012. v. 01, p. 214–217.

LUKS. Cryptsetup and luks - open-source disk encryption. 2015. Accessed 2015 Dez 14.

MATHER, T.; KUMARASWAMY, S.; LATIF, S. *Cloud security and privacy: an enterprise perspective on risks and compliance*. [S.l.]: " O'Reilly Media, Inc.", 2009.

MEGA. *Developers - Documentation*. 2015. <https://mega.nz/#doc>. Accessed 2015 Dez 10.

MEISSER, L. *Wuala Blog. Wuala's Encryption For Dummies*. 2011. <https://www.wuala.com/blog/2011/04/wualas-encryption-for-dummies.html>. Accessed 2015 Jul 10.

MICROSOFT. Visão geral da criptografia de unidade de disco bitlocker. 2015. Accessed 2015 Dez 14.

MURDOCH, S.; DANEZIS, G. Low-cost traffic analysis of tor. In: *Security and Privacy, 2005 IEEE Symposium on*. [S.l.: s.n.], 2005. p. 183–195. ISSN 1081-6011.

NITROKEY. *Introduction*. 2015. <https://www.nitrokey.com/>. Accessed 2015 Dez 10.

NUFIRE, T. *BackBlaze: How to make strong encryption easy to use*. 2008. <https://www.backblaze.com/blog/how-to-make-strong-encryption-easy-to-use/>. Accessed 2015 Dez 10.

ONEDRIVE. *A sua vida em um único lugar*. 2015. <https://onedrive.live.com/about/pt-br/>. Accessed 2015 Dez 10.

ORACLE. *Java SE Security*. 2015. <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html>. Accessed 2015 Dez 10.

ORACLE. *Package java.nio.file*. 2015. <https://docs.oracle.com/javase/7/docs/api/java/nio/file/package-summary.html>. Accessed 2015 Dez 10.

OWNCLOUD. *Deduplication on Owncloud: Frequently Asked Questions*. 2013. <https://owncloud.org/faq/deduplication>. Accessed 2015 Jul 10.

PADMAJA, N.; KODURU, P. Providing data security in cloud computing using public key cryptography. *IJESR*, v. 4, n. 01, 2013.

PATTERSON, D. A.; GIBSON, G.; KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). *SIGMOD Rec.*, ACM, New York, NY, USA, v. 17, n. 3, p. 109–116, jun. 1988. ISSN 0163-5808.

PFITZMANN, A.; HANSEN, M. *A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management*. 2010.

PHUONG, T.; OMOTE, K.; LUYEN, N.; THUC, N. Improvement of multi-user searchable encrypted data scheme. In: *Internet Technology And Secured Transactions, 2012 International Conference for*. [S.l.: s.n.], 2012. p. 396–401.

PROTONMAIL. *How are ProtonMail keys distributed?* 2014. <http://security.stackexchange.com/questions/58541/how-are-protonmail-keys-distributed>. Accessed 2015 Dez 10.

PTACEK, T. *Javascript Cryptography Considered Harmful*. 2011. <https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2011/august/javascript-cryptography-considered-harmful/>. Accessed 2015 Dez 10.

RNP. *ICPEdu*. 2015. <http://www.rnp.br/servicos/servicos-avancados/icpedu>. Accessed 2015 Dez 10.

SCHIESSLE, B. *Owncloud: Introduction to the new ownCloud Encryption App*. 2013. <http://blog.schiessle.org/2013/05/28/introduction-to-the-new-owncloud-encryption-app/>. Accessed 2015 Dez 10.

SCHNJAKIN, M.; ALNEMR, R.; MEINEL, C. A security and high-availability layer for cloud storage. In: *Proceedings of the 2010 International Conference on Web Information Systems Engineering*. Berlin, Heidelberg: Springer-Verlag, 2011. (WISS'10), p. 449–462. ISBN 978-3-642-24395-0.

SCHNJAKIN, M.; KORSCH, D.; SCHOENBERG, M.; MEINEL, C. Implementation of a secure and reliable storage above the untrusted clouds. In: *Computer Science Education (ICCSE), 2013 8th International Conference on*. [S.l.: s.n.], 2013. p. 347–353.

SCHNJAKIN, M.; METZKE, T.; MEINEL, C. Applying erasure codes for fault tolerance in cloud-raid. In: *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*. [S.l.: s.n.], 2013. p. 66–75.

SHAMIR, A. How to share a secret. *Commun. ACM*, ACM, New York, NY, USA, v. 22, n. 11, p. 612–613, nov. 1979. ISSN 0001-0782.

SHATILOV, K.; BOIKO, V.; KRENDELEV, S.; ANISUTINA, D.; SUMANEEV, A. Solution for secure private data storage in a cloud. In: *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. [S.l.: s.n.], 2014. p. 885–889.

SILVA, C.; RODRIGUES, L. A fault-tolerant secure corba store using fragmentation-redundancy-scattering. In: CITESEER. *ECOOP Workshops*. [S.l.], 1998. p. 287.

SPIDEROAK. *Engineering. The Details Behind What We Do*. 2015. <https://spideroak.com/features/private-by-design#engineering-matters-10-ribbon>. Accessed 2015 Dez 18.

STALLINGS, W. *Criptografia e segurança de redes 4ª ed*. [S.l.]: São Paulo: Pearson Prentice Hall, 2008.

STANDARDIZATION., I. O. for. *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*. [S.l.: s.n.], 1998.

STERNSTEIN, A. *WHITE HOUSE TELLS AGENCIES TO TIGHTEN UP CYBER DEFENSES 'IMMEDIATELY'*. 2015. <http://www.nextgov.com/cybersecurity/2015/06/white-house-tells-agencies-tighten-online-security-immediately/115216/>. Accessed 2015 Dez 10.

STORJ. Descentralized cloud storage. 2016. Accessed 2016 Jan 15.

SUBASHINI, S.; KAVITHA, V. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, Elsevier, v. 34, n. 1, p. 1–11, 2011.

TCNEXT. *TrueCrypt will not die*. 2015. <https://truecrypt.ch/>. Accessed 2015 Dez 10.

TRESORIT. *How is my password managed in Tresorit?* 2015. <https://tresorit.zendesk.com/entries/23577091-How-is-my-password-managed-in-Tresorit>. Accessed 2015 Dez 10.

TRESORIT. *White Paper*. 2015. <https://tresorit.com/files/tresoritwhitepaper.pdf>. Accessed 2015 Dez 10.

TRUECRYPT. *Migrating from TrueCrypt to BitLocker*. 2014. <http://truecrypt.sourceforge.net/>. Accessed 2015 Dez 10.

VERACRYPT. *User's guide*. 2015. <https://veracrypt.codeplex.com/documentation>. Accessed 2015 Dez 10.

WANG, C.; QIN, Z.-g.; PENG, J.; WANG, J. A novel encryption scheme for data deduplication system. In: IEEE. *Communications, Circuits and Systems (ICCCAS), 2010 International Conference on*. [S.l.], 2010. p. 265–269.

WILKINSON, S.; BOSHEVSKI, T.; BRANDOFF, J.; BUTERIN, V. Storj a peer-to-peer cloud storage network. Citeseer, 2014.

WILKINSON, S.; LOWRY, J. Metadisk a blockchain-based decentralized file storage application. Citeseer, 2014.

XU, H.-M.; SHI, Y.-J.; LIU, Y.-L.; GAO, F.-B.; WAN, T. Integration of cloud computing and p2p: A future storage infrastructure. In: IEEE. *Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2012 International Conference on*. [S.l.], 2012. p. 1489–1492.

XU, L.; WU, X.; ZHANG, X. Cl-pre: a certificateless proxy re-encryption scheme for secure data sharing with public cloud. In: ACM. *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. [S.l.], 2012. p. 87–88.

YIN, X. C.; LIU, Z. G.; LEE, H. J. An efficient and secured data storage scheme in cloud computing using ecc-based pki. In: *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. [S.l.: s.n.], 2014. p. 523–527.

ZHOU, M.; ZHANG, R.; XIE, W.; QIAN, W.; ZHOU, A. Security and privacy in cloud computing: A survey. In: *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*. [S.l.: s.n.], 2010. p. 105–112.

ZISSIS, D.; LEKKAS, D. Addressing cloud computing security issues. *Future Generation computer systems*, Elsevier, v. 28, n. 3, p. 583–592, 2012.

Appendix

APPENDIX A – Survey: Privacy

In this appendix, we show the details about the survey mentioned in Chapter 3.

A.1 Introduction

We conducted a survey in order to evaluated the privacy level perceived by users when using some techniques to preserve their privacy on cloud data storage environment. In this section, we will show the details about the survey as well as the results obtained.

The survey was conducted using a software-based and open-source online platform, called LimeSurvey. This service is offered by the University of Campinas, and was used to collect users' opinion about the techniques. A brief description was made, as well as how they can be used to preserve users' privacy. We expected that each survey participant expressed his/her perception of privacy using the techniques, considering preferences, priorities and experiences. In each case, the participants had to choose values in a scale from 0 to 10, where 0 means no privacy at all and 10 means that the highest level of privacy is obtained. They should consider the entire process of storing and recovering files in public clouds and all possible issues they could see related to the exposure of sensitive information, especially to the cloud administrators.

We had a total of 76 participants. The target audience was people with any expertise in computer, in any related field. Among the participants, we had professors, master, doctor, undergraduate students, and some professionals of the field. In the next sections, we will explain how we elaborated the survey and the results obtained.

A.2 Description

When users started the survey, a message was showed explaining its purpose and what was expected from them. The introduction message is shown below.

This survey aims to estimate the level of privacy an average cloud user perceives for different techniques used to preserve his/her privacy in cloud data storage services. Each technique has a different purpose and provides a different level of privacy.

We made a brief description for each one of them about their characteristics, as well as how they can be used to preserve user privacy. We expect that the survey participants express their perception of privacy for the techniques, considering preferences, priorities and experiences.

The results of this survey will be used for calculating a relative level of privacy got from the enlisted techniques according to users' point of view. This survey is part of a research about

the security and privacy on public cloud data storage held at the Department of Computer Engineering and Industrial Automation (DCA), School of Electrical and Computer Engineering (FEEC), University of Campinas (Unicamp).

The results will be made public in the web page shown below.

http://www.dca.fee.unicamp.br/~vhgmoia/cloud_security/results.html

Further details about the survey, can be obtained from:

Vitor H. G. Moia (vhgmoia@dca.fee.unicamp.br) or Marco A. A. Henriques (marco@dca.fee.unicamp.br)

Further details about the techniques, can be obtained in:
http://www.dca.fee.unicamp.br/~vhgmoia/cloud_security/dissertation.pdf

A.3 Questions

Following the introduction, the survey participants were asked to answer some questions. We divided these questions in two groups which were presented in different pages. The first one was about personal information, with three optional questions on the participant name, institution, and address; there was also one mandatory question on how the participant classified its own expertise on information security, with the following possible answers: Low, medium and high.

The second group was related to the privacy level obtained from the techniques in the users' point of view. There were six mandatory questions and for each one, the participants had to choose grades from 0 to 10 in order to classify the techniques. A brief explanation was made for each technique highlighting its definition and goal. Users also had space for optional comments in each question. The questions and explanation about the techniques are presented below.

- Question: *From your point of view, choose a number from 0 to 10 that better describes the level of privacy you would perceive when using one of the techniques described below to protect the data you store in a public cloud. The number 0 means no privacy at all and 10 means that the highest level of privacy is obtained. Consider the entire process of storing and recovering files in public clouds and all possible issues they can see related to the exposure of sensitive information, especially to the cloud administrators. Obs.: Comments are optional (English or Portuguese).*

- Techniques description:

1. **Data fragmentation**

   Goal: *Offer redundancy in data storage and make it difficult unauthorized access to the whole file.*

   Description: *In this technique, data is broken in many small fragments which are spread over several CSPs (Cloud Service Providers). Normally, the main reason to use data fragmentation is to provide fault tolerance and/or improve performance. But here, data is spread over different clouds to avoid attackers to get the whole data in case of security breaches in some of them.*

2. **Data encryption**

   Goal: *Keep data safe from unauthorized access.*

   Description: *Data encryption is a technique used to keep information confidential and accessible only by those who know a secret (a key), used in the codification process. The secrecy is achieved by a codification based on the key, which will generate an unintelligible version of the original data. In this context, we only consider the encryption of file content, as implemented in several public clouds. Data attributes (as file's name, type, size, creation and modification date etc.) will not be encrypted, but stored in plaintext (legible).*

3. **Metadata encryption**

   Goal: Keep the file attributes unavailable to third parties.

   Description: Data encryption is a technique used to keep information confidential and accessible only by those who know a secret (a key), used in the codification process. The secrecy is achieved by a codification based on the key, which will generate an unintelligible version of the original data. In this context, only data attributes, as filename, size, last modification and creation dates etc., will be encrypted. The data itself will remain in plaintext (legible).

4. **VPN-Proxy**

   Goal: *Keep user's location (IP) secret with the help of a third party.*

   Description: *A trusted VPN (Virtual Private Network) service combined with a proxy creates a secure (encrypted) communication channel between user's device and a VPN-Proxy server. The main function of this server is to receive, decrypt and forward the users' packets to the cloud, but using as the source address its own IP in place of the original one, providing IP (localization) anonymity. Although all communication between user's device and the VPN-Proxy servers goes through a secure (encrypted) communication channel, those between VPN-Proxy and cloud are not necessarily encrypted.*

5. **TOR (The Onion Router)**

   Goal: *Keep users' location secret with the help of TOR servers.*

   Description: *Technology to protect users against surveillance, masking their location (IP address) and encrypting all traffic while surfing on the Internet. All users'*

messages goes through a series of random TOR nodes spread over the network in secure communication channels. In addition, several layers of encryption are put over the messages in order to protect them and keep their source confidential. It is considered that all configurations needed to access cloud servers from TOR network are in place and working properly.

6. **Federated Identity**
   Goal: *Keep user's real identity secret from the CSPs.*
   Description: *Delegates the authentication process to a trusted third party, the Identity Provider (IdP). When users want to access an online service which requires authentication, instead of creating a new account in that service, they can choose to authenticate using an identity provider (Facebook or Google, for example). This technology could be applied to the cloud environment and used to hide user's real identity. When accessing cloud services, users authenticate with a trusted IdP, which creates tickets (pseudonyms) for them. Users are therefore known to that CSP only by these pseudonyms and the access to their files in the cloud is done without the CSP linking the data stored to their owners' true names.*

A.4 Results

We had a total of 76 participants in the survey. Table A.1 shows the values chosen by each participant according to each technique. Fig. A.1 to A.6 presents the results for each technique, describing the amount of the votes each level of privacy received. Finally, Table A.2 illustrates statistical data derived from the results.

Table A.1 – Values attributed by the participants for each technique

User	Data Fragmentation	Data Encryption	Metadata Encryption	VPN-Proxy	TOR	Federated Identity
1	6	9	3	3	4	5
2	7	8	0	0	6	5
3	8	9	3	4	4	3
4	3	8	6	3	9	6
5	7	9	4	6	9	5
6	5	6	6	4	8	8
7	9	10	8	7	6	0
8	7	8	8	8	9	9
9	6	8	5	6	7	6
10	7	7	5	4	6	5

Continued on next page

Table A.1 – Values attributed by the participants for each technique (continued)

User	Data Fragmentation	Data Encryption	Metadata Encryption	VPN-Proxy	TOR	Federated Identity
11	7	8	3	5	9	5
12	6	9	7	7	6	6
13	6	10	3	3	6	4
14	2	4	3	9	4	4
15	6	7	6	7	7	7
16	8	6	4	3	8	6
17	5	7	2	6	8	7
18	3	7	5	4	9	6
19	9	8	9	9	9	2
20	7	6	6	7	8	7
21	5	8	9	7	6	4
22	8	9	8	8	9	8
23	5	10	10	0	0	5
24	7	9	8	9	6	8
25	7	8	5	9	5	9
26	3	0	0	6	8	9
27	8	8	6	5	7	7
28	7	8	5	7	8	5
29	4	10	1	5	10	0
30	1	8	2	7	9	5
31	6	9	6	5	8	7
32	3	8	5	9	10	8
33	3	5	6	7	6	5
34	8	10	2	5	5	7
35	6	6	6	6	6	6
36	3	9	1	7	8	2
37	8	7	6	8	10	9
38	5	10	10	9	7	8
39	3	6	1	0	0	0
40	7	7	4	6	7	4
41	7	8	9	6	7	7
42	9	8	9	8	8	5
43	3	5	5	5	7	8
44	5	7	8	5	9	10

Continued on next page

Table A.1 – Values attributed by the participants for each technique (continued)

User	Data Fragmentation	Data Encryption	Metadata Encryption	VPN-Proxy	TOR	Federated Identity
45	0	9	9	6	6	2
46	7	8	4	8	9	9
47	4	8	6	5	6	7
48	8	9	4	2	2	5
49	7	10	5	7	8	3
50	5	5	7	6	7	7
51	0	7	3	5	10	3
52	2	8	1	6	9	5
53	5	7	4	4	5	4
54	6	7	7	2	4	4
55	5	9	3	5	8	3
56	6	8	8	7	9	8
57	5	7	0	4	5	8
58	7	5	7	5	8	7
59	2	10	10	2	1	8
60	7	5	8	2	6	7
61	6	7	6	6	8	8
62	7	5	4	4	9	6
63	3	10	6	8	6	4
64	8	8	8	6	7	7
65	8	10	2	7	7	5
66	7	9	9	9	10	5
67	5	8	8	5	6	7
68	3	7	8	8	9	9
69	4	7	5	8	8	8
70	0	6	0	0	5	6
71	2	5	5	5	3	3
72	5	9	9	6	8	2
73	7	9	3	6	8	4
74	4	10	0	0	0	0
75	7	6	8	7	9	8
76	7	5	3	5	6	5

