

Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Cryptographic Storage Service on a Cloud System

Candidato:
Giacomo Veronelli

Relatore:
Ing. Carlo Ferrari

Anno Accademico 2010-2011

*Ai miei genitori
che mi hanno sempre sostenuto.*

Contents

1	Summary	1
2	Cloud Computing	2
2.1	Understanding Cloud Computing,	2
2.2	Cloud Service Models	2
2.3	Cloud Deployment Models	5
2.4	Control over security	6
2.5	Public Cloud	7
2.6	Representative Commercial Cloud Architectures	8
3	Securing the Cloud	11
3.1	Security Concerns	11
3.2	Data Privacy and Security	14
3.3	Organizational Responsibility: Ownership and Custodianship	15
3.4	Data at Rest	17
3.5	Data in Motion	18
3.6	Cryptography on Cloud	19
3.6.1	Data Encryption: Applications and Limits	19
3.6.2	Impediments to Encryption in the Cloud	19
3.6.3	State of the Art	19
4	Project	21
4.1	Problem definition	21
4.2	Proposal	22
4.3	Actors	23
4.3.1	Service Company	23
4.3.2	Public Cloud System Provider	23
4.3.3	KeyServer	23
4.3.4	Users/Owner	24
4.4	Global Functionality	24
4.4.1	Data Creation	24
4.4.2	Data reading	26
4.4.3	Data Updating	27
4.4.4	Data Deletion	27

4.5	Different scenario	27
4.5.1	User unique owner of data	27
4.5.2	Company owner of data	27
4.5.3	User and Partners	27
4.5.4	Company and Users groups	29
4.6	Advanced Features	30
4.6.1	Encrypted OwnerKey with MasterKey	30
4.6.2	Checksum Store into KeyServer	30
5	Developing	31
5.1	Web-application side	31
5.1.1	Interface Overview	31
5.1.2	Add new Data	33
5.1.3	Read my data	38
5.1.4	Add new File	39
5.1.5	Share my Data	41
5.1.6	LAMP platform	43
5.1.7	Google Appengine	44
5.1.8	Client browser side	46
5.1.9	Server side on VPS solution	46
5.1.10	CMS with MVC	47
5.2	Datakey service	48
5.2.1	Overview	48
5.3	Encryption	50
5.3.1	Browser-Based Cryptography Tools	50
5.4	Conclusion	51

List of Figures

2.1	The SPI model: software, platform, and infrastructure as a service.	3
2.2	The SPI model: relating services to infrastructure.	4
2.3	NIST Cloud Computing Model	6
2.4	Extent of control over security in SaaS, PaaS, and IaaS.	7
2.5	commercial cloud architectures timeline	9
2.6	Cloud Marketing Penetration	9
2.7	Amazon web services, Force.com. Google app engine comparison	10
3.1	Biggest cloud concerns	13
3.2	Meeting security needs: public, community, and private clouds. .	16
3.3	Owning organization has increasing control and responsibility over data	17
4.1	Representation of encryption	22
4.2	Data Creation	25
4.3	Data Reading	26
4.4	User send the data to the Partners	28
4.5	User create a copy of Data	28
4.6	User add Partner into Data Owner	29
4.7	User add Partner into Data Owner	29
5.1	Application Login Screenshot	32
5.2	Application Registration Screenshot	32
5.3	Functionality Menu	33
5.4	Session countdown	33
5.5	KeyServer Access Admin Panel	34
5.6	Insert of OwnerKey	34
5.7	GetDataKey Javascript Code	35
5.8	Get DataKey Procedure	35
5.9	Get DataKey Request Status	36
5.10	Data insertion	36
5.11	Encrypt button	37
5.12	Cipher Text	37
5.13	My data panel	37

5.14	Read data panel	38
5.15	Read data panel	39
5.16	Drop File	40
5.17	Base64 String	40
5.18	Encrypted file	41
5.19	Data sharing panel	42
5.20	Registered User list	42
5.21	User create a copy of Data	43
5.22	Lamp Solution	44
5.23	Google App Engine and GWT	45
5.24	Sql Creation code	47
5.25	KeyServer Admin Panel	49
5.26	Sql Creation code	50

Chapter 1

Summary

In this work, we propose a secure protocol to storage data into a cloud system application. The approach we followed allows the potential of cloud computing for applications and systems, all this while maintaining full control over the properties and content of data. We developed an application to prove the efficiency of the protocol in many different scenarios on various cloud platforms like Google Appengine and on a hosting virtual server. The application that we made can store simple text message or complex data such as file into a cloud system and share it with other users or groups of users. Moreover, its applicability to store data in safety has been tested and discussed and every single function has been discussed.

Chapter 2

Cloud Computing

2.1 Understanding Cloud Computing,

Cloud computing is an innovation of prior computing approaches, which builds upon existing and new technologies. Even as cloud presents new opportunities around shared resources, the relative news of the model makes it difficult to separate reasonable claims from hype. In part, excessive marketing claims have led to completely unrealistic perspectives of cloud security. Claims that cloud computing is inherently insecure are as absurd as are claims that cloud computing brings no new security concerns. Prospective cloud users can sense that there is value here, but their understanding of the problem is often incomplete. Cloud computing represents a paradigm shift for delivering resources and services; this results in important benefits for both cloud providers and cloud consumers. From how we build Information Technology (IT) systems and how we use them to how we organize and structure IT resources, cloud is refactoring the IT landscape. Instead of uncrating computers and racking them in your server closet, the cloud allows for virtually downloading hardware and associated infrastructure. By abstracting IT infrastructure and services to be relatively transparent, the act of building a virtual data center is now possible in minutes, with minimal technical background and at a fraction of the cost of buying a single server.

2.2 Cloud Service Models

The three service models for cloud computing are SaaS, PaaS, and IaaS.