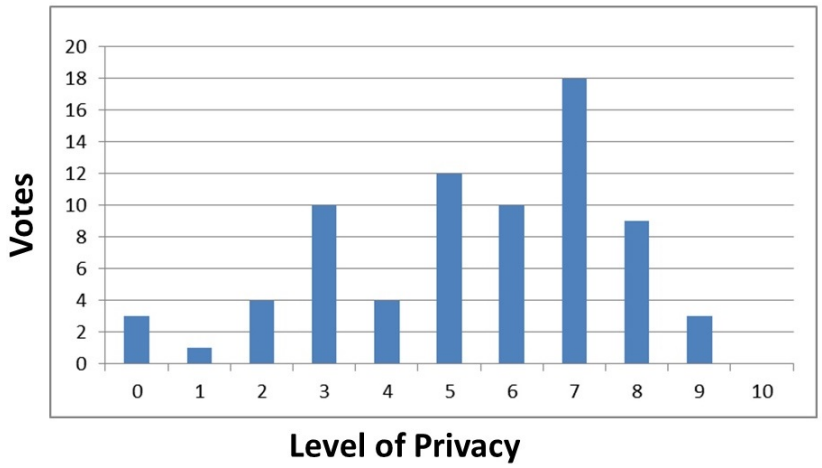


Figure A.1 – Data Fragmentation technique votes

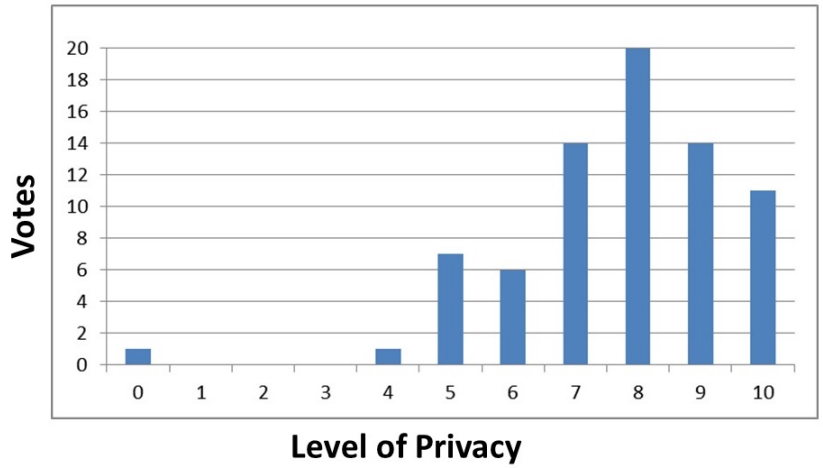


Figure A.2 – Data encryption technique votes

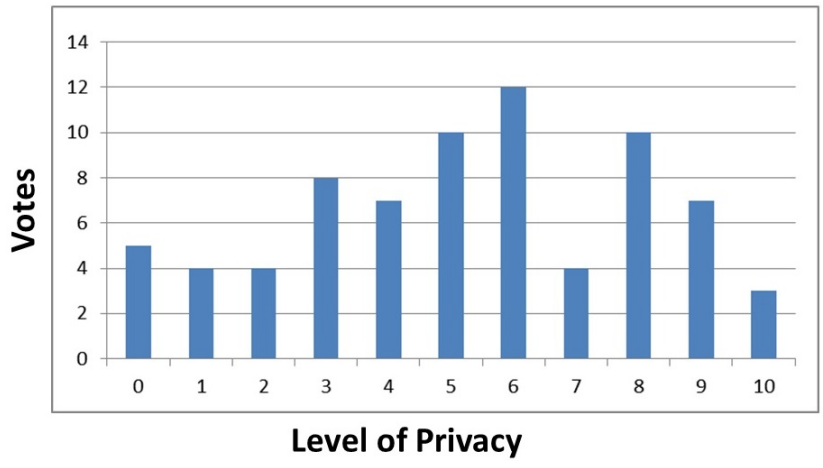


Figure A.3 – Metadata encryption technique votes

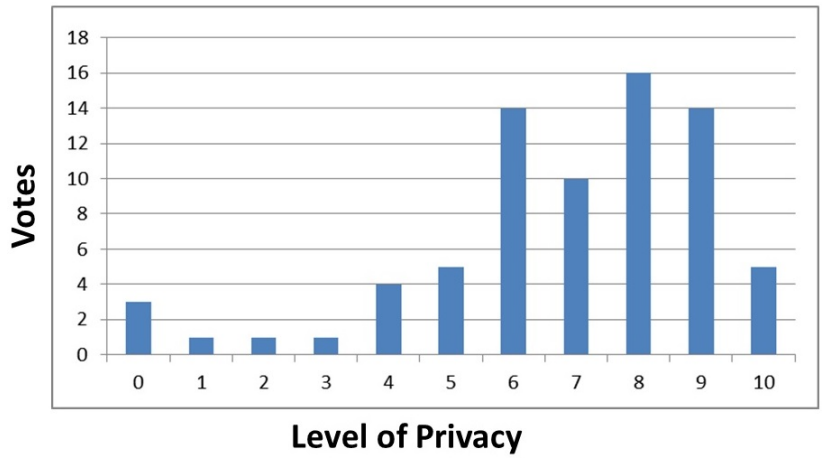


Figure A.4 – TOR technique votes

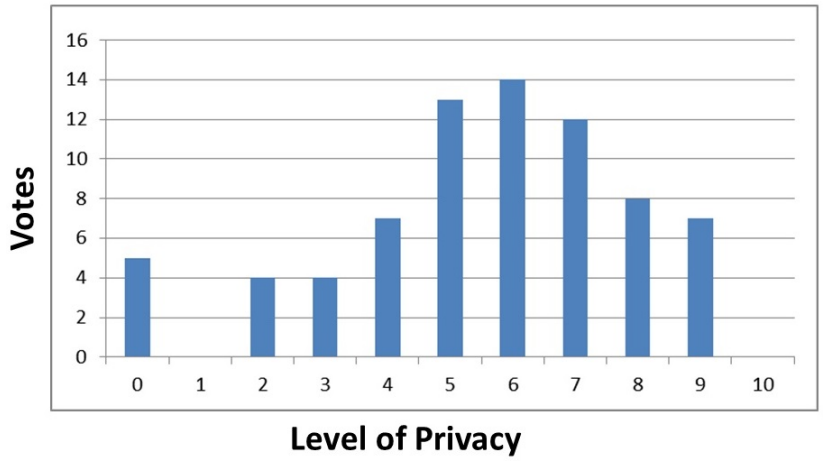


Figure A.5 – VPN-Proxy technique votes

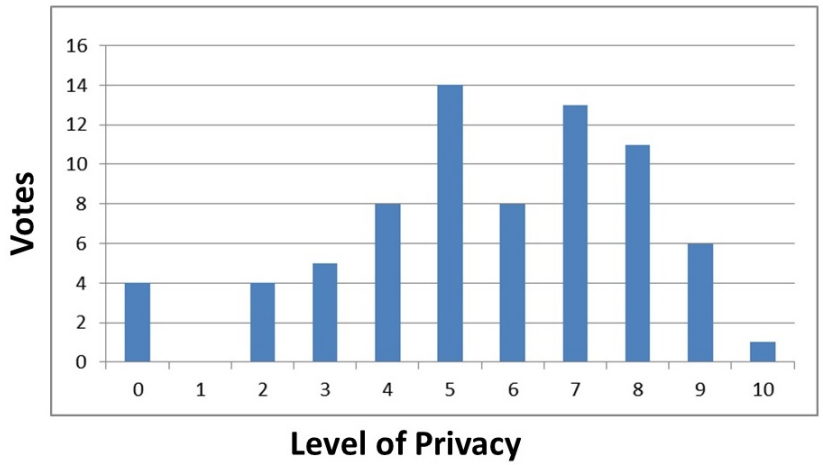


Figure A.6 – Federated Identity technique votes

Analyzing all these data, we can conclude that the values have a little variation from technique to technique, which means that they give, according to the participants' point of view,

Table A.2 – Statistical results derived from the survey

Measure	**Data Fragmentation**	**Data Encryption**	**Metadata Encryption**	**VPN-Proxy**	**TOR**	**Federated Identity**
Average	5,45	7,63	5,24	5,53	6,84	5,64
Variation	4,93	3,18	7,60	5,43	5,45	5,54
Median	6,00	8,00	5,00	6,00	7,00	6,00
Mode	7,00	8,00	6,00	6,00	8,00	5,00
St. Deviation	2,22	1,78	2,76	2,33	2,33	2,35

the same benefits with respect to privacy. However, we do not agree with these values since we strongly believe that some techniques are more effective than others. For example, if we use data encryption we will have more privacy than metadata encryption. We concluded that the way we constructed the survey may have influenced users judgement, or even that was not clear for them the problems and the purpose of each technique. Thus, we decided to stick with the other method to evaluate users' privacy since this survey was not sufficient to allow a good evaluation of users perspective related to cloud data storage.

APPENDIX B – Analysis of CSP's key management

In this appendix, we show how some CSPs handle their cryptography keys.

B.1 Introduction

A study was conducted on commercial solutions available nowadays to store users data in the cloud. Their designs were analyzed in order to understand how they work. In Chapter 4, a discussion of these CSPs was made, analyzing mostly how they approach the security requirements and presenting some of their main characteristics. In this section, further information regarding these CSPs will be presented, highlighting how they manage the cryptographic keys in their solutions.

B.1.1 Encryption App - ownCloud

In ownCloud, users can enable an optional application called Encryption App aimed to protect their data through cryptography. This way, all their data is encrypted on the server side. In the encryption process, presented in Fig. B.1, all data is encrypted using a random and unique symmetric key (DEK or Data Encryption Key) with AES algorithm. This DEK is wrapped by a key called shared key (KEK or Key Encryption Key) using RC4 algorithm. User's public key is then used to encrypted this KEK. Users can enable the password recovery feature or even choose to share their data using links. In these cases, the Shared Key (KEK) will also be encrypted using the administrator's public key (Figs. B.3 and B.4) and/or the Sharing Public Key(Figs. B.5 and B.6). In the decryption process (Fig. B.2), users need to provide their passwords to the system. A symmetric key will be derived from their passwords and used to unwrap user's private key. Then, the Shared Key is decrypted and also the DEK, used to open the data. The process of how user's key is created is illustrated in Figs. B.7 and B.8.

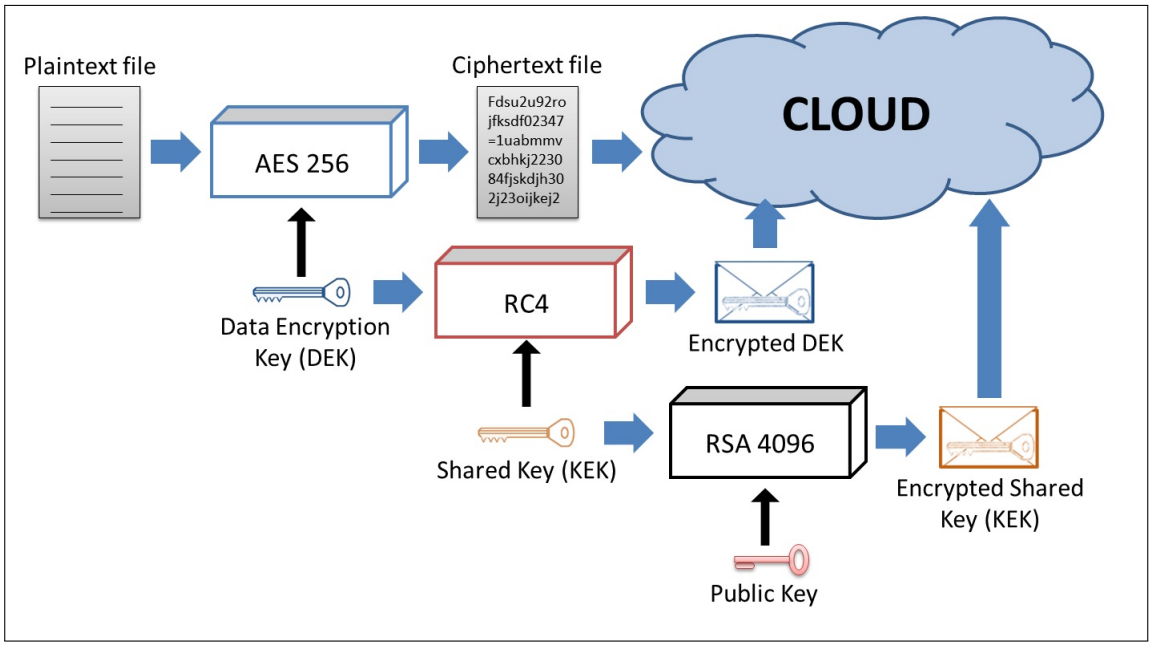


Figure B.1 – Encrypting data with ownCloud Encryption App

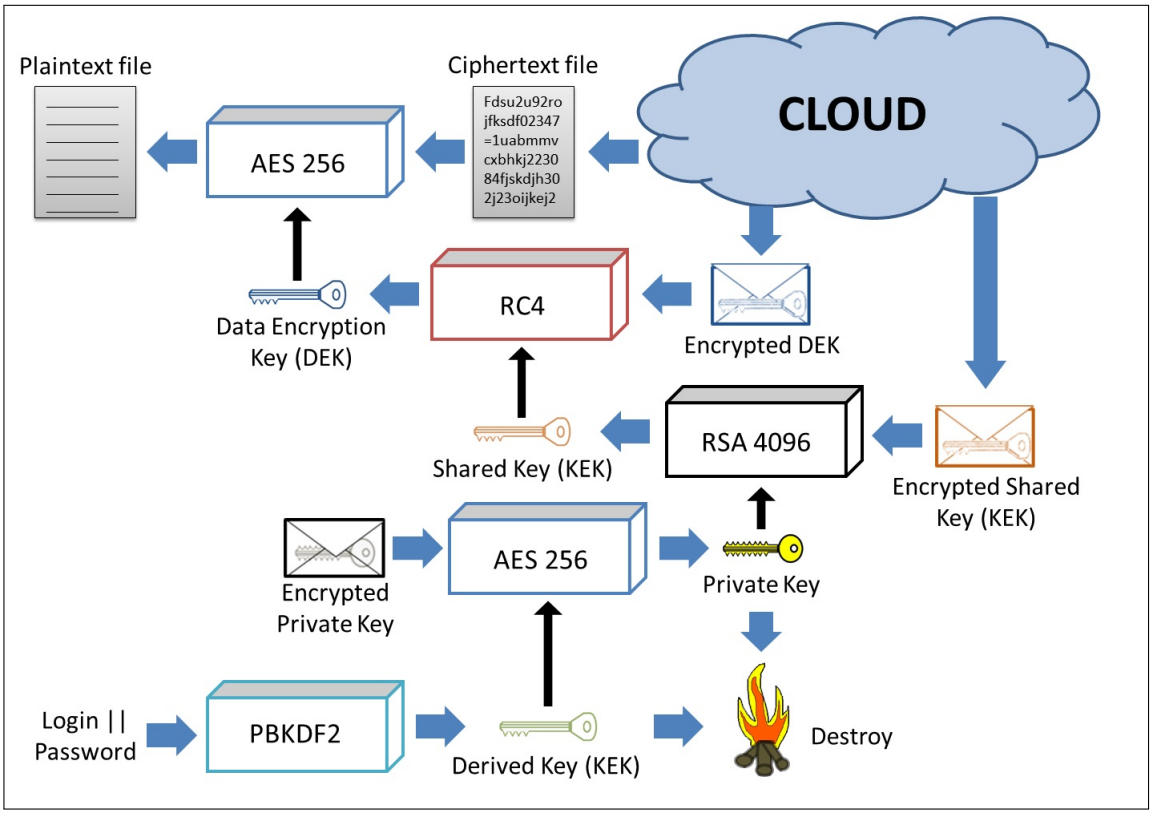


Figure B.2 – Decrypting data with ownCloud Encryption App

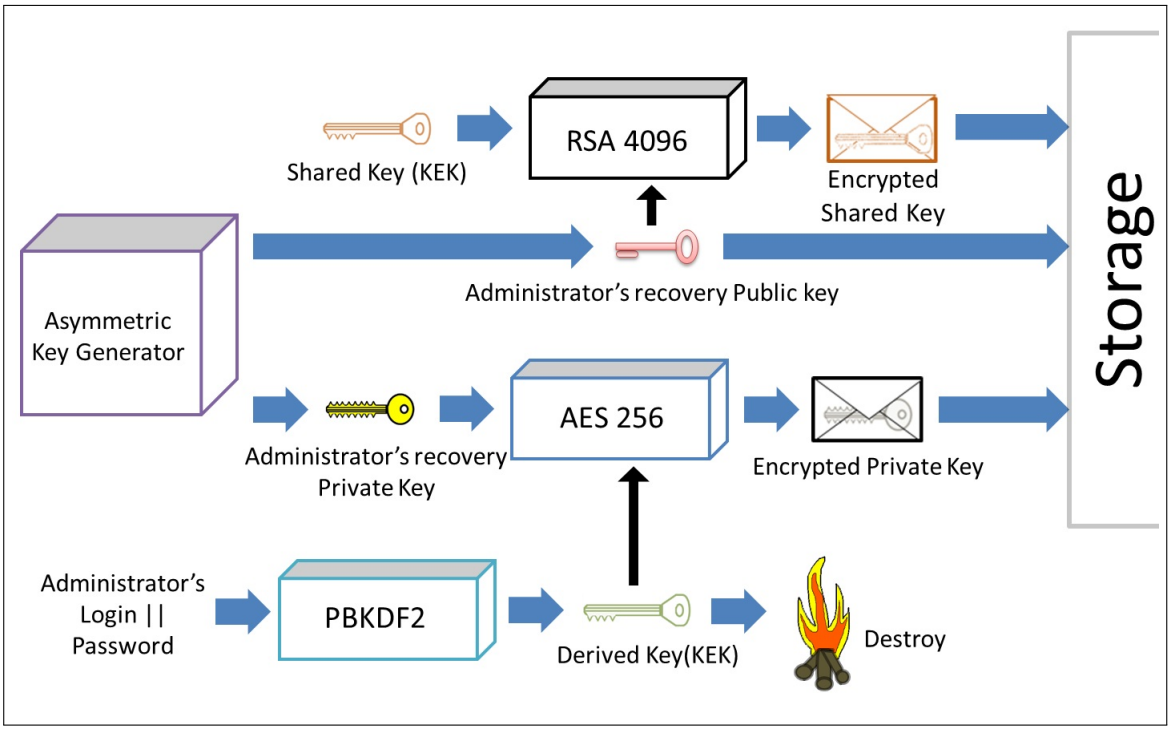


Figure B.3 – Password recovery feature - ownCloud key generation

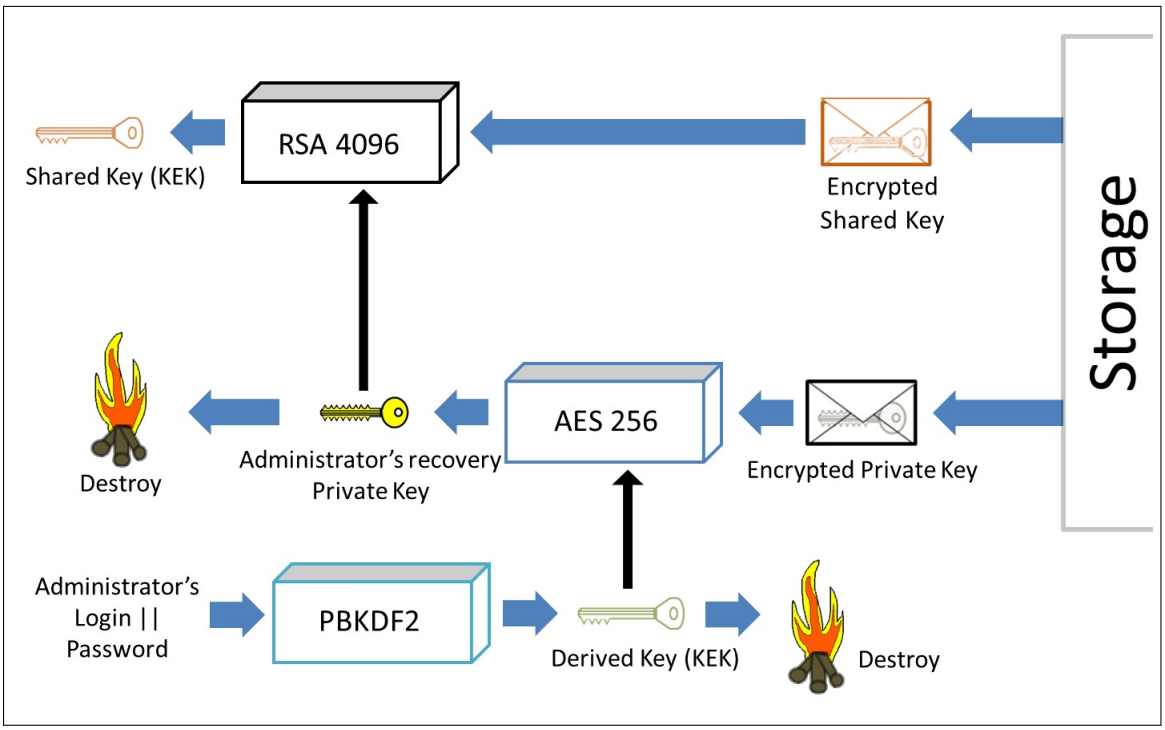


Figure B.4 – Password recovery feature - ownCloud key recovery

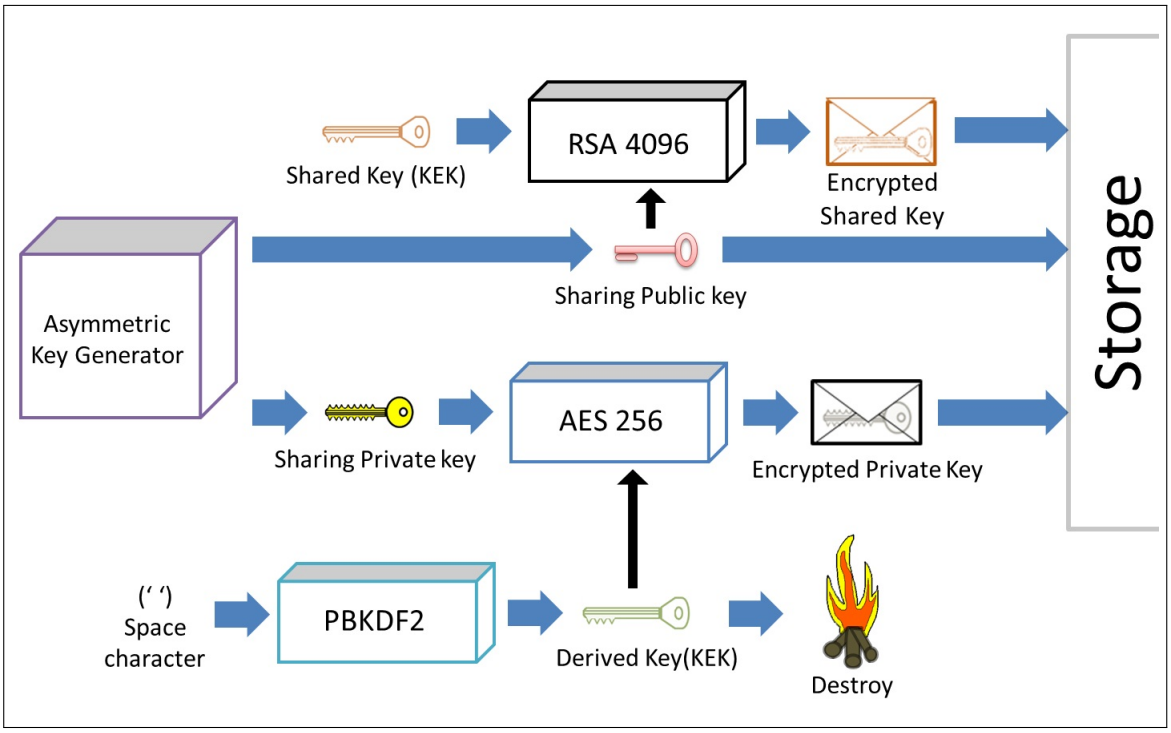


Figure B.5 – Sharing files through links - key generation

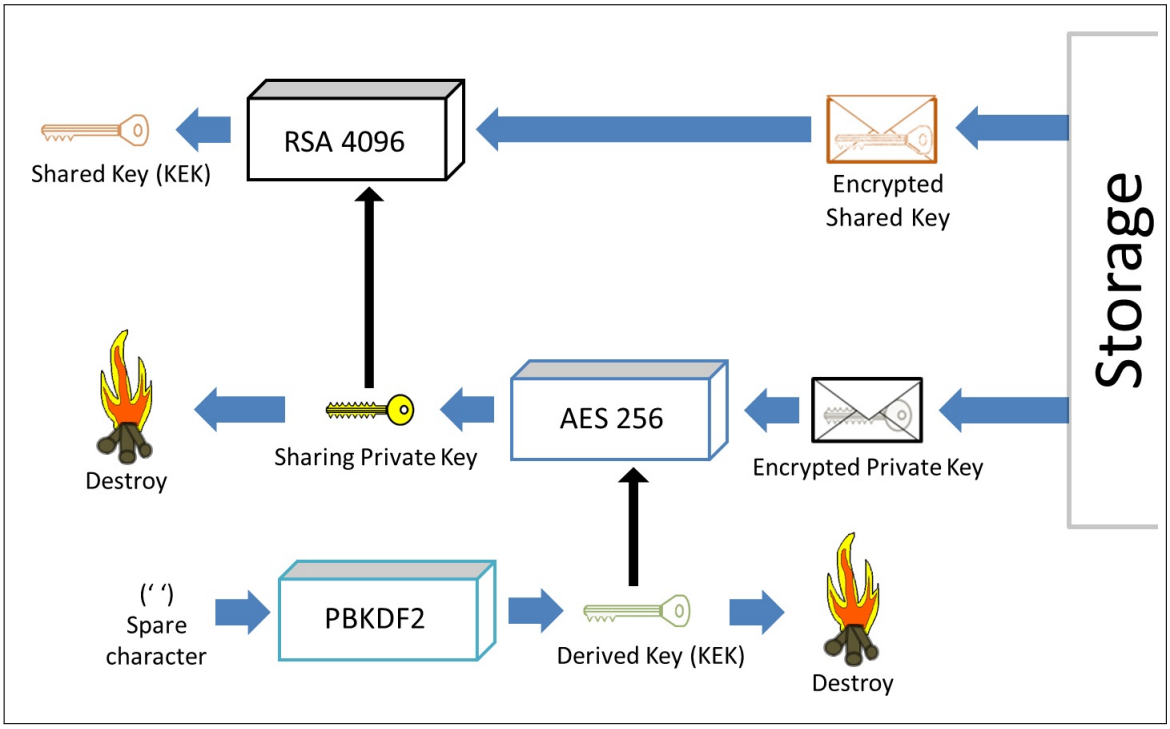


Figure B.6 – Sharing files through links - key recovery

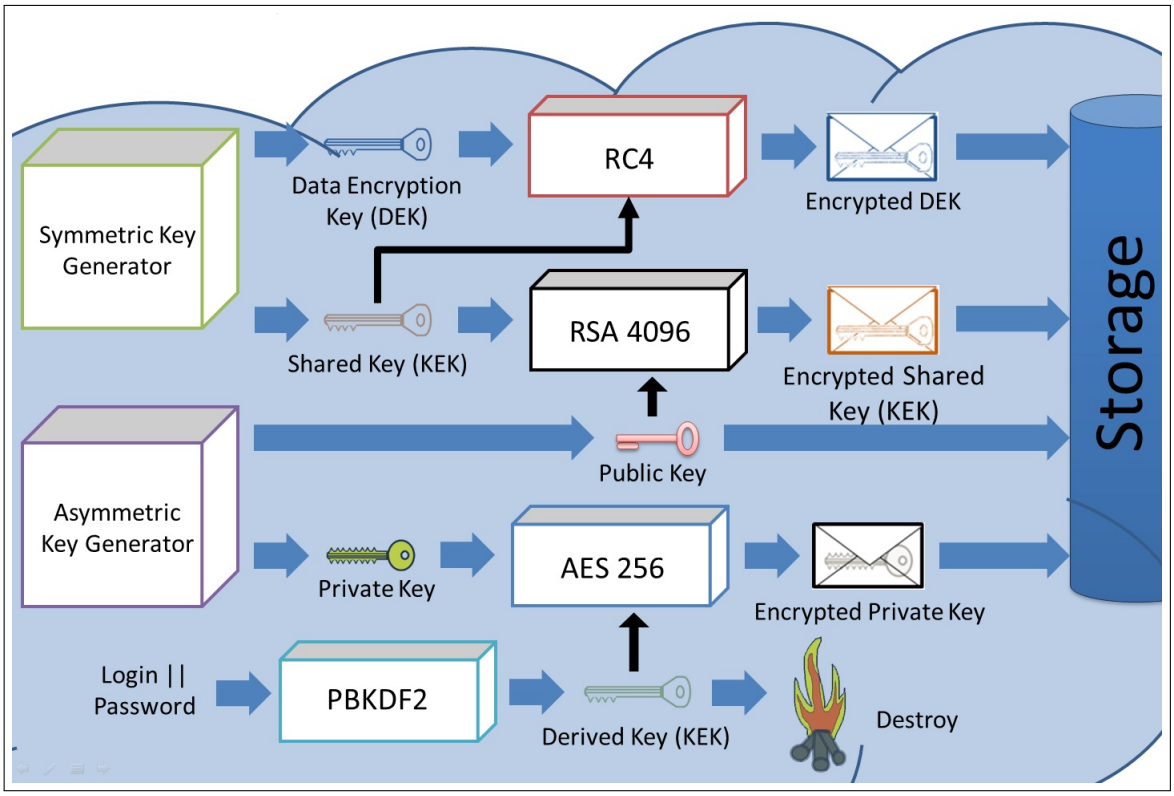


Figure B.7 – ownCloud key generation

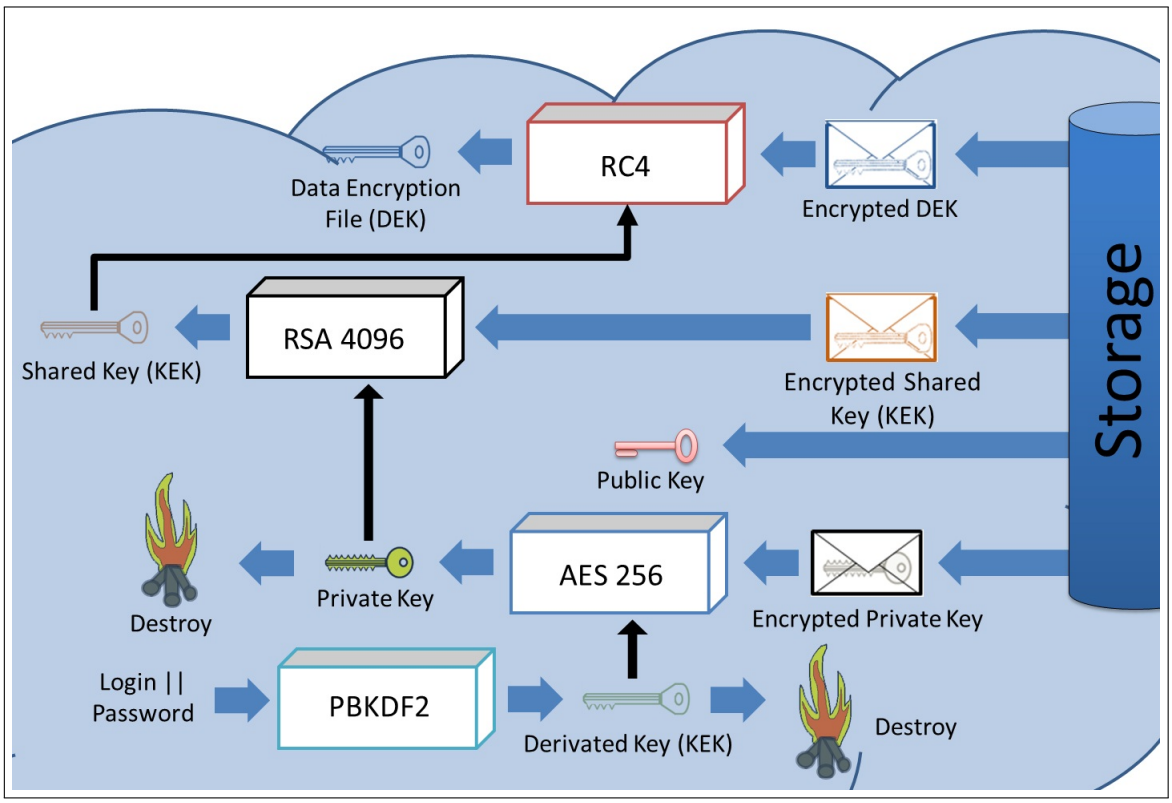


Figure B.8 – ownCloud key recovery

B.1.2 SpiderOak

SpiderOak uses symmetric encryption to protect users data. The encryption process is illustrated in Fig. B.9, where each file has a DEK created from its own content. Then, a Folder Key (KEK) is used to encrypt the DEK and it is wrapped by a symmetric key derived from users' password using the PBKDF2 algorithm. In the decryption process, presented in Fig. B.10, users need to supply their password in order to generate the derived key again, used to unwrap the folder key, which it is used to decrypt the DEK responsible for decrypting data.