As Mell and Grance define them[?]:

- Cloud Software-as-a-Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a client interface such as a Web browser (e.g., Web-based e-mail). The

consumer doesn't manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

- Cloud Platform-as-a-Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer doesn't manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.
- Cloud Infrastructure-as-a-Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer doesn't manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

We refer to these three as the SPI model. What we are really describing are three broad classes of capabilities that reside on top of physical cloud infrastructure, as depicted in These can be layered IaaS as

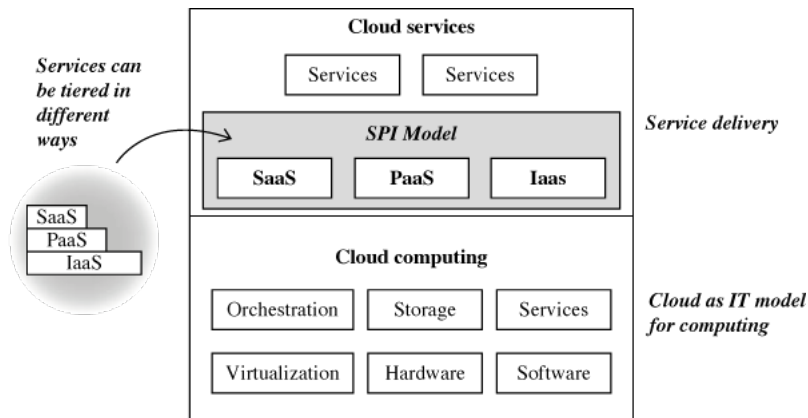


Figure 2.1: The SPI model: software, platform, and infrastructure as a service.

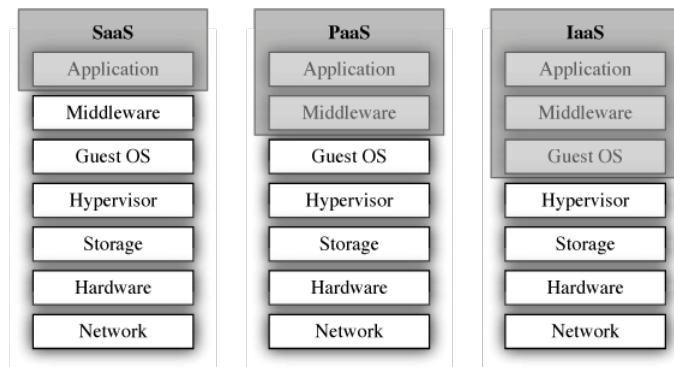


Figure 2.2: The SPI model: relating services to infrastructure.

a foundation for PaaS, and PaaS as a foundation for SaaS or they can stand alone. How services are implemented will depend on the provider. The case can be made that “IaaS and PaaS are special purpose versions of SaaS that enable new cloud services.” The Cloud Security Alliance has taken the following view: IaaS is the foundation of all cloud services, with PaaS building upon IaaS, and SaaS in turn building upon PaaS. . . . In this way, just as capabilities are inherited, so are information security issues and risk. It is important to note that commercial cloud providers may not readily fit into the layered service models. However, the reference model is important for relating real-world services to an architectural framework and understanding the resources and services requiring security analysis. Their position is well taken as it certainly would be more agile for a cloud provider to express SaaS as a service of PaaS, and PaaS as a service of IaaS. However, most cloud providers don’t implement services delivery in that way. The point is that infrastructure, platform, and software are three forms of cloud service delivery and that they can be delivered independently or as layered services. But these services classes are also quite similar; each offers a container with specific interfaces, capabilities, and limitations. Some of the containers provide interfaces that act like a whole operating system, and some are so application specific that they can’t be generically programmed. These definitions are really just examples of interesting points on a continuum of offered services. Beyond SaaS, PaaS, and IaaS, several other service delivery models have been proposed, these include Data center-as-a-Service, Security-as-a-Service, Monitoring as- a-Service, and Identity-as-a-Service, but these should be seen as specialized cases of the SPI model. While many new and innovative products and services have been enabled because of the cloud model, many marketing organizations have had a field day in representing anything as a service. But the increase in fine-grained as-a-service definitions is evidence that the SPI model is not necessarily universal and that we are rapidly evolving toward more useful definitions of overall cloud services models.

2.3 Cloud Deployment Models

Mell and Grance next define the four Cloud Deployment models:

- Private cloud. The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.
- Community cloud. The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.
- Public cloud. The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
- Hybrid cloud. The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

These four deployment models can see significant variation depending on other factors that we will discuss in the next section, but they serve to address the broad questions as to how one can deploy pooled cloud resources. Before we move on, it is important to make two points about the NIST Cloud Model:

- A customer or tenant can have greater security control over more resources as one moves from SaaS to PaaS and again from PaaS to the IaaS service model.
- A customer or tenant can achieve greater security control over more resources when moving from a Public cloud to a community cloud and again from a community cloud to a Private cloud.

Figure 2,3 is an adaption of the NIST Cloud Computing Model, which has been annotated to reflect the discussion in this section on customer and tenant control. We will examine the issue of control in greater detail in the next section.

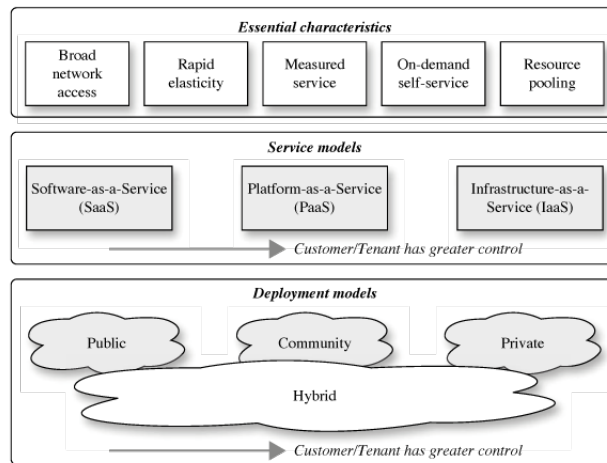


Figure 2.3: NIST Cloud Computing Model

2.4 Control over security

In part, the SPI service model represents increasing abstraction from complex underlying IT infrastructure. As represented in Figure 2.4, cloud-based IaaS doesn't typically expose actual hardware or networking layers to the tenant of the service, rather these underlying resources are abstracted for the consumer. PaaS abstracts infrastructure to a greater extent and generally presents middleware containers that are made for categories of usage such as development.

These containers provide tools to simplify application development and limit application interactions with the underlying systems. SaaS abstracts even further and generally exposes narrow-functionality software-based services such as Customer Relationship Management (CRM) or e-mail. At every step up the SPI continuum, there are increasing limitations on lower-level computing functions. In other words, from IaaS to SaaS underlying computing functions are more and more abstracted. With SaaS, the burden of security lies with the cloud provider. In part, this is because of the degree of abstraction, but the SaaS model is based on a high degree of integrated functionality with minimal customer control or extensibility. The PaaS model, instead, offers greater extensibility and greater customer control but fewer higher-level features. Largely because of the relatively lower degree of abstraction, IaaS offers greater tenant or customer control over security than do PaaS or SaaS. Another way to consider this is that with SaaS the provider is responsible for most aspects of security, compliance, and liability, but with these responsibilities, the provider is righter to change important aspects of the service or associated service contracts (that is SLAs). Given this discussion of service models and security, we should consider how cloud deployment impacts the degree of owner data/application control over security. Clearly, the degree of control that a tenant or customer has in a public cloud is

minimal, whereas the tenant organization has maximum control with a private cloud. The degree of control will vary for community and hybrid clouds and may not be relevant depending on what such external computing resources are used for. But the public and private deployment vector is not the only aspect of this discussion. For a private cloud especially, we should also consider where the cloud infrastructure resides and who operates it.

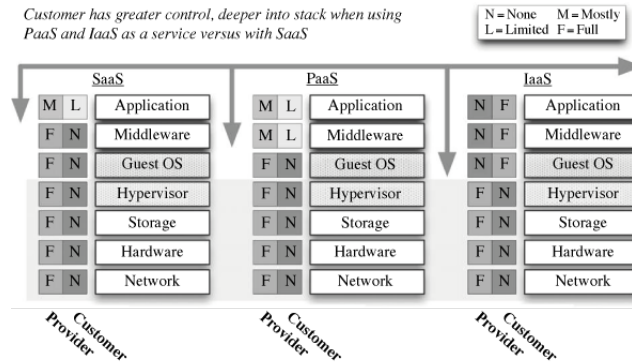


Figure 2.4: Extent of control over security in SaaS, PaaS, and IaaS.

But this is not necessarily true as a private cloud can benefit greatly from the physical security that a hosting facility can offer. Likewise, outsourcing operations can be just as secure and potentially less expensive than having ample 24×7 IT personnel on staff. This last point is especially true for security monitoring services as there is true benefit to using a security monitoring staff that sees more incidents and issues than a single cloud may present. Knowledge does scale, and it can be expensive to develop. When considering how to secure public versus private cloud architectures, the security concerns are more different than common. If a cloud is private, internal on a customer premises, and owned/-managed/maintained exclusively by the organization utilizing it, the principles in securing it vary greatly from those of a public cloud hosted externally by a third party. A private cloud doesn't have the data confidentiality and legality concerns that a public cloud might.

2.5 Public Cloud

To define it in the simplest way, a public cloud exists externally to its end user and is generally available with little restriction as to who may pay to use it. As a result, the most common forms of public clouds are ones that are accessed by Internet. There has been terrible development in the public cloud space, resulting in very sophisticated Infrastructure-as-a-Service offerings from companies like Amazon, with their Elastic Compute Cloud (EC2), Rackspace's Cloud Offerings, and IBM's BlueCloud. Other forms of public cloud offerings

can take the form at more of the application layer, or Platform-as-a-Service, like Google's AppEngine and Windows' Azure Services platform, as well as Amazon's service-specific cloud hosting SimpleDB, Cloud Front, and S3 Simple Storage. At a basic level, public clouds have unique security components and evaluation criteria when compared with private clouds. Public clouds can be formed by service providers wishing to build out a high-capacity infrastructure and lease pieces of it to a variety of clients. As a result, data might become comingled on common storage devices, making identity, access control, and encryption very important. There is a certain amount of inherent trust (even if it should be a measured, tested, and verified) by subscribers with their public cloud providers.

2.6 Representative Commercial Cloud Architectures

Although the concept of a cloud has been around for decades, in reality, cloud computing in the forms, we know today, are relatively new. For example, below are the dates at which the various types of public and private clouds, SaaS, PaaS, IaaS providers, and technologies associated with them have been in existence.

- Amazon Web Services (Public Cloud, IaaS) Arguably one of the most mature clouds, launched in July 2002 not really with an IaaS offering, more just pieces of it. It's EC2, or Elastic Compute Cloud, which is classified as an IaaS offering launched officially (non-beta) in October 2008. Many new components of this cloud are still being launched today see Amazon VPC.
- Amazon Virtual Private Cloud or VPC (Hybrid Cloud Technology) Marries, an Amazon public cloud with an enterprise's private cloud, is still in beta at the time of publishing in 2010.
- Rackspace Cloud Hosting (Public Cloud, IaaS) Launched publicly in February 2008.
- GoGrid (Public Cloud, IaaS) Launched in April 2008.
- Salesforce.com (Public Cloud, SaaS, and PaaS) Although the company was launched March 1999, Salesforce's PaaS, Force. com was launched in January 2008.
- Google Apps Engine (Public Cloud, PaaS) Its first public beta was launched in April 2008. GovCloud, Google's form of Google Apps that addresses and meets government security mandates was only launched in September 2009.
- VMware (Private Cloud Technology Provider) Although the company was officially founded in 1998, VMware Server didn't exist publicly until 2001.
- Microsoft Hyper-V (Private Cloud Technology Provider) Virtualization technology created by Microsoft and deployed in Windows Server 2008, officially launched in June, 2008.

- Force.com is a cloud computing platform as a service system from Salesforce.com,[1] that developers use to build multi tenant applications hosted on their servers as a service.
- Heroku is a cloud Platform as a Service (PaaS) run by the San Francisco, California-based company with the same name. Heroku led the way for a multi-language PaaS, introducing the 'polyglot platform'. Heroku initially supported the Ruby programming language, with Rack and Ruby on Rails.

It would be fair to summarize that most modern cloud computing architectures, in their form as they exist today, are generally around 3 years old. This is a far cry from the maturation of modern architectures or common security standards.

In the figure 2,5 we can see the evolution on commercial cloud architectures.