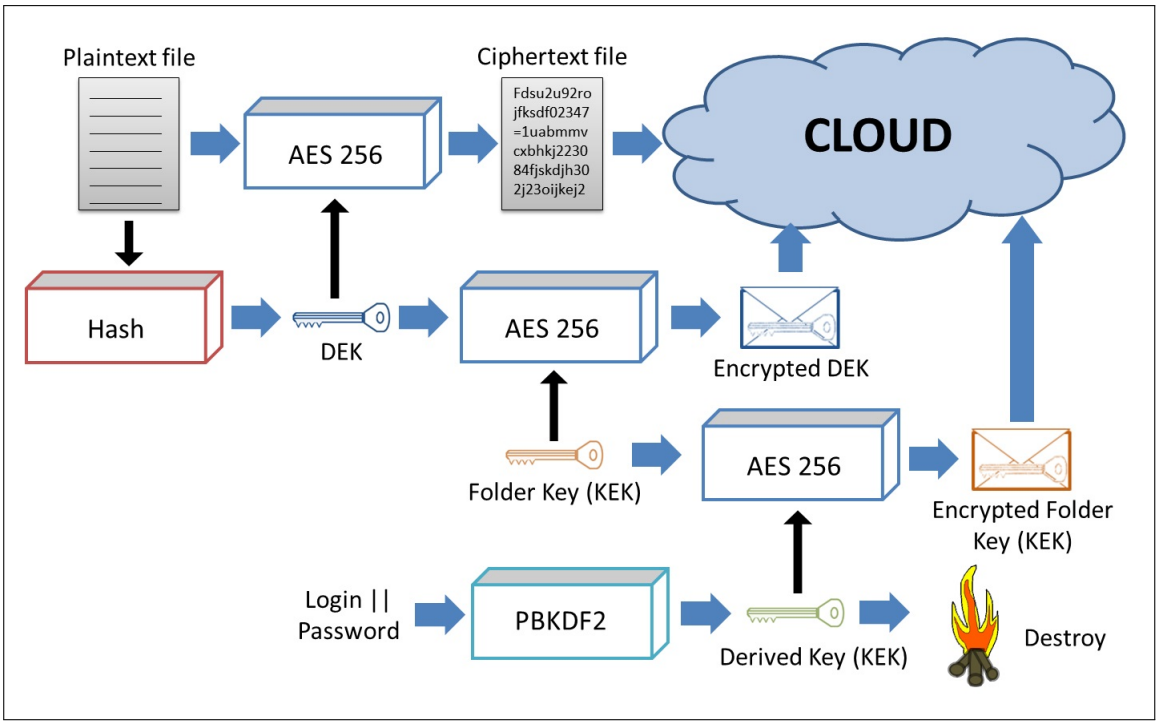


Figure B.9 – Encryption process on SpiderOak

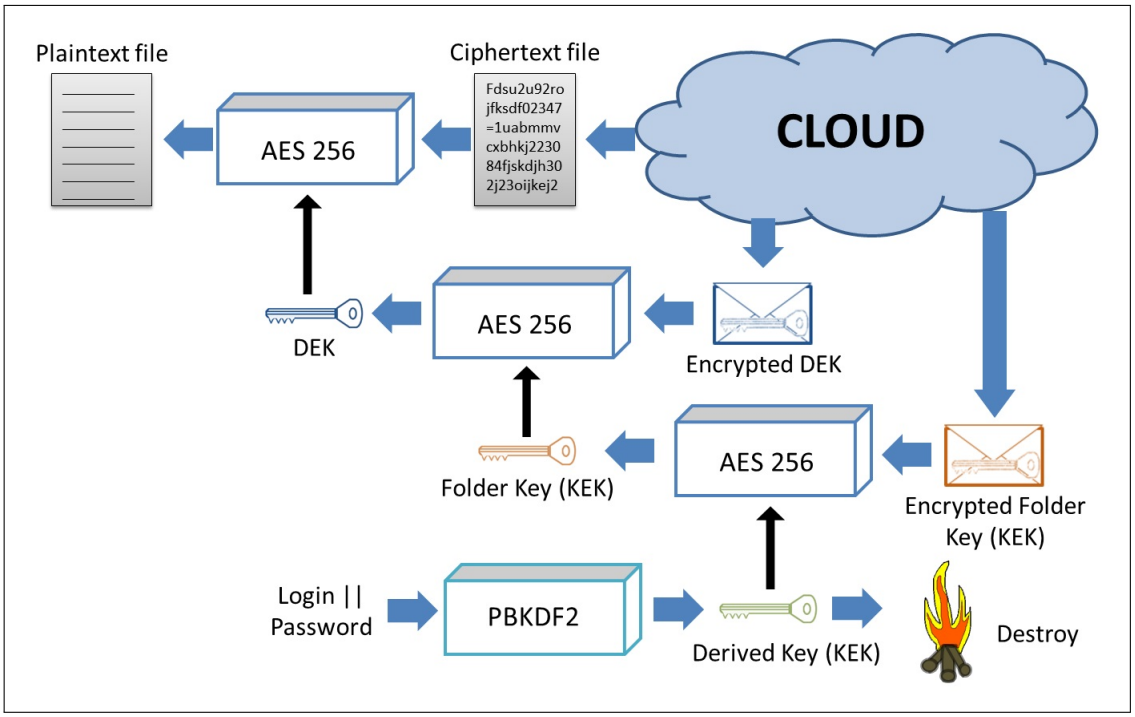


Figure B.10 – Decryption process on SpiderOak

B.1.3 BoxCryptor

BoxCryptor adopts a combination of symmetric and asymmetric encryption. Data is encrypted first with a DEK, which is wrapped by user's public key (Fig. B.11). In order to allow users to access their data, BoxCryptor uses their own password to derived a symmetric key and unwrap their private key, stored in encrypted form. Then, the DEK is opened using the user's private key and used to decrypt users data (Fig. B.12).

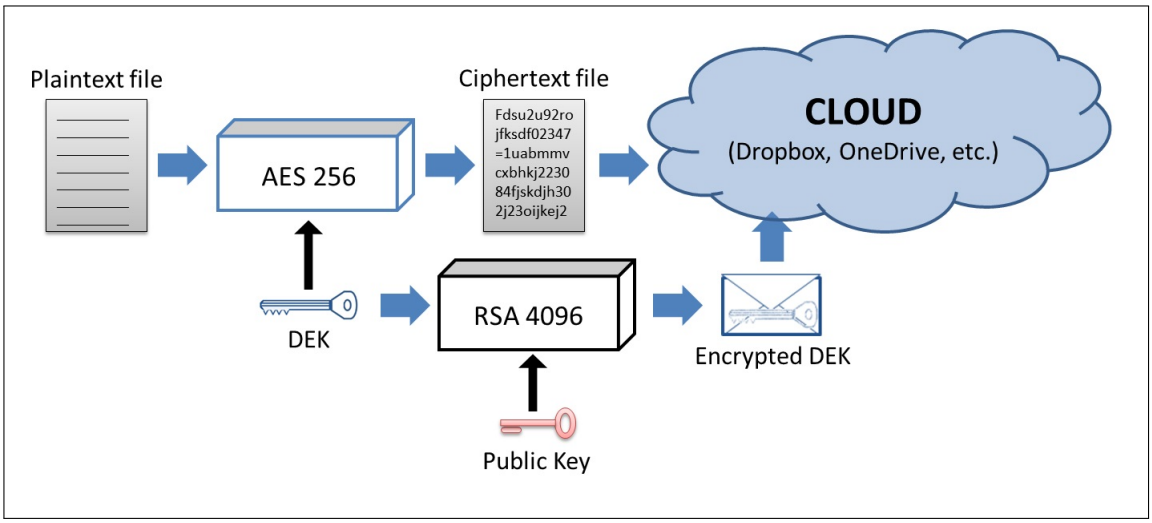


Figure B.11 – Storing users data in BoxCryptor: Encryption

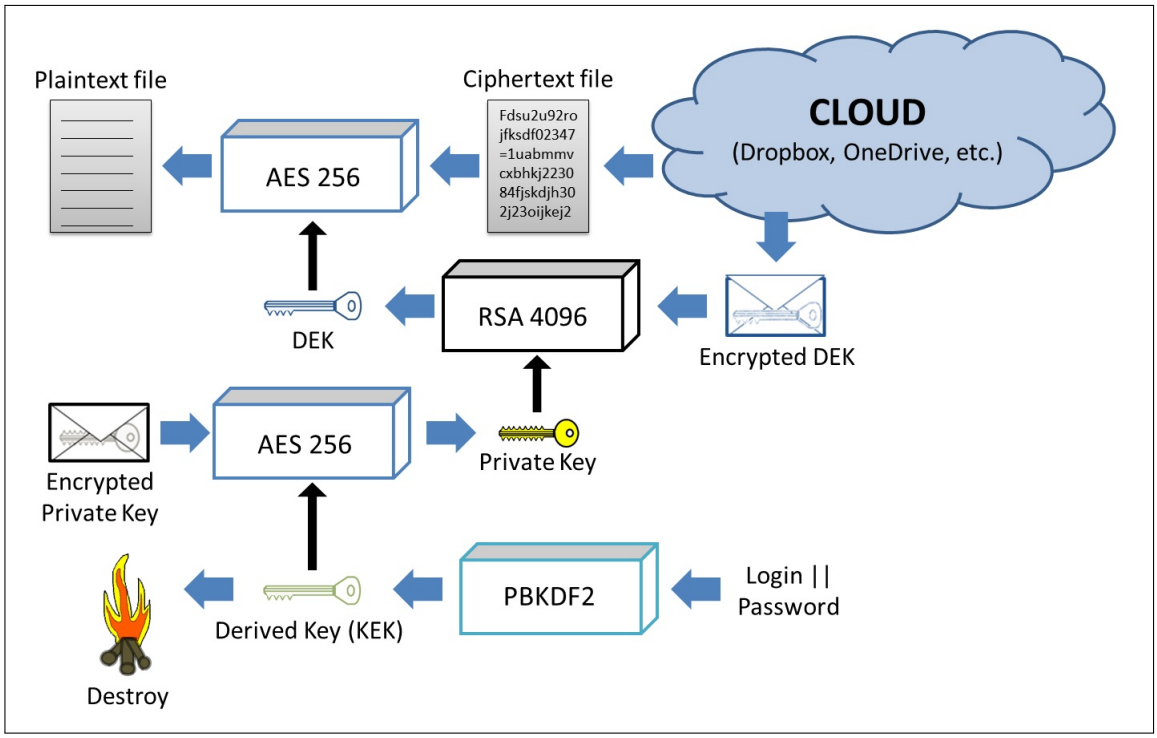


Figure B.12 – Accessing users data in BoxCryptor: Decryption

B.1.4 Credeon

Credeon uses only symmetric encryption to protect users data. In the encryption process, presented in Fig. B.13, a random and unique DEK is created and used to encrypt user's data. This DEK is then wrapped by a symmetric key derived from user's password (KEK). In the decryption process, illustrated in Fig. B.14, after users provide their password, a symmetric key is derived and used to decrypt the DEK needed to open the data.

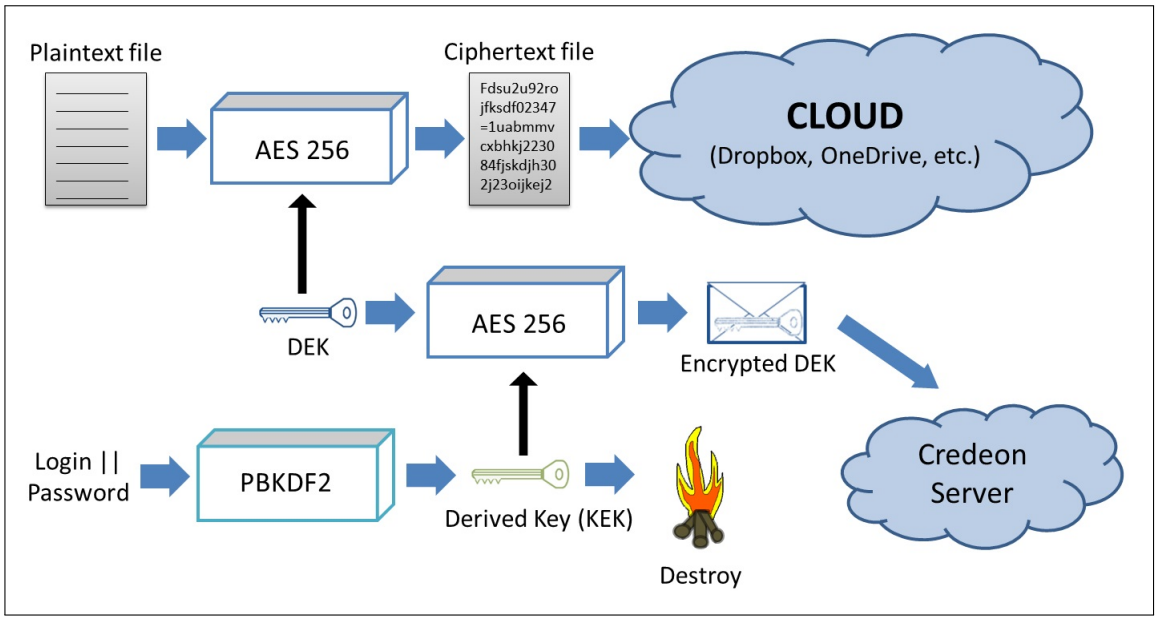


Figure B.13 – Encryption process on Credeon

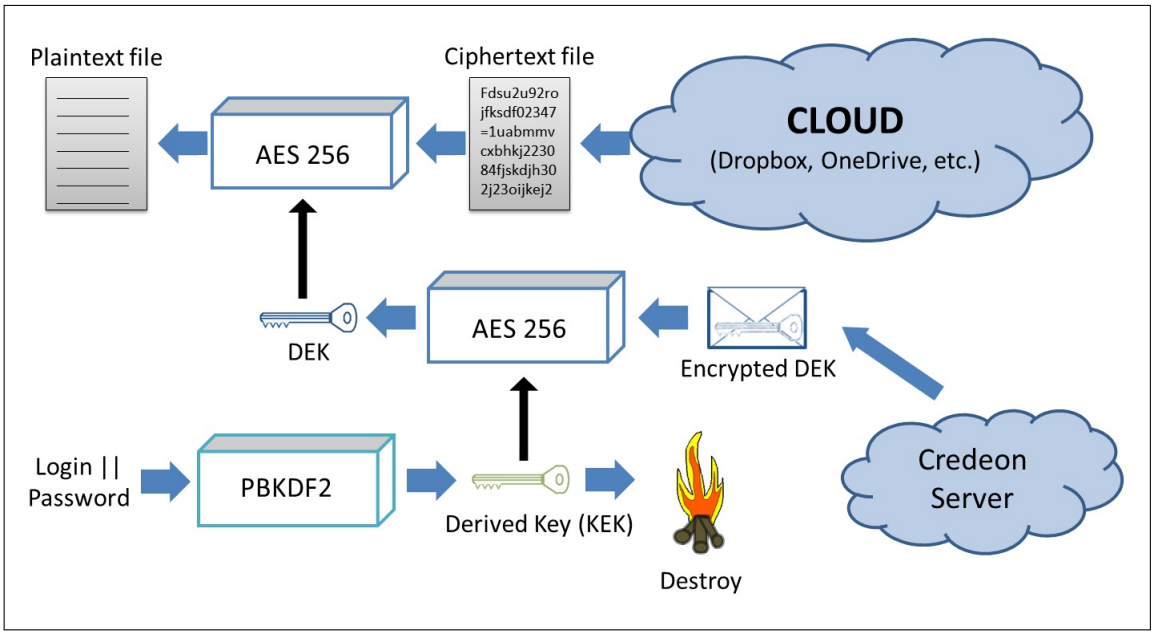


Figure B.14 – Decryption process on Credeon

B.1.5 ProtonMail

ProtonMail protects the privacy of their users by the encryption of all their emails. It combines the symmetric and asymmetric cryptography for this end, encrypting the recipient's message with a symmetric key (DEK) and wrapping this key using the public key belonging to the recipient (Fig. B.15). In order to access this encrypted email, the recipient has to provide his password to ProtonMail application, which is used to unwrap his private key. This key is then used to decrypt the DEK, responsible for opening the message (Fig. B.16). However, this process works only for ProtonMail users. If an user wants to send encrypted message for someone who do not use this email provider, a secret must be agreed beforehand. The email will be encrypted using an asymmetric algorithm (Fig. B.17) and the secret established. After receiving the encrypted message, the recipient will need to access an interface provided by ProtonMail and enter the right secret in order to decrypt the message (Fig. B.18).

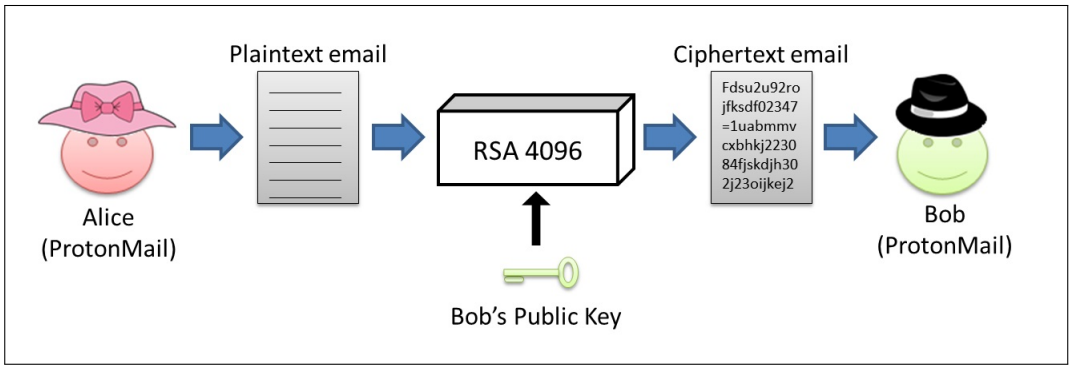


Figure B.15 – Encrypting and sending messages - ProntonMail users

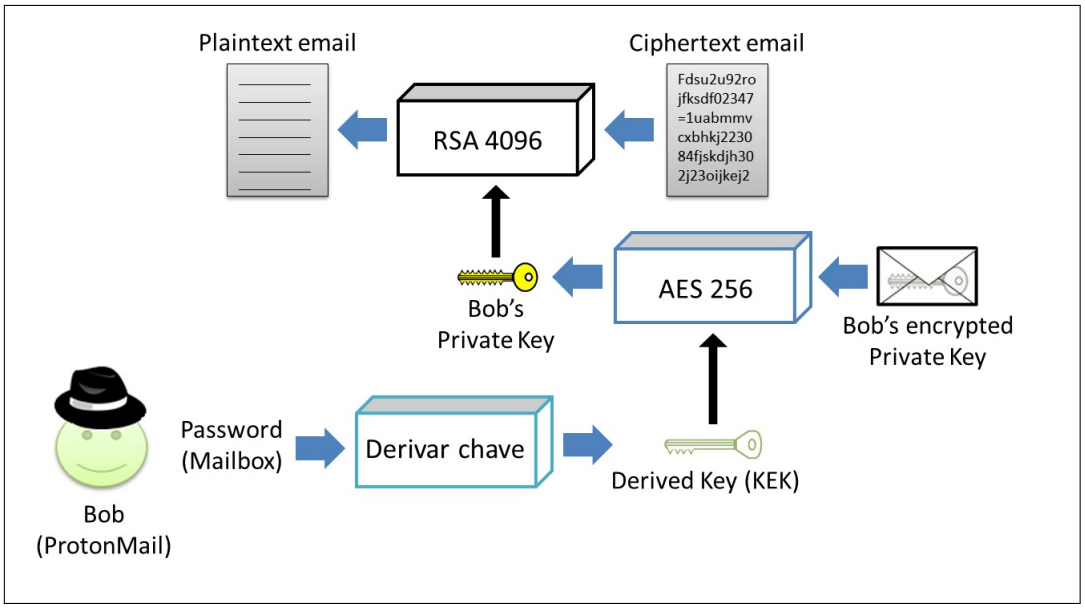


Figure B.16 – Receiving and decrypting messages - ProntonMail users

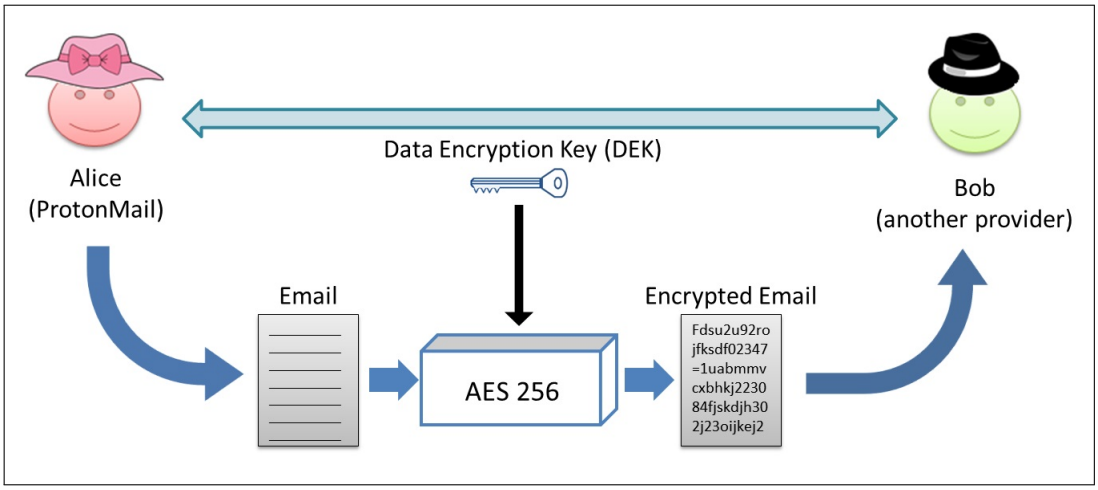


Figure B.17 – Encrypting and sending messages from ProntonMail to another provider

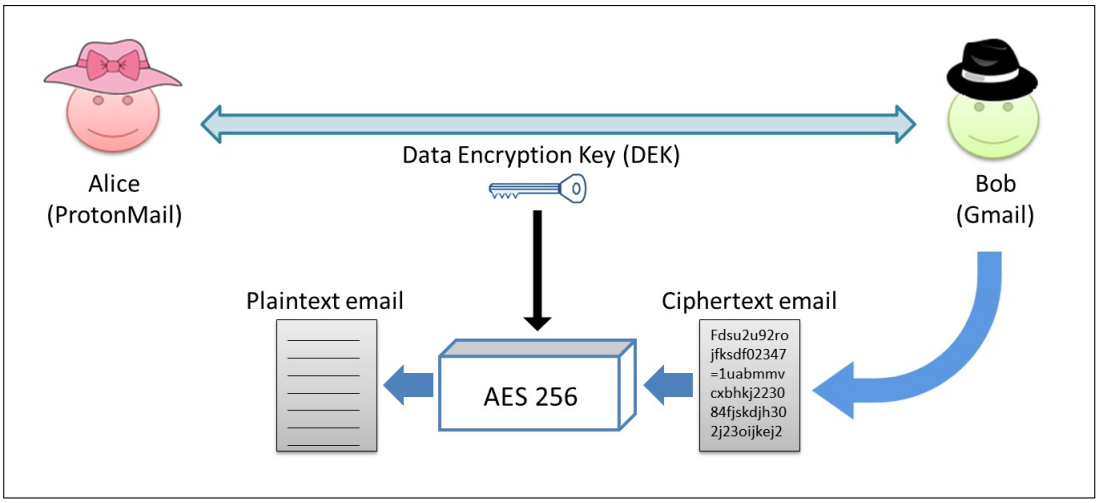


Figure B.18 – Receiving and decrypting messages from another provider to ProntonMail

B.1.6 Cyphertite

Cyphertite adopts only symmetric encryption to protect their users, using AES algorithm in XTS mode. Before sending data to the cloud, every file is first broken in several fixed-size fragments, which are compressed and encrypted. Then, all fragments are sent to be stored in the cloud, as illustrated in Fig. B.19. During the encryption process, each file chunk uses two symmetric keys, the Chunk key K_1 and Tweak key K_2, used to encrypt the file. These keys are protect by a master key K_m, which is wrapped by a secret passphrase K_s derived from user's password. This process is presented in Fig. B.20. In the decryption process, users only need to provide their passwords to the system in order to generate the K_s used to unwrap user's master key K_m, which is used to decrypt both K_1 and K_2, necessary to open the desired file. This process is illustrated in Fig. B.21.

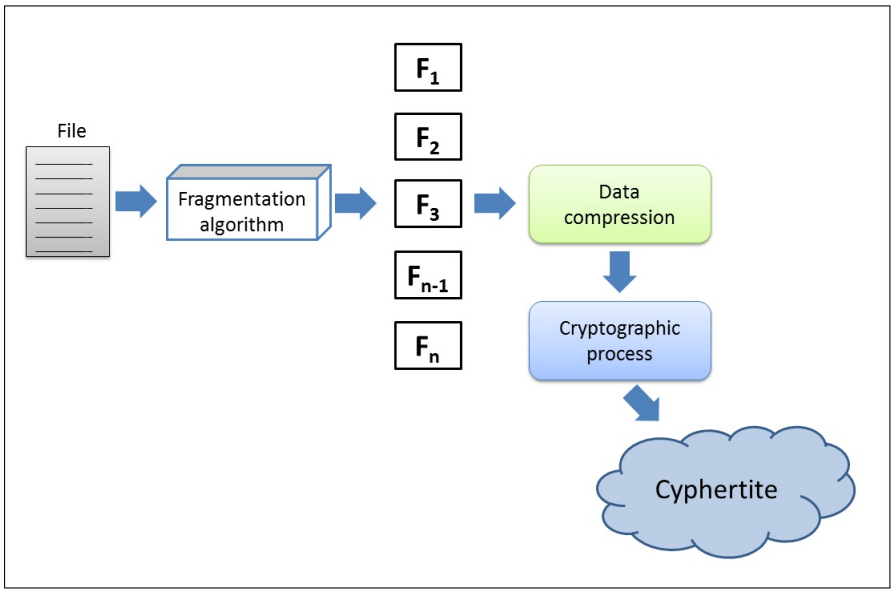


Figure B.19 – Cyphertite - Sending files encrypted to the cloud

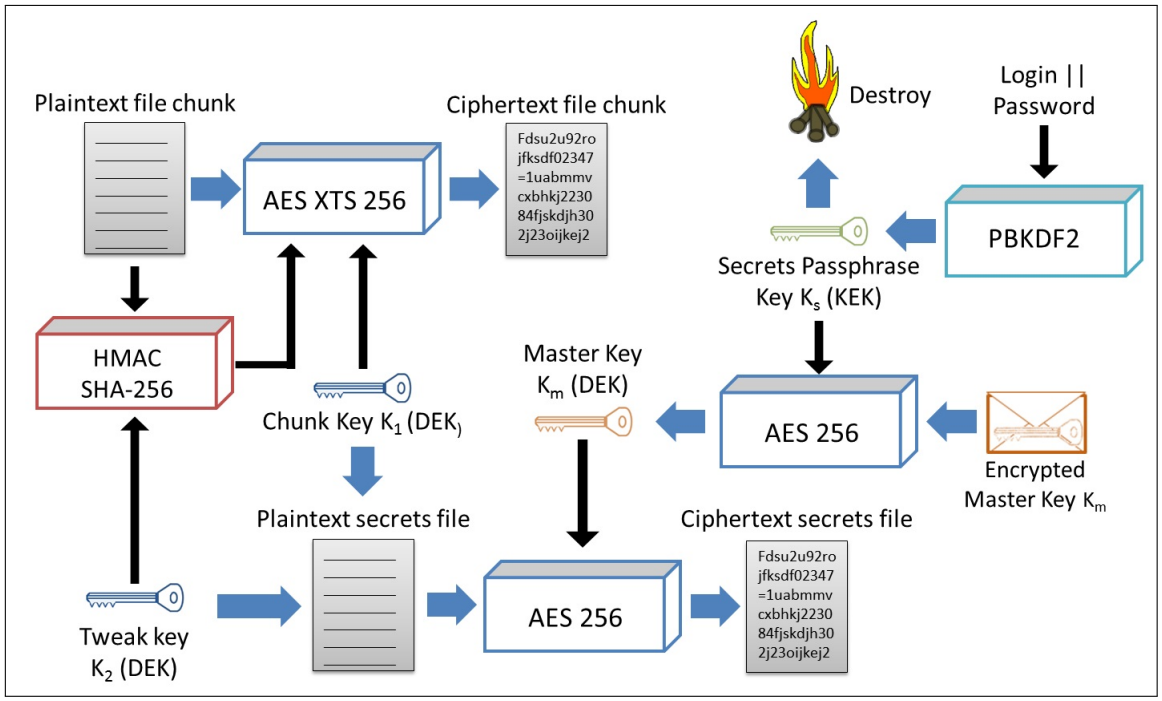


Figure B.20 – Cyphertite - Encryption model

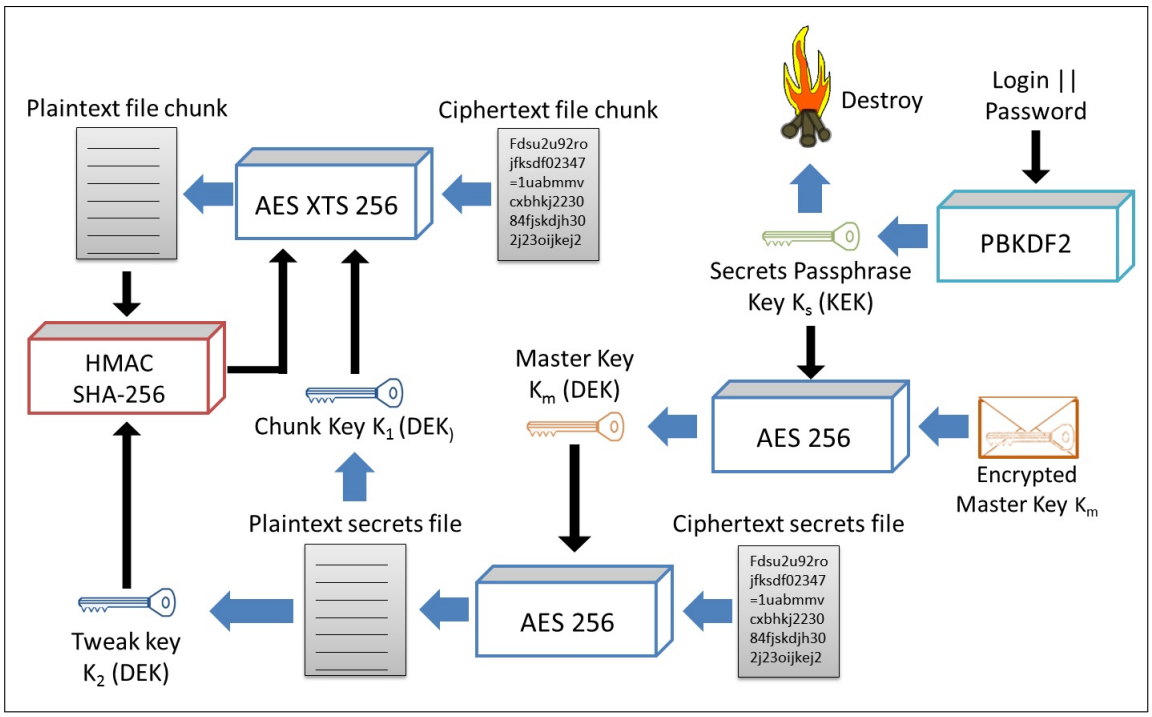


Figure B.21 – Cyphertite - Decryption model

B.1.7 Wuala

Waula uses a model where every file is encrypted using a symmetric DEK created from its own content, in order to allow deduplication. This DEK is encrypted with another symmetric key, the key folder, protected by a KEK derived from user's password, according to

Fig. B.22. In the decryption process, presented in Fig. B.23, after users supply their passwords to the system, the KEK is generated and used to decrypt the folder key corresponding to the desired file. This key is then used to open the DEK, to decrypt the file.

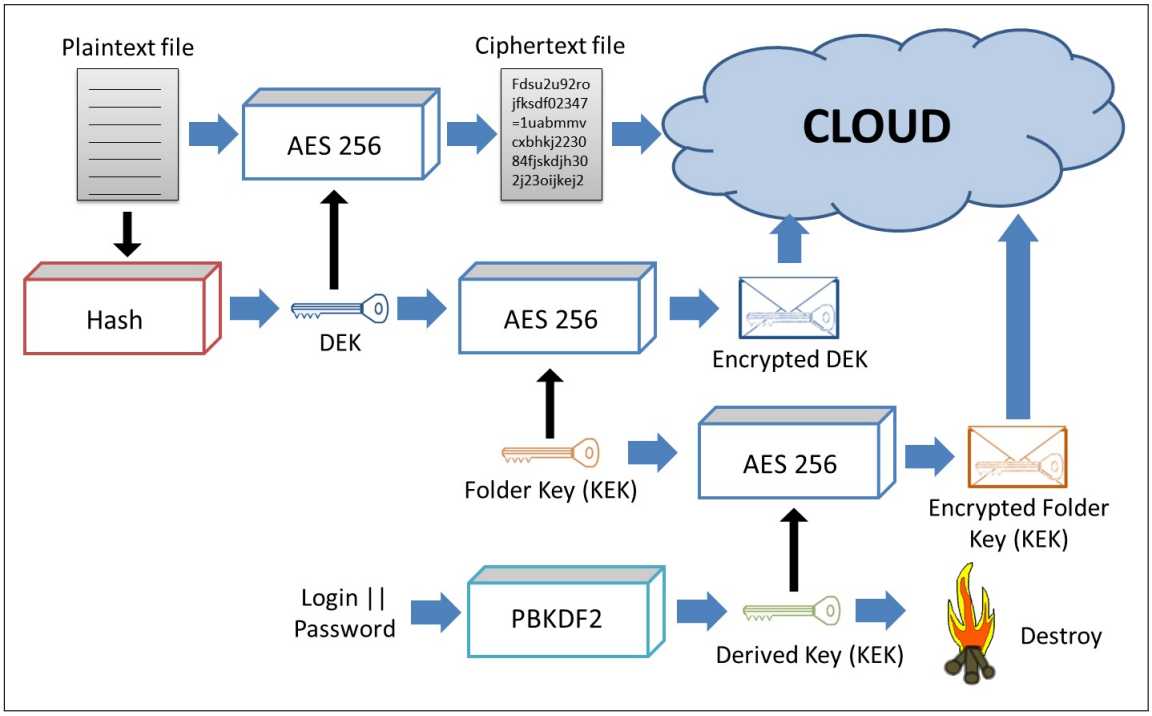


Figure B.22 – Encrypting data with Wuala

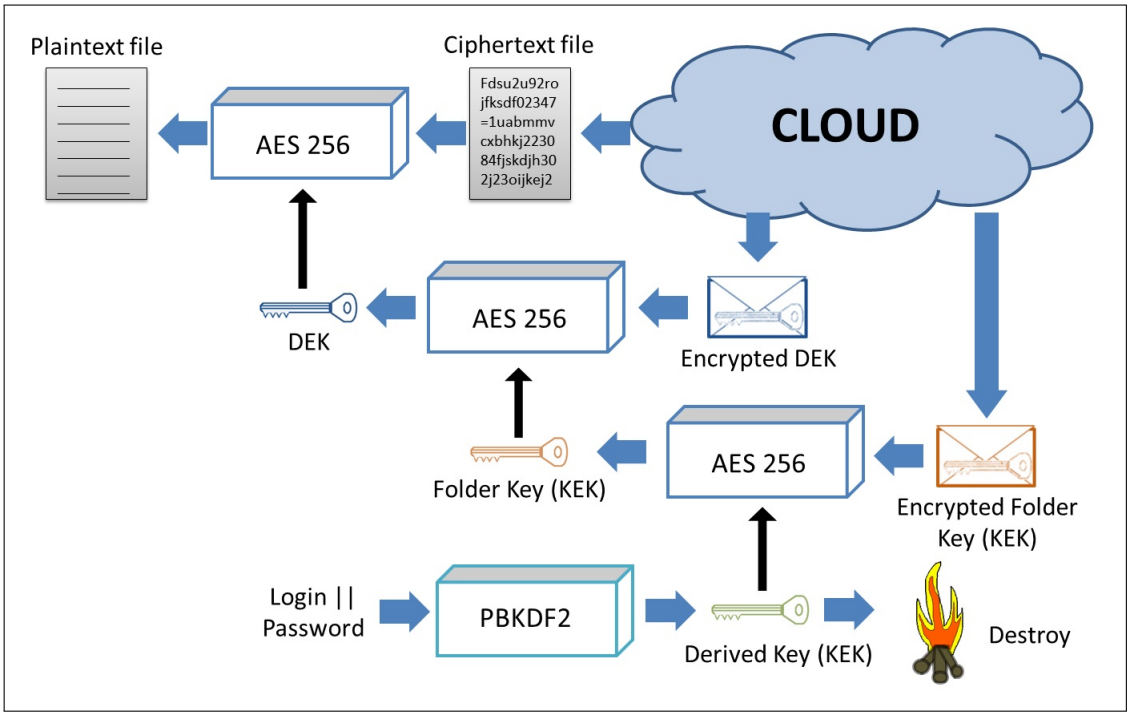


Figure B.23 – Decrypting data with Wuala

B.1.8 arxShare

This CSP protects users data through the use of unique and random DEKs. These keys are encrypted by a symmetric KEK belonging to the corresponding folder where data is placed, and users' password is used to derive a KEK to wrap the folder key and protect it. This process is presented in Fig. B.24. When sharing data, the corresponding folder key is encrypted using the recipient's public key and delivered to him, as illustrated in Fig. B.25. With the recipient' password, it is possible to decrypt his private key, open the folder key and decrypt the DEK to open the file.