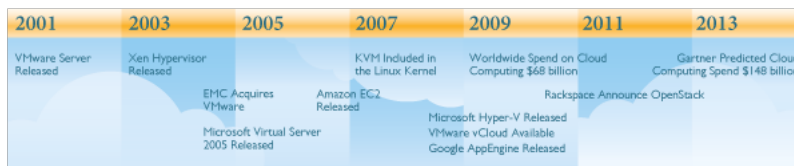


Figure 2.5: commercial cloud architectures timeline

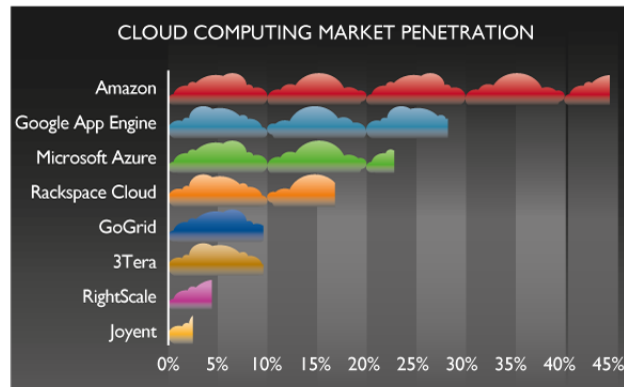


Figure 2.6: Cloud Marketing Penetration

In the follow Figure 2.7 we can compare some modern best solutions on the public cloud architectures like Google Appengine, Amazon and Force.

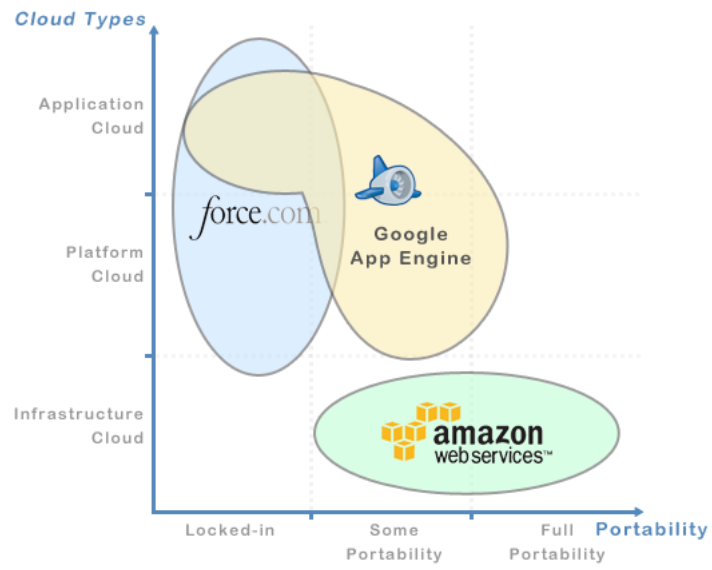


Figure 2.7: Amazon web services, Force.com. Google app engine comparison

Chapter 3

Securing the Cloud

In Chapters 1 and 2, we covered many of the qualities and promises of cloud computing. In addition, we examined the three models for cloud services (SPI) and the four models for cloud deployment (public, private, community and hybrid). While developing a background in cloud computing, we also discussed many security aspects of clouds. In this chapter, we are going to investigate some of those security issues more closely. While some might find the cloud inappropriate from a security standpoint, we will attempt to show that this amounts to a wrong conclusion. As we stated frequently, by its inherent qualities, cloud computing has tremendous potential for organizations to improve their overall information security posture. There are many reasons for this, but the best way to sum up the argument is to state that the cloud model enables the return of effective control and professional operation over Information Technology (IT) resources, processing, and information. By virtue of public cloud scale, tenants and users can get better security since the provider's investment in achieving better security costs less per consumer. For the same reasons, a private cloud can obtain significant advantages for security. But there are wrinkles: You won't get the benefit without investment, and not every model is appropriate for all consumers. But, regardless of which services delivery model or deployment model you select, you will transfer some degree of control to the cloud provider which would be completely reasonable if control is managed in a manner and at a cost that meets your needs.

3.1 Security Concerns

To begin with, we will list some common security concerns:

- **Network Availability** The value of cloud computing can only be realized when your network connectivity and bandwidth meet your minimum needs: The cloud must be available whenever you need it. If it is not, then the consequences are no different than a denial-of-service situation.

- **Cloud Provider Viability** Since cloud providers are relatively new to the business, there are questions about provider viability and commitment. This concern deepens when a provider requires tenants to use proprietary interfaces, thus leading to tenant lock-in.
- **Disaster Recovery and Business Continuity** Tenants and users require confidence that their operations and services will continue if the cloud provider's production environment is subject to a disaster.
- **Security Incidents** Tenants and users need to be appropriately informed by the provider when an incident occurs. Tenants or users may require provider support to respond to audit or assessment findings. Also, a provider may not offer sufficient support to tenants or users for resolving investigations.
- **Transparency** When a cloud provider does not expose details of their internal policy or technology implementation, tenants or users must trust the cloud provider's security claims. Even so, tenants and users require some transparency by providers as to provider cloud security, privacy, and how incidents are managed.
- **Loss of Physical Control** Since tenants and users lose physical control over their data and applications, this results in a range of concerns:
 - **Privacy and Data** With public or community clouds, data may not remain in the same system, raising multiple legal concerns.
 - **Control over Data User** or organization data may be commingled in various ways with data belonging to others.
 - A tenant administrator has limited control scope and accountability within a Public infrastructure-as-a-service (IaaS) implementation, and even less with a platform-as-a-service (PaaS) one. Tenants need confidence that the provider will offer appropriate control, while recognizing that tenants will simply need to adapt their expectations for how much control is reasonable within these models.
- **New Risks, New Vulnerabilities** There is some concern that cloud computing brings new classes of risks and vulnerabilities. Although we can postulate various hypothetical new risks, actual exploits will largely be a function of a provider's implementation. Although all software, hardware, and networking equipment are subject to unearthing of new vulnerabilities, by applying layered security and well-conceived operational processes, a cloud may be protected from common types of attack even if some of its components are inherently vulnerable.
- **Legal and Regulatory Compliance** It may be difficult or unrealistic to utilize public clouds if the data you need to process is subject to legal restrictions or regulatory compliance. While we should expect providers to

build and certify cloud to address the needs of regulated markets, achieving certifications may be challenging due to the many nontechnical factors including the current stage of general cloud knowledge. As best practices for cloud computing encompass greater scope, this concern should largely become a historical one. The second half of this chapter is devoted to legal and regulatory issues.

Although the public cloud model is appropriate for many nonsensitive needs, the fact is that moving sensitive information into any cloud that is not certified for such processing introduces inappropriate risk. Let's be completely clear:

- It is at best unwise to use a public cloud for processing sensitive, mission critical, or proprietary data.
- It is expensive and excessive to burden nonsensitive and low-impact systems with high assurance security.
- It is irresponsible to either dismiss cloud computing as being inherently insecure or claim it to be more secure than alternatives.
- Selection of a cloud deployment model along with ensuring that you have appropriate security controls should follow a reasonable assessment of risks.

In the follow figure we can see how is the percent of security into the actual cloud scenario.

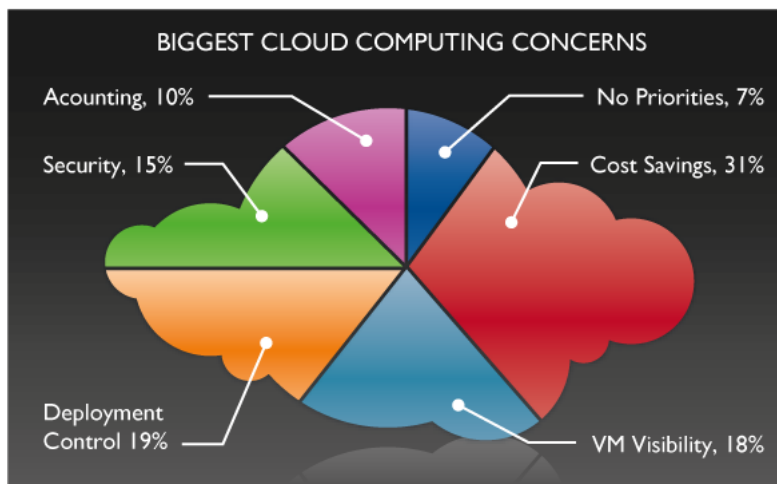


Figure 3.1: Biggest cloud concerns

3.2 Data Privacy and Security

The issue of data privacy is very much to the forefront of everybody's mind, with many television commercials advertising products and news programs describing another data breach. Any organization has a legal obligation to ensure that the privacy of their employees and clients is protected. Laws prohibit some of this data to be used for secondary purposes other than for what it was collected. You cannot surreptitiously collect data on say, the health of your employees, and then use this to charge smokers with higher insurance premiums. In addition, you cannot share this data with third parties. In the world of cloud computing, this becomes much harder as you now have a third party operating and managing your infrastructure, and hence by inference will have access to your data. If your organization is collecting and storing data in the cloud and this is subject to the legal requirements of one or more regulations (for instance, HIPAA or GLBA), then you must ensure that the cloud provider protects the privacy of the data in the appropriate manner. In the same way as data collected within your organization, data collected in the cloud must only be used for the purpose that it was collected for. If the individual specified that the data collected be used for one purpose, then that must be upheld. Often, privacy notices specify that individuals can have access to their data and to have this data deleted or modified. If this data is in a cloud provider's environment, privacy requirements still apply and the enterprise must ensure that this is allowed within a similar timeframe as if the data were held within a traditional IT implementation. If this can only be accomplished by personnel in the cloud provider's enterprise, you must be satisfied that they can undertake the task as you need. If you have entered into a click-wrap contract, you will be constrained to what the cloud provider has set out in these terms. Even with a tailored contract, the cloud provider may try to limit the control over your data to ensure that all its clients have a unified approach, hence reducing their overhead and the need to have specialist staff on hand. If complete control over your data is a necessity, then you need to ensure upfront that this can be accomplished and not try to bend to the cloud provider's terms. There are a number of cloud provider companies that specialize in distinct markets and tailor their services to those markets. This is likely to become more prevalent in the upcoming years and there will also likely be niche cloud providers. For instance, cloud providers that offer services in the health care marketplace would be bound by the relevant regulations for that market (HIPAA in this case) and we would expect them to charge for the special handling and controls that are needed.

Depending on the type of cloud provider you contract to, you will have to consider if your data is going to be mined by the supplier or others. The use of your data may occur unbeknownst to you or by virtue of a configuration error on the provider's part. Based on the sensitivity of your data, you may wish to ensure that your contract prohibits or at least limits the access the cloud provider has to use this data. This may be especially hard when you enter into a click-wrap agreement and as we all know, very few of us will read the fine print at all and just click the I agree box when it appears. In 2009, when Facebook

(www.facebook.com) changed its terms around security of data, many people complained, but the majority of users carried on using the service because they found it useful. It is likely that your users will react in the same way, which may well give you security issues. The data you are storing in the cloud may be confidential or hold personal data which you want to ensure is secure. The cloud provider is likely to have full access to this data to maintain and manage the servers for you. You will need to ensure that this access is not abused in any way. Although a contract may protect.

3.3 Organizational Responsibility: Ownership and Custodianship

In addition to privacy and confidentiality concerns, further concern arises with ownership of information assets. The problem is that there is potential for erosion of information asset ownership when moving such resources to any external system. There is a fundamental difference between data ownership and having responsibility as a data custodian. Although the legal ownership of data will remain with the originating data owner, one potential area for concern with a public cloud is that the cloud provider may become responsible for both roles. There is no better example of this as when a law enforcement entity serves a warrant to a cloud provider for access to a tenant's information assets. Related to ownership concerns are concerns with where data resides and what jurisdictions it may traverse. The Internet presents a grand opportunity for the nosey and the wicked when it comes to the opportunity for surreptitiously examining someone else's secrets. In response to this, the European Union (EU) directive on Data Protection⁴ stipulated in which countries EU private and personal data may or may not traverse or reside. This has profound implications for all computing by EU member states. From the standpoint of cloud computing, the impact of this directive is likely shaping how public cloud providers, along with SPI service providers implement their services. This is a perfectly reasonable model for limiting the jurisdictional footprint of data to minimize the mischief that data is subject to in extraterritorial traversal, processing or storage. All tenants or end users of cloud services should be concerned by the potential that a public cloud or SPI service may push data or applications out of the jurisdiction in which the tenant resides or has legal obligations.