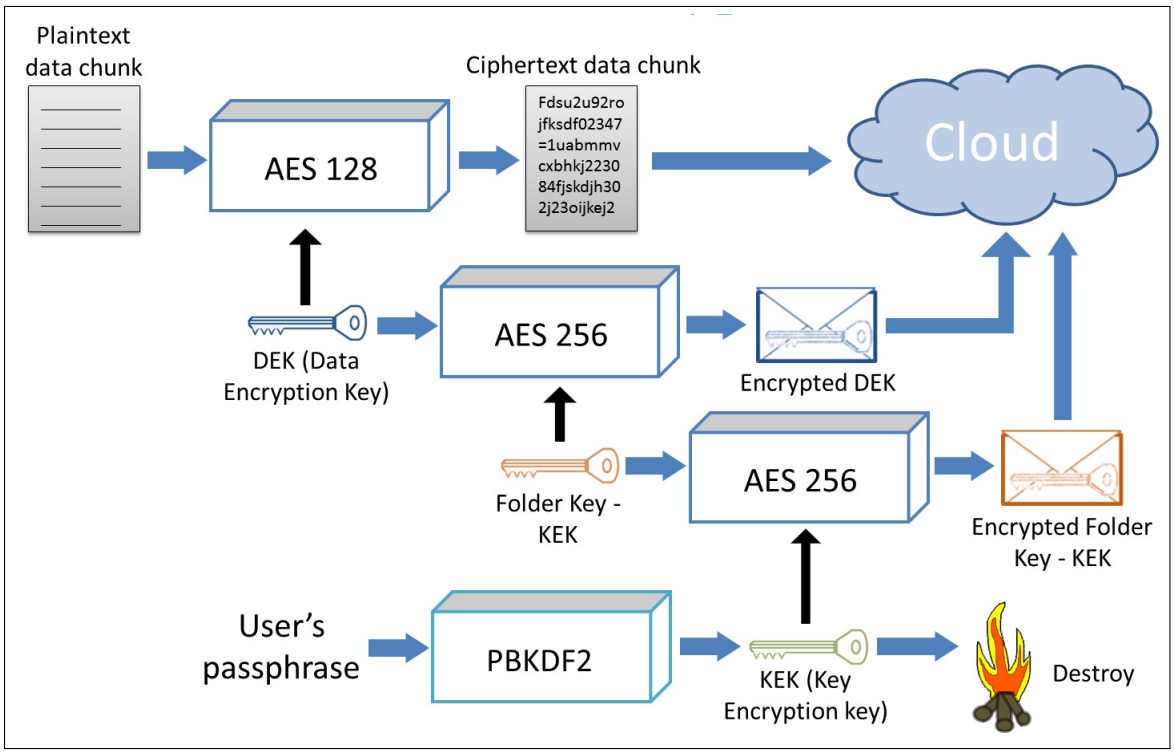


Figure B.24 – Data encryption process on arXshare

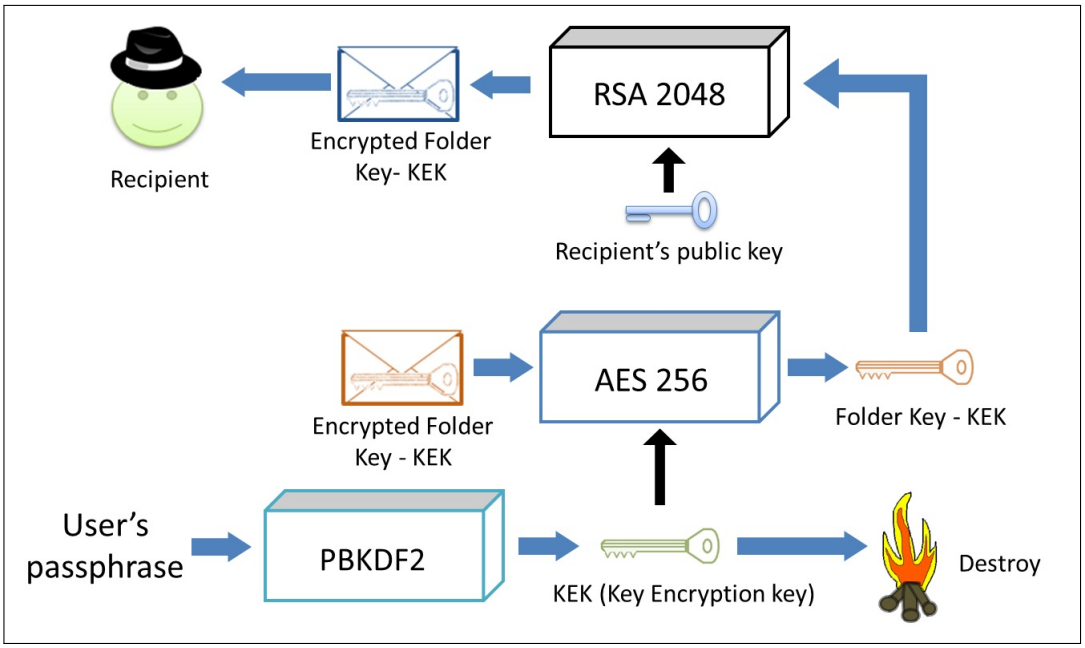


Figure B.25 – Sharing data in arXshare

B.1.9 BackBlaze

BackBlaze uses symmetric and asymmetric cryptography to protect users data. The encryption process is presented in Fig. B.26, where data is encrypted using a symmetric DEK, which is wrapped by users' public key. In Fig. B.27, users must supply their password to derive a symmetric KEK used to unwrap their private key. Then, the DEK is decrypted and used to open the data.

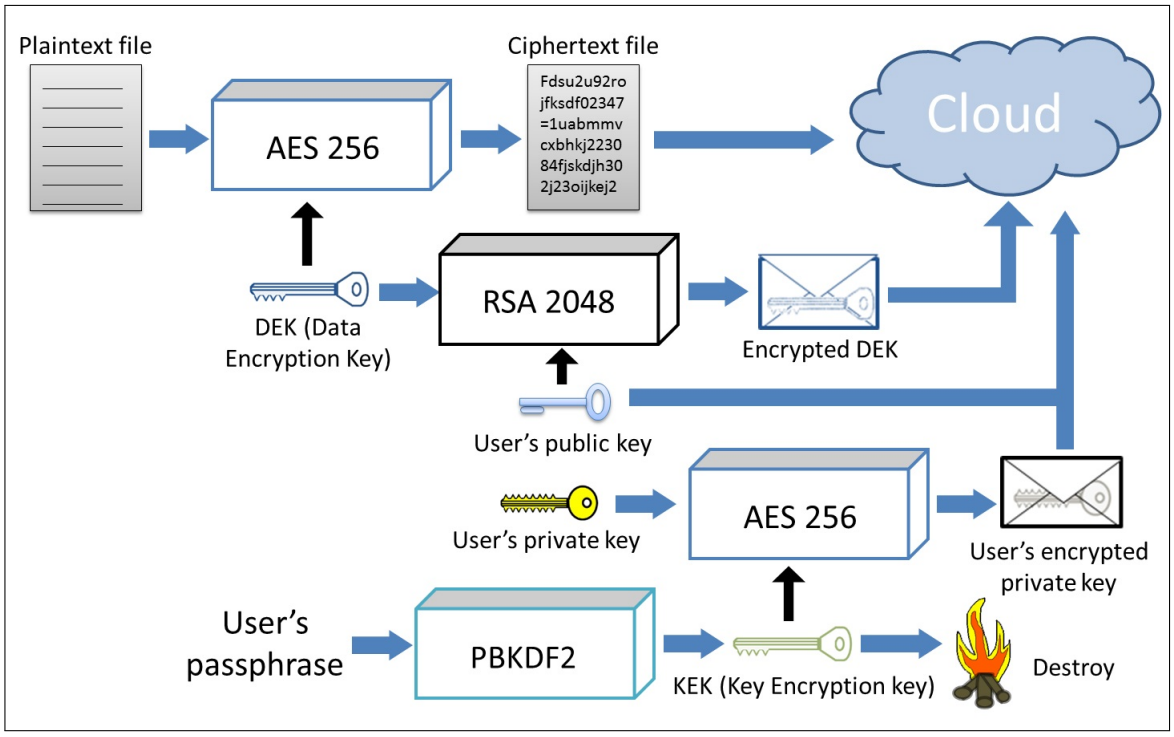


Figure B.26 – Encrypting data in BackBlaze

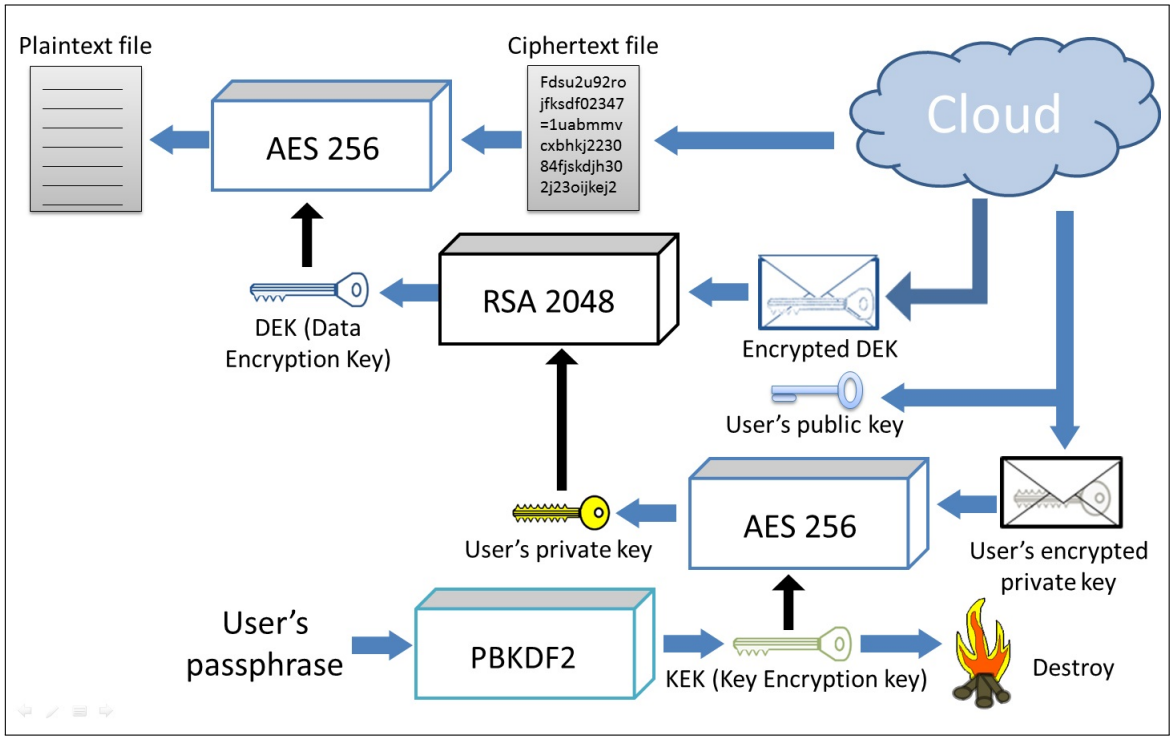


Figure B.27 – Decrypting data in BackBlaze

B.1.10 Carbonite

Carbonite uses a cryptography process similiar as Credeon. Using symmetric encryption, a random and unique DEK is generated and used to encrypt user's data, which is later

wrapped by another symmetric key derived from user' password (KEK). This process is illustrated in Fig. B.28. Users can access their files just providing their password to Carbonite, which will derive a KEK and use it to decrypt the DEK, which is used to decrypt the data, according to Fig. B.29.