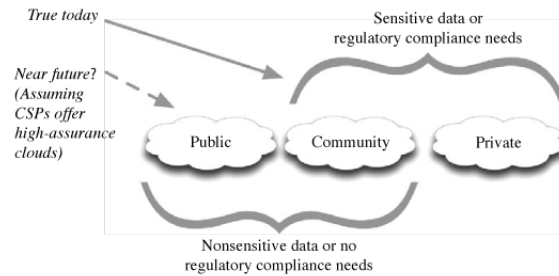


Figure 3.2: Meeting security needs: public, community, and private clouds.

While an organization has responsibility for ensuring that their data is properly protected as discussed above, it is often the case that when data resides within premises, appropriate data assurance is not practiced or even understood as a set of actionable requirements. When data is stored with a CSP, the CSP assumes at least partial responsibility (PaaS) if not full responsibility (SaaS) in the role of data custodian. But even with divided responsibilities for data ownership and data custodianship, the data owner does not give up the need for diligence for ensuring that data is properly protected by the custodian. By the nature of the service offerings, and as depicted in Figure 3.3, a data owning organization can benefit from their CSP having control and responsibility for customer data in the SaaS model. The data owning organization is progressively responsible beginning with PaaS and expanding with IaaS. But appropriate data assurance can entail significant security competence for the owning organization. Ultimately, risks to data security in clouds are presented to two states of data: data that is at rest (or stored in the cloud) and data that is in motion (or moving into or out of the cloud). Once again, the security triad (confidentiality, integrity, and availability) along with risk tolerance drives the nature of data protection mechanisms, procedures, and processes. The key issue is the exposure that data is subject to in these states.

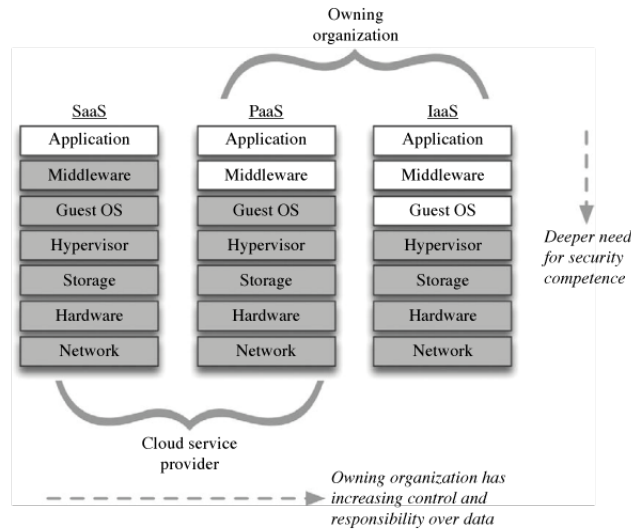


Figure 3.3: Owning organization has increasing control and responsibility over data

3.4 Data at Rest

Data at rest refers to any data in computer storage, including files on an employee's computer, corporate files on a server, or copies of these files on off-site tape backup. Protecting data at rest in a cloud is not radically different than protecting it outside a cloud. Generally speaking, the same principles apply. As discussed in the previous section, there is the potential for added risk as the data owning enterprise does not physically control the data. But as also noted in that discussion, the trick to achieving actual security advantage with on-premises data is following through with effective security. Referring back to Figure 3.2, the less control the data owning organization has decreasing from private cloud to public cloud the more concern and the greater the need for assurance that the CSP's security mechanisms and practices are effective for the level of data sensitivity and data value. (But in Figure 3.3, we saw that the owning organization's responsibility for security runs deeper into the stack for the owning organization as they move from SaaS to PaaS and again to IaaS.) If you are going to use an external cloud provider to store data, a prime requirement is that risk exposure is acceptable. Risk exposure varies in part as a function of service delivery as it does for deployment. A secondary requirement is to verify that the provider will act as a true custodian of your data. A data owning organization has several opportunities in proactively ensuring data assurance by a CSP. To begin with, selecting a CSP should be based on verifiable attestation that the CSP follows industry best practices and implements security that is appropriate for the kinds of data they are entrusted with. Such certifications

will vary according to the nature of the information and whether regulatory compliance is necessary. Understandably, one should expect to pay more for services that involve such certifications. One likely trend here is that higher assurance cloud services may come with indemnification as a means of insurance or monetary backing of assurance for a declared level of security. Whatever the future may hold, we can expect that practices in this space will evolve.

3.5 Data in Motion

Data in motion refers to data as it is moved from a stored state as a file or database entry to another form in the same or to a different location. Any time you upload data to be stored in the cloud, the time at which the data is being uploaded data is considered to be data in transit. Data in motion can also apply to data that is in transition and not necessarily permanently stored. Your username and password for accessing a Web site or authenticating yourself to the cloud would be considered sensitive pieces of data in motion that are not actually stored in unencrypted form. Because data in motion only exists as it is in transition between points such as in memory (RAM) or between end points securing this data focuses on preventing the data from being tampered with as well as making sure that it remains confidential. One risk has to do with a third party observing the data while it was in motion. But funny things happen when data is transmitted between distant end points, to begin with packets may be cached on intermediate systems, or temporary files may be created at either end point. There is no better protection strategy for data in motion than encryption.

3.6 Cryptography on Cloud

3.6.1 Data Encryption: Applications and Limits

In a recent article,¹ Bruce Schneier discussed how the information age practice of encrypting data at rest deviates from the historical use of cryptography for protecting data while it is communicated or in transit. One of Schneier's key points is that for data in motion, encryption keys can be ephemeral, whereas for data at rest, keys must be retained for as long as the stored data is kept encrypted. As Schneier points out, this does not reduce the number of things that must be stored secretly; it just makes those things smaller (the size of a key is far smaller than a typical data file). As Schneier states: "This whole model falls apart on the Internet. Much of the data stored on the Internet is only peripherally intended for use by people; it's primarily intended for use by other computers. And therein lies the problem. Keys can no longer be stored in people's brains. They need to be stored on the same computer, or at least the network, that the data resides on. And that is much riskier."² In meeting this challenge, there has been a recent rise in the number of security appliances that are intended to address this and related security implementation issues for data security in clouds.

3.6.2 Impediments to Encryption in the Cloud

In one example, a Software-as-a-Service public cloud, because of its very nature, might not allow subscribers to encrypt their data. This may be due to functional limitations with the actual service itself. In the example of currently available social networks including Facebook, MySpace, and LinkedIn, it is simply not possible to use encryption to ensure the confidentiality of your personal information. Nor would the cloud provider have any motivation to agree to allow this kind of data to be encrypted since many SaaS operators might not be able to provide revenue generating services if they have an obscured view to the data they are interacting with. For instance, if Facebook was unable to intelligently interpret what kind of activities were occurring in their cloud, then how could they target you with advertisements that are most effective if they relate to your posted activities? If your data was encrypted, then that aspect of the provider's business model would be broken. This same fact holds true to other kinds of clouds as well. IaaS providers might not be capable of encrypting at the operating system level because it would hinder their ability to monitor and therefore manage these instances.

3.6.3 State of the Art

Now the cryptography on cloud is a matter of interest between lots of Universities and Companies. For example Microsoft is doing some research on Homomorphic encryption^{page}. They are designing cryptosystems that support a variety of computations on encrypted data, ranging from general-purpose computations

(i.e., fully-homomorphic encryption) to special-purpose computations (e.g., voting and search). Their work on homomorphic encryption includes searchable & structured encryption. A searchable encryption scheme encrypts data in such a way that a token can be generated to allow a third party to search over the encrypted data. Using a searchable encryption scheme, a client can safely store its data with an untrusted cloud provider without losing the ability to search over it. They are also working on the related problem of structured encryption which allows a client to encrypt various types of data (e.g., social networks or web graphs) in such a way that complex queries can be performed over the encrypted data.

The most important interest of the research is on how to do some data mining on the crypted data. In our project we have not taken inspiration from other researches project.

Chapter 4

Project

4.1 Problem definition

The problem that we want to define belongs to the security of cloud computing especially in data storage. As we discussed in previous chapters, the cloud offers a perfect solution economically speaking and looking at performances. The only downside is the degree of control over the data that the provider gives you, whereas he has complete control over them.

We can make an real example to understand the problem:

Description of possible application scenario: B2B Company

The company “A” has a b2b("Business to Business") relationship, inter-trade, with N Companies. A company with a web-based solution manages all the relationships between the companies, such as: display of goods available, procurement, planning and monitoring of production, payments.

The e-commerce platform is at the heart of the company’s business model, the main features of the platform shall be: scalability, security, availability 24 / 7 in real time, accessible at any time regardless of user location. The main problem the company needs to solve is to maintain the functionalities of the system; to do this end there are two possible solutions:

- Private Cloud: This solution allows to have a platform in their company, with the advantage of having the highest safety integrity of the data to the detriment of a high cost of maintaining of hardware and specialist staff to monitor the proper functioning of the platform.
- Public Cloud: with this solution one of the differences is the advantage of drastically reducing the costs of continued operation of the platform, as it will be performed by an outside company for maximum system availability, with greater performance than the solution in-house. Outside company can offer the latest hardware that the company could not afford. This Cloud-based solution, however, demands the complete transfer of the

company's platform and data: in doing so the company will lose complete control over data.

The company is aware that the second solution would lead to a better platform and to the arrival of new potential customers, but at the same time it must consider the other side of the coin, that is give in custody to an "untrusted provider" its future. One of the problems is linked to the company's business model. In fact, the company sells the same products to several companies simultaneously, but the price is different for each company: the disclosure of the various discounts could compromise the future of the company, and especially the loss of customers.

4.2 Proposal

Our proposal wants to resolve the security issue of the Public cloud by ensuring the control over the data with the encryption of it. The protocol that we develop is a set of policies that give to the owner of data the complete assurance that the provider of the Public Cloud Infrastructure can't see the data in any way.

This is possible because the data and the decryption keys are in different place. The company, that offers the service, manages part of it with the key-server and the owner of the data has to manage the rest of it.

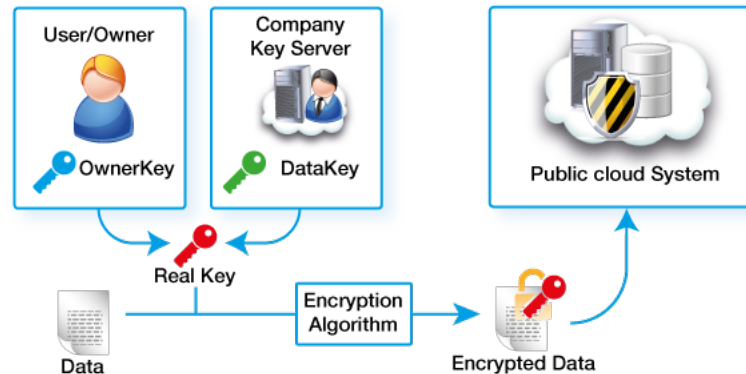


Figure 4.1: Representation of encryption

In the Figure 5.1 we can see that the OwnerKey belongs to the User or the Owner of data, such as a group of Users. The second part of the RealKey belongs to the company of service, that manages the keys with the KeyServer.

The company with the keyServer has control of the data access, in fact it can block the data in every moment. In the following part of this chapter we want to describe, in detail, all the actors and the main functions of our proposal.

4.3 Actors

4.3.1 Service Company



The server company is the owner of the application that is delivered into the public cloud. The company provides different services with the complete control over data.

4.3.2 Public Cloud System Provider



Providers of public cloud services where it is installed and runs the application of the Company. The provider is able to guarantee platform performance that the Company could not afford. The provider has to ensure data integrity and availability.

4.3.3 KeyServer



KeyServer is Company service which is installed on a different public cloud platform.

The role of the KeyServer is to control the access to the datakey and store it. The Keyserver is one of the most important features of the system because it has to be always be available to send the DataKey to the owner of the data, otherwise the owner can't decrypt the data.

4.3.4 Users/Owner



The User is the owner of the data that the service company gives to him or allows him to store. The User has the OwnerKey and he is the only one that has it. He has control over the data content with the Company thanks to KeyServer. He can use the Company service with a common web-browser, and when he has to share the owner key with others, we assume that he must do that through a secure channel.

4.4 Global Functionality

In this section we want to describe the main functions of the protocol with a step by step description. Therefore we start with the creation of the data by an Owner, then we will examine the reading, the update and sharing of data.

4.4.1 Data Creation

We want to describe the procedure to create a data from the client side(web-browser). In the following figure we can see the protocol step by step.