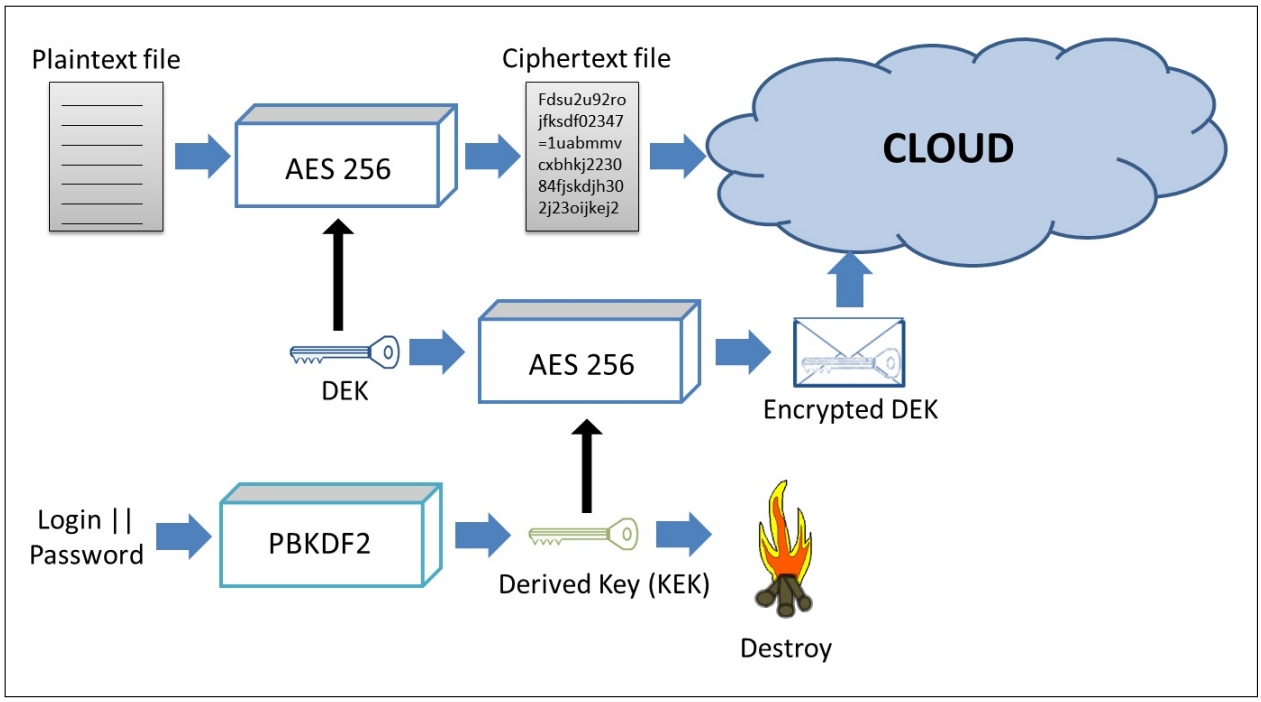


Figure B.28 – Carbonite: Protecting files

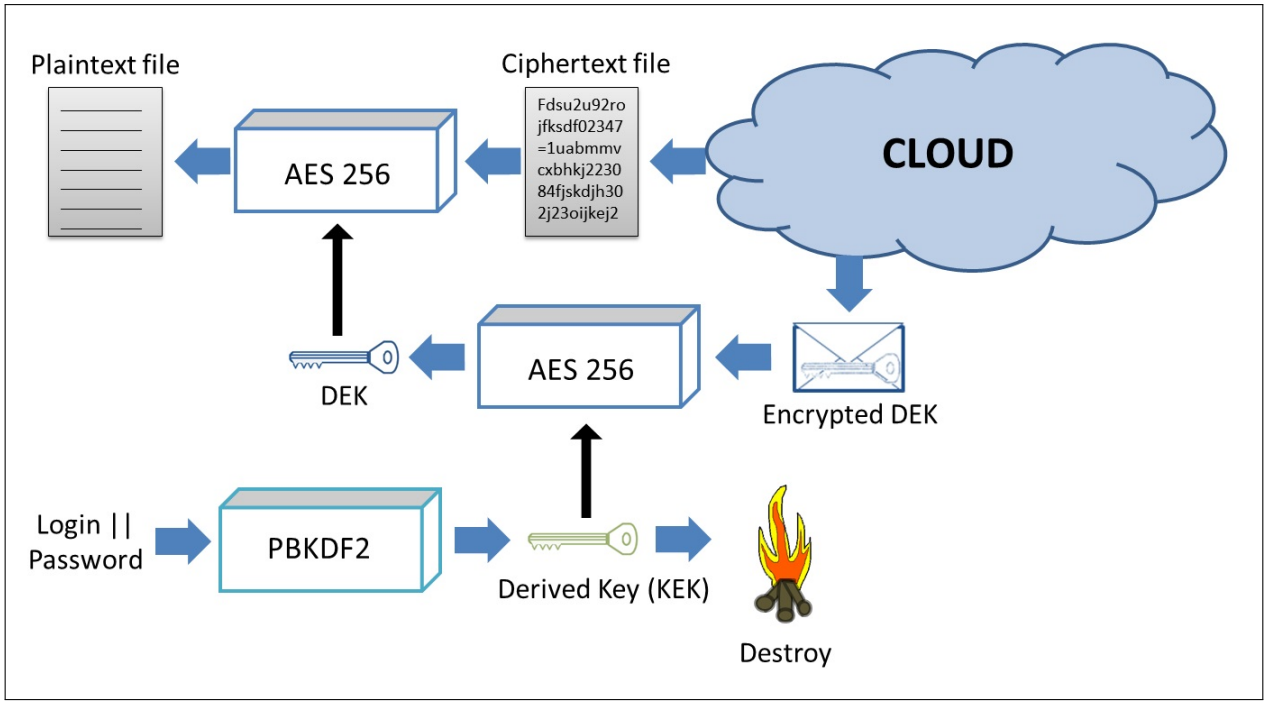


Figure B.29 – Carbonite: Accessing encrypted files

B.1.11 Mega

Mega protects its users files through symmetric encryption, using random and unique DEKs to encrypt them. The DEKs are protected by a specific symmetric master key created for each user in the system. A KEK derived from users' passwords is used to wrap this master key. The entire encryption process is illustrated in Fig. B.30. To access encrypted files, users must provide their passwords to Mega, which uses them to derive a KEK and hence decrypts the master key. The DEK is then decrypted by users' master key, used to open their files. This process can be seen in Fig. B.31.

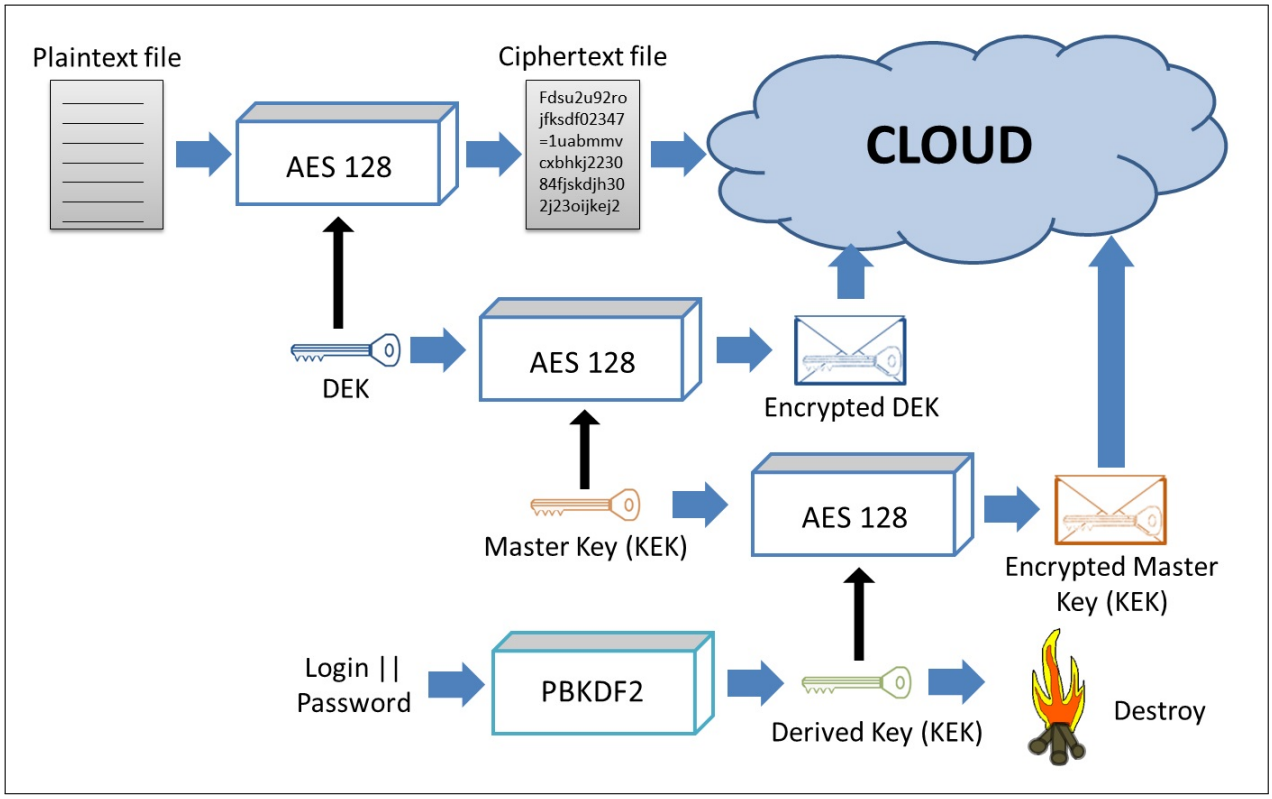


Figure B.30 – Encrypting files with Mega

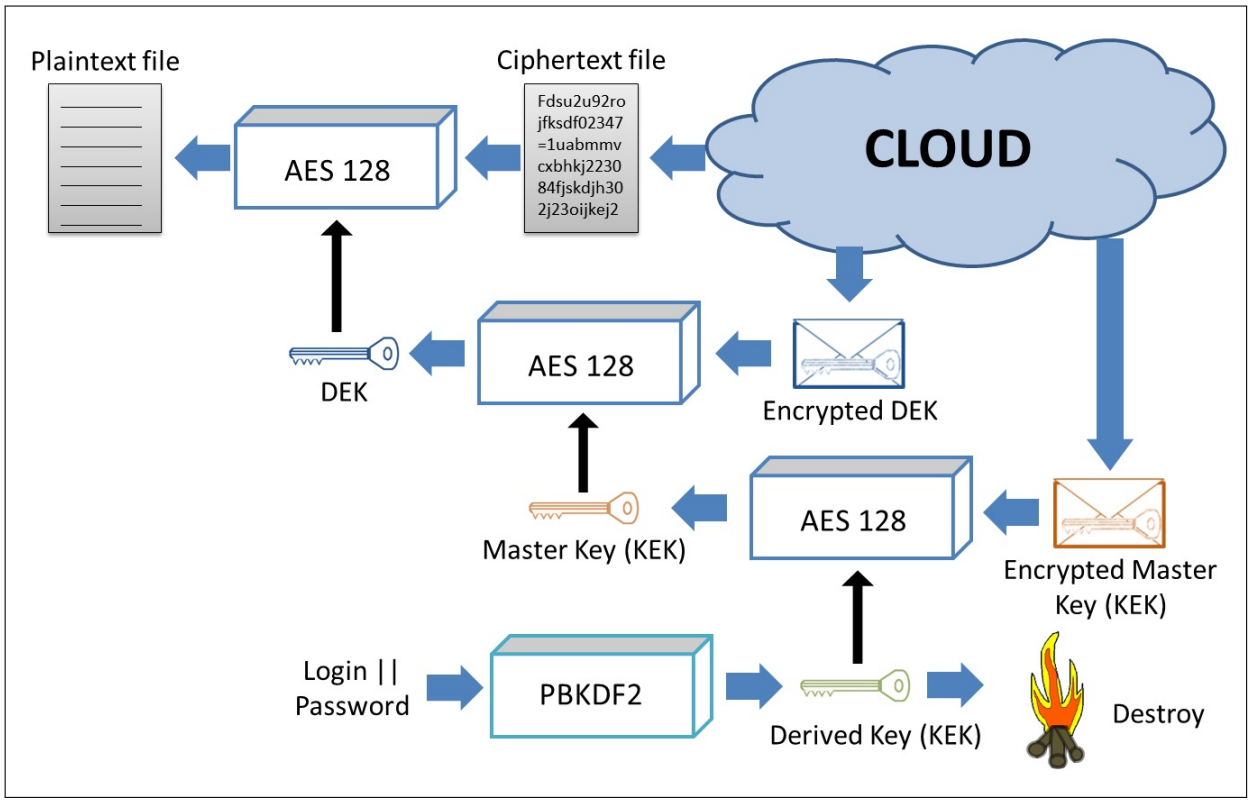


Figure B.31 – Decrypting files with Mega

APPENDIX C – CPG Use Cases

In this appendix we show the main use cases of the proposed CPG application.

C.1 Introduction

In order to understand the problems and challenges in building an application like CPG, we designed use cases covering the main scenarios during users' interactions. Before presenting the use cases, a few definitions must be made in order to understand the system components, as well as the actors.

C.1.1 Services provided

CPG offers two main services to users: data sharing and encryption/decryption of data content and attributes.

C.1.2 Requirements demanded by CPG

Users need to have the following items in order to use CPG:

- An account previously created and configured in any CSP which works with the concept of synchronization between the cloud and a folder in user's device.
- The client application of that CSP installed and configured in user's computer.
- A personal ICPEdu, ICP-Br or similar (X.509) digital certificate.

C.1.3 Actors and components

The actors involved in the use cases are:

- Alice: User who wants to store data securely in the cloud.
- Bob: User who needs a file shared by Alice.
- CPG: Proposed application.
- CSP: Cloud Service Provider chosen by Alice and Bob to store their data.

The folders used by CPG are:

- *Encrypted Files* folder: Folder where encrypted files are stored. As it is located inside the clod synchronization folder, its content is synchronized with the cloud by the cloud client application.
- *Secret Files* folder: Folder used to store files to be encrypted and sent to the *Encrypted Files* folder. The files remain in plain text format in this folder.

C.2 Basic events on CPG

In this section, we detail the main use cases presented in Table C.1. For each one, a description will be given, along with the basic flow.

Table C.1 – Events of CPG basic flow

Use case ID	Use case Name	Actor
1	Authentication	Alice
2	CPG Configuration settings	Alice
3	Storing encrypted files	Alice
4	Recovering encrypted files from the cloud	Alice
5	Sharing files	Alice
6	Erasing files	Alice/CPG
7	Closing CPG	Alice
8	Starting CPG	CPG
9	Creating the cryptographic keys	CPG
10	Validating users' passphrase	CPG
11	Encrypting files	CPG
12	Decrypting files	CPG

C.2.0.1 Authentication

Use case ID: 1
Actor: Alice.
Description: Alice must authenticate providing her passphrase, in order to access CPG and perform any activity involving her private key.
Basic flow:

1. Alice starts CPG or starts any operation involving her private key.
2. CPG shows a window for Alice to provide her passphrase (Fig. C.1).
3. Alice enters her passphrase.
4. CPG performs the validation process (Use Case 10 - Validating user's passphrase).

5. If the authentication succeeds, CPG starts the intended task; otherwise, an error message will be shown to Alice, requesting her passphrase again. This is done for a pre-established number of times when the application blocks access for a given period of time.

Observations: CPG will show in the Login window (Fig. C.1) the usersname got from the operating system. This is done in order to decrease users' interaction with CPG, avoiding they also have to supply an username every time they authenticate with the application. Also, the passphrase provided by Alice has a lifetime and after this time expires, Alice will be requested to supply it again in the next interaction with CPG.

Figure C.1 – User authentication window

C.2.0.2 CPG Configuration settings

Use case ID: 2
Actor: Alice.
Description: After Alice starts CPG for the first time, she will be requested to set the standard folders used by CPG. Alice also can change the folders at any time through the *"Settings"* option, which can be accessed by clicking with the right mouse button in CPG icon available in the system tray (Fig. C.2).
Basic flow:

1. CPG shows Alice a configuration window (Fig. C.3).
2. Alice needs to select folders for CPG usage, as demanded in the configuration window. The folders are:
   a) *Encrypted Files*: Folder where encrypted files are stored. It has to be inside the cloud synchronization folder, in order to have its content synchronized with the cloud.
   b) *Secret Files*: Folder used to receive data to be encrypted and sent to the cloud.
   c) *Cryptographic keys*: Folder used to store users' key pair (private and public). By default, this folder is created automatically by CPG when used for the first time.
3. Alice fills out each item. She also can click in the button beside the text box to find and select the folders.

4. To finish the process, Alice must click in the *"Save"* button.

5. CPG stores a configuration file in its own folder with these information provided by Alice.

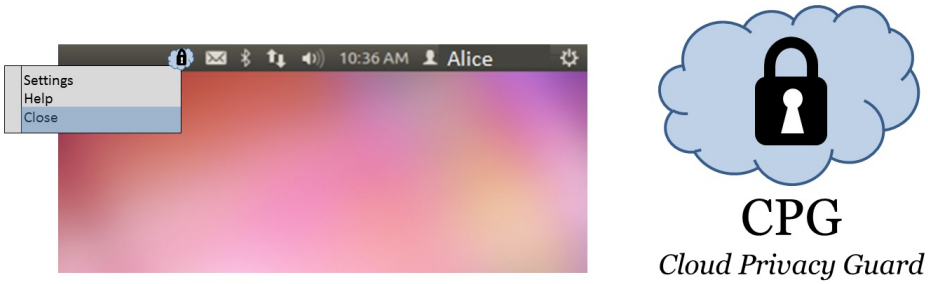


Figure C.2 – Activating CPG settings option

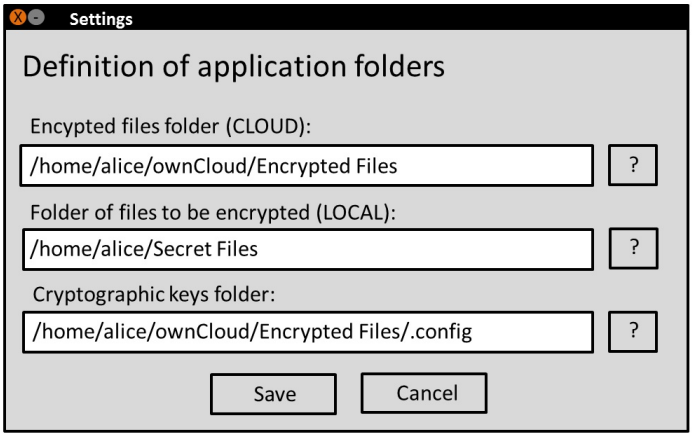


Figure C.3 – Settings

C.2.0.3 Storing encrypted files

Use case ID: 3
Actor: Alice.
Description: Alice wants to store one or more files in the cloud in encrypted form.
Basic flow:

1. Alice starts CPG and performs the authentication process (Use Case 1 - Authentication).

2. Alice stores the intended file(s) in the *Secret Files* folder (Figs. C.4 and C.5).

3. The file(s) is(are) encrypted (Use Case 11 - Encrypting Files) and stored in the *Encrypted Files* folder to be synchronized with the cloud servers (figs. C.6 and C.7).

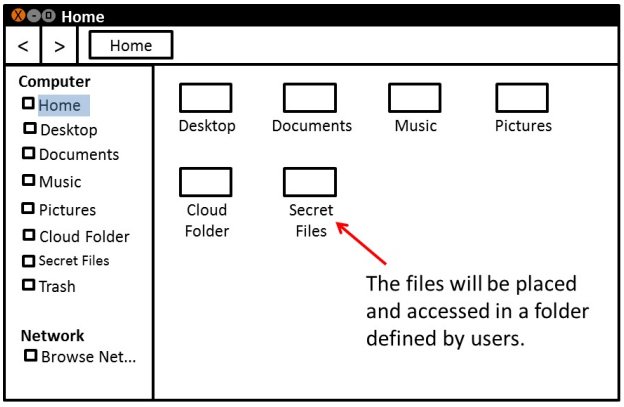


Figure C.4 – Folder of files to be encrypted

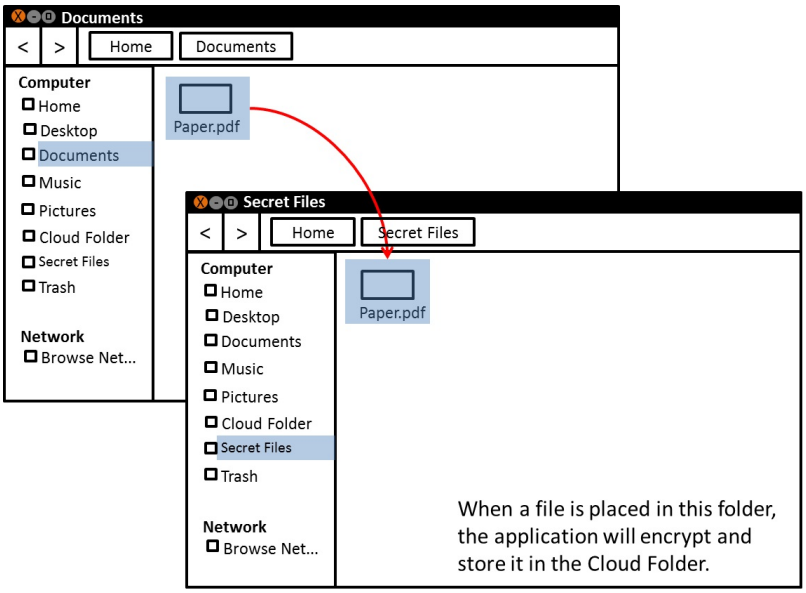


Figure C.5 – Sending a file to be encrypted and stored in the cloud

C.2.0.4 Recovering encrypted files from the cloud

Use case ID: 4
Actor: Alice.
Description: Alice wants to recover one or more files stored in the cloud in encrypted form.
Basic flow:

1. If the file was stored in the cloud by the current device being used by Alice, no additional steps are necessary. She can access her file in the *Secret Files* folder, where the plain text version of it is stored.

Alternative flow: If the file was stored in the cloud by Alice using a different device or she is trying to access a file shared with her by some user, the following steps are necessary:

1. Alice starts CPG and performs the authentication process (Use Case 1 - Authentication).

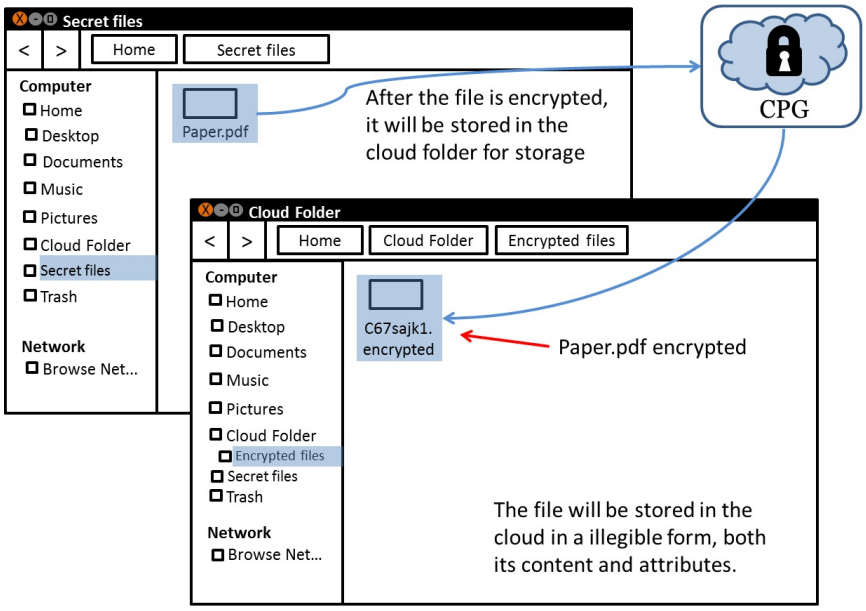


Figure C.6 – File encryption

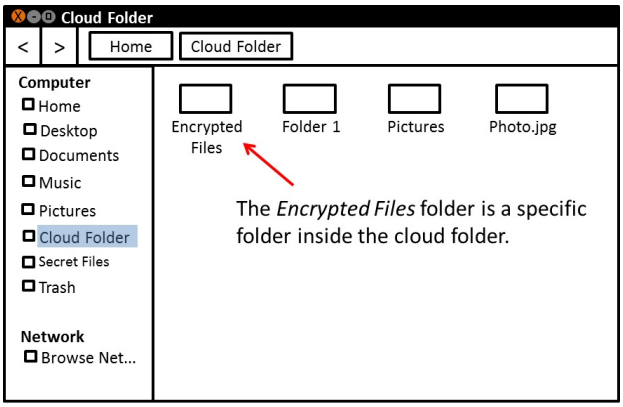


Figure C.7 – *Encrypted Files* folder

2. After authentication, CPG will decrypt every file in the cloud whose plain version is not stored in Alice's current device.

3. Alice can access her file (or the shared file) in the *Secret Files* folder.

C.2.0.5 Sharing files

Use case ID: 5
Actor: Alice.
Description: Alice wants to share a file with Bob.
Basic flow:

1. Bob expresses desire to get/access Alice's file (by email, for example).

2. Bob sends his certificate to Alice (by email, for example).

3. Alice chooses or creates in *Secret Files* a folder for sharing data with Bob. She accesses her CSP's interface, creates and shares a similar folder with Bob using the means offered by her provider.

4. Alice starts CPG and performs the authentication process (Use Case 1 - Authentication).

5. Alice puts Bob's certificate into the folder selected previously (Fig. C.8).

6. CPG will encrypt every file in this folder using Alice and Bob's certificates (Use Case 11 - Encrypting Files) and stores them in *Encrypted Files* folder.

   a) CPG encrypts only the DEK (Data Encryption Key) with Bob's public key.

   b) The new encrypted DEK is attached in the file container.

   c) For separating the DEKs as there will be one for each user with access to the file, it could be used a XML file with users' name and the respective encrypted keys (Fig. C.10 case 3).

7. Bob starts CPG and performs the authentication process (Use Case 1 - Authentication).

8. Bob's CPG decrypts the new files and stores them in the folder shared between him and Alice (Use Case 12 - Decrypting Files).

9. Bob has access to the file in his *Secret Files* folder.

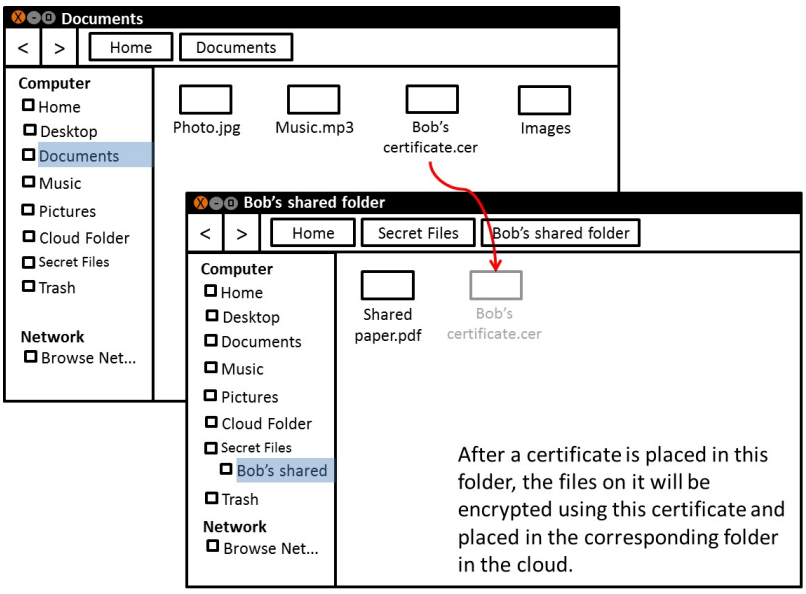


Figure C.8 – Sharing files

C.2.0.6 Erasing files

Use case ID: 6
Actor: Alice / CPG.
Description: After Alice selects one or more files in the *Secret Files* folder and deletes them (in the same way she deletes files in her operating system), CPG will delete these files from the cloud too.
Basic flow:

1. Alice starts CPG and performs the authentication process (Use Case 1 - Authentication).
2. Alice delete the file(s) of the *Secret Files* folder.
3. CPG deletes the encrypted version of it(them) from the *Encrypted Files* folder, which is reflected in the cloud servers when a synchronization is performed.

Observations: CPG does not guarantee that the files will be deleted from the cloud servers. It only deletes them from users' accounts in the same way used by users to delete them from the cloud interface (cloud client). Also, users' files that are supposed to be deleted could stay in the cloud servers for a long period of time, due to the deduplication process or for allow users restore them later in case of an accidental deletion. However, with CPG these files will be encrypted and the CSPs will not have access to their content neither their attributes, preserving users' privacy.

C.2.0.7 Closing CPG

Use case ID: 7
Actor: Alice.
Description: CPG will be closed after Alice selects the *"Close"* option.
Basic flow:

1. Alice clicks with the right mouse button in CPG's icon available on the system tray (beside the system clock) and selects the *"Close"* option.
2. CPG shows a confirmation window for Alice clicking in *"Yes"*, in case she wants to proceed, or *"Cancel"*, in case she does not want to close the application any more (Fig. C.9).
3. In case Alice selects the *Yes* option, CPG releases all open files, clean variables and temporary files. After that, it ends. Otherwise, CPG keeps running.

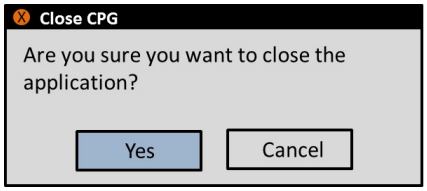


Figure C.9 – Confirmation message

C.2.0.8 Starting CPG

Use case ID: 8
Actor: CPG.
Description: This process occurs every time Alice starts CPG.
Basic flow:

1. Alice starts CPG.
2. CPG searches for Alice's username in the operating system.
3. If the configuration file do not exist or are not found, CPG requests their configuration (Use Case 2 - CPG Configuration settings).
4. CPG searches for Alice's cryptographic keys. If the keys do not exist or are not found, CPG requests their creation (Use Case 9 - Creating the cryptographic keys). Otherwise, CPG requests Alice's passphrase to authenticate her (Use Case 10 - Validating users' passphrase).
5. CPG keeps running in background.

C.2.0.9 Creating the cryptographic keys

Use case ID: 9
Actor: CPG.
Description: CPG creates user's cryptographic keys (public and private).
Basic flow:

1. When CPG is started, it checks for the user's cryptographic key pair in the *Cryptographic keys* folder.
2. If the keys were not found, CPG requests Alice to select an existent key pair or create a new one. If she does not have one and wants to proceed, CPG generates a public and private key for Alice and stores them in separated files; otherwise, she needs to select the key pair.

3. CPG requests Alice to create a passphrase which will be necessary every time she uses the application or performs a cryptographic operating using her private key. This passphrase will be stored nowhere and can not be recovered in case it gets lost.

4. CPG uses Alice's passphrase to derive a KEK (Key Encryption Key) and encrypts her private key. The public key is stored in a digital certificate format. Both keys are stored locally, in *Cryptographic keys* folder, and in the cloud, in a hidden folder (.config) inside *Encrypted Files*.

5. CPG keeps running in background.

C.2.0.10 Validating users' passphrase

Use case ID: 10
Actor: CPG.
Description: CPG verifies if the passphrase provided by Alice is the right one.
Basic flow:

1. CPG shows a window requesting Alice's passphrase (Fig. C.1).

2. Alice enters her passphrase.

3. CPG decrypts Alice's private key with the provided passphrase.

4. To perform the verification, CPG encrypts a random text $Text_0$ with Alice's public key and decrypts it with the private key got from the previous step, resulting in $Text_1$. If text $Text_0$ is the same as $Text_1$, the passphrase is correct; otherwise, an error message is sent to Alice asking her to try again.

Observations: Users' passphrase will be only valid for a period of time. After that, it has to be provided again in order to perform any operation that request users' private key.

C.2.0.11 Encrypting files

Use case ID: 11
Actor: CPG.
Description: CPG will encrypt the file(s) in the *Secret Files* folder and store it (them) in the *Encrypted Files* folder to be sent and stored in the cloud in a secure way.
Basic flow:

1. Alice stores the file(s) desired to be sent in encrypted form in *Secret Files* folder.

2. The(se) file(s) will be encrypted and stored in the *Encrypted Files* folder to be synchronized with the cloud server.

a) The file is compressed and placed in a temporary folder.

b) CPG generates a unique, random and symmetric key (DEK - Data Encryption Key) for this file.

c) The file is encrypted with this key using AES algorithm. Then, the DEK is encrypted with Alice's public key and the result stored in a file. If there is any certificate belonging to another user inside the folder where the file was stored, this DEK is also encrypted using the public key of this user.

d) The filename is also encrypted with the DEK. The result is stored in a file in a base-32 form, and a fixed-size part of it is used to rename the new encrypted file. The base-32 form was chosen to avoid the special characters slash (\) and backslash (/) in the name of files, since they are not permitted in operating systems. Other attributes as last modification date, size, creation data etc., are hidden in the compression process, and can not be accessed after the file is encrypted, since new attributes will be defined for it. Only after the is file decrypted, the original attributes can be restored.

e) At this point, there will be at least three files created by this process: the file, DEK(s) (there will be a DEK file for each user that has access to the file) and the filename file. All these files are encrypted and renamed with the same name, except in the extension.

f) There are at least three different ways to store all these files, which are:

   i. Creates a new and single object and store each file content in a pre-defined layout (Fig. C.10 case 1).

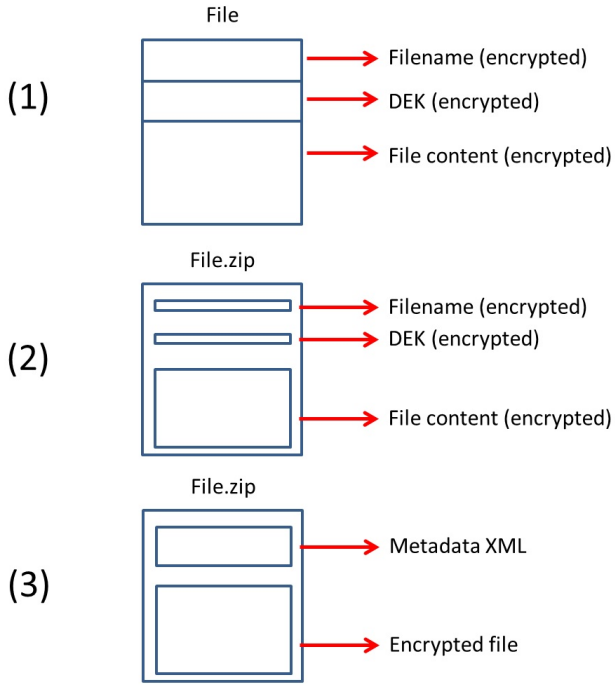
   ii. Creates a container and store the files inside of it as distinct files (Fig. C.10 case 2).

   iii. Creates a metadata file to store the encrypted key and filename content. Then, the metadata and content files are stored in a container as distinct files (Fig. C.10 case 3).

Observation: In CPG' design, we choose the second way.

C.2.0.12 Decrypting files

Use case ID: 12
Actor: CPG.
Description: CPG will decrypt the files in the *Encrypted Files* folder which do not have a plain version in the *Secret Files* folder and stores them in the latter one.
Basic flow:



METADATA

```
<file>
    <users>
        <Alice id=1>
            <filekey> dashuf79qhk1çojijkncds</filekey>
        </Alice>
        <Bob id=2>
            <filekey> GBxh78kl02sjdgtykdodp</filekey>
        </Bob>
    <users>
    <filename> ahfdhiuds9018274kkaskjhdsas </filename>
</file>
```

Figure C.10 – Encrypted file structure

1. CPG will decrypt every file in *Encrypted Files* folder which do not have a plain text version in the *Secret Files* folder. This process occurs when users send files to the cloud from one device and access them from another, or when files are shared among users. In such cases, the encrypted files are downloaded and then decrypted by CPG.

2. The(se) file(s) will be decrypted and stored in the *Secret Files* folder where it (them) can be accessed.

    a) Before decrypting the file, CPG separates it considering the structure adopted in the encryption process (Fig. C.10). It will be considered the second case since it is the one adopted by CPG on its first implementation. After separating the files, there will be at least three files: file content, DEK and filename. These files are placed in a temporary folder.

    b) CPG first uses Alice's passphrase to decrypt her private key.

    c) Then, the DEK is decrypted with Alice's private key and used to decrypt the content and filename files.

    d) A new file is created with the name decrypted and content. The DEK is discarded.

e) The final step is related to decompress the file. All files original attributes are restored in this step.

3. Alice accesses her file.

Observations: Usually, Alice can access her files without decrypting it(them), once they are stored in plain text format in the device. The decryption is only needed when files are sent to the cloud from a different device or when users share files with her. In such cases, the files will be downloaded in the device and then decrypted by CPG.

APPENDIX D – Publications derived from this work

- Vitor H. G. Moia; Marco A. A. Henriques. Relação custo/benefício de técnicas utilizadas para prover privacidade em computação nas nuvens. In *XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais SBSeg 2014*, pages 322-325, Belo Horizonte, MG, Brazil, Nov 2014.
- Vitor H. G. Moia; Marco A. A. Henriques. Cloud privacy guard (CPG): Security and privacy on data storage in public clouds. In *VIII Congresso Iberoamericano de Seguridad Informática CIBSI 2015*, pages 88-95, Quito, Ecuador, Nov 2015.
- Vitor H. G. Moia; Marco A. A. Henriques. Security requirements for data storage services on public clouds. In *XXXIII Simpósio Brasileiro de Telecomunicações - SBrT 2015*, Juiz de Fora, MG, Brazil, Set 2015.
- Vitor H. G. Moia; Marco A. A. Henriques. Uma forma de tratar o desafio de proteger a privacidade dos usuários de armazenamento de dados em nuvens. In *XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais SBSeg 2015*, pages 644-655, Florianópolis, SC, Brazil, Nov 2015.