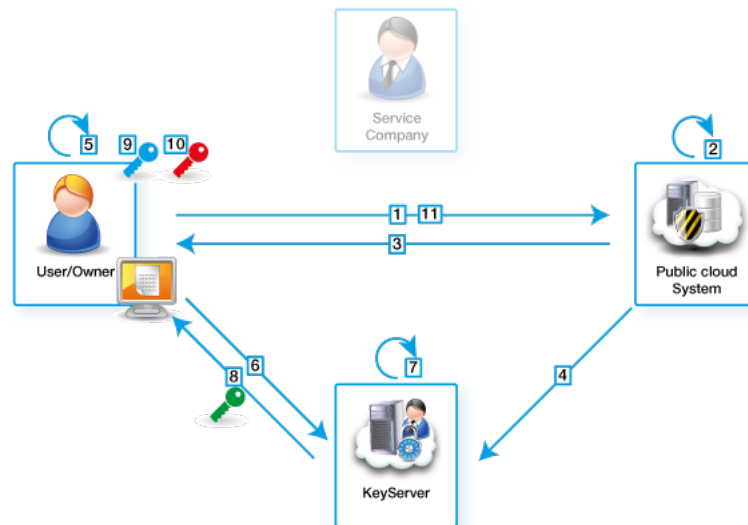


Figure 4.2: Data Creation

Data creation procedure:

1. User registration or login into Public Cloud Company Application.
2. Application creates a new valid session for the User; with that he can access all the services. The session has an expiration time.
3. Application sends the session to the User.
4. Application sends the session, the UserId and the expiration time for the session to the KeyServer.
5. User has the Data in the client-side.
6. User with the web-browser request from the Keyserver a new DataKey and a new DataID, The User sends also his current session that allows to access the new DataKey.
7. The KeyServer, after the request from the User checks if he has the right requirements and a valid session. If so, the KeyServer creates a new DataKey and a new DataID.
8. The KeyServer, after the creation sends Datakey and DataID to the User.
9. User in the web-browser inserts the Owner key, a new key that only he knows and he is the only person in charge of it. The system can't save the OwnerKey in any way.
10. User crypts the data using the Datakey+OwnerKey.
11. User uploads the crypted data into Public Cloud Company Application.

4.4.2 Data reading

The data reading allow the User to access to his Data content.

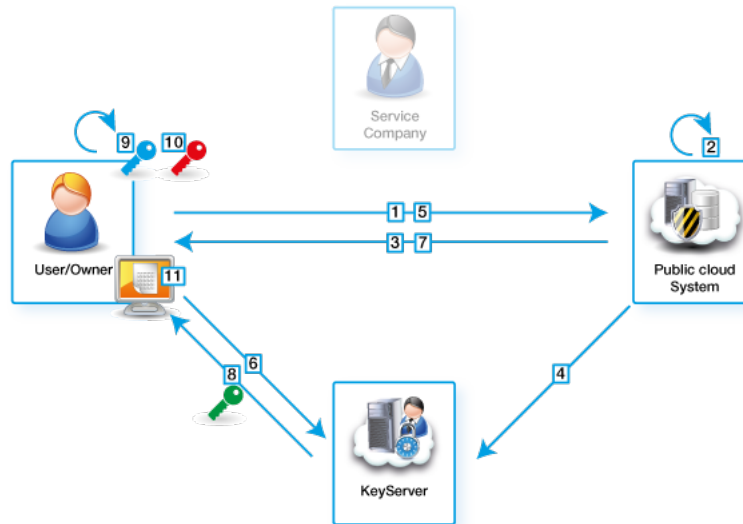


Figure 4.3: Data Reading

Data reading procedure:

1. User login into Public Cloud Company Application.
2. Application creates a new valid session for the User; with that he can access all the services. The session has an expiration time.
3. Application sends the session to the User.
4. Application sends the session, the UserId and the expiration time for the session to the KeyServer.
5. User requests the crypted Data to Public Cloud Company Application.
6. User with the web-browser request from the Keyserver the DataKey with the DataID, The User sends also his current session that allows access to the DataKey.
7. The Public Cloud Company Application sends the crypted data to the User.
8. The KeyServer after the request from the User checks if he has the right requirements and a valid session. If so, the KeyServer send the DataKey.
9. User in the web-browser inserts the Owner key.

10. User decrypts the data using the Datakey+OwnerKey.
11. User can read the data content.

4.4.3 Data Updating

This procedure is a combination of reading and creation. The User reads the data with the DataId after the decryption; he can update the data content afterwards, crypt with the old keys or change it with a new one. It's up to him.

4.4.4 Data Deletion

In this procedure the User sends the deletion request to Public Cloud Company Application. They then deletes the crypted data and also sends the request of deletion to the KeyServer. The KeyServer deletes the Datakey for the DataId.

4.5 Different scenario

4.5.1 User unique owner of data

In this scenario, the user benefits from a service that the company has developed and that run is done by a public cloud provider. In this specific case the data is property of the user which is the one and only, able to view it. The Company is solely responsible for the management of the data. All the main features that we described in the previous section are perfect for this scenario.

4.5.2 Company owner of data

In this scenario, the user benefits from a service that the company has developed and that run is done by a provider. In this specific case the data is owned by the Company and the content is related to the User. The Company is the only one that can create and update the data. User can only read the data. Company has the OwnerKey, that is sent to the User through a secure channel.

4.5.3 User and Partners

In this scenario we describe the sharing of data with partners. There are two different way to share it:

- **User sends the data to the Partners:** The user after a reading will copy the data and send it into a secure channel. This is a simple way that allows the User to share data outside the Application. The company hasn't got any control of that, only the User has the control of this operation.

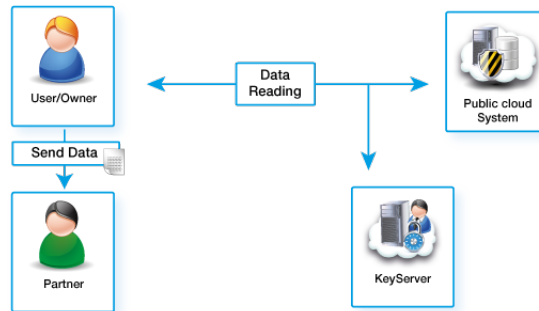


Figure 4.4: User send the data to the Partners

- **User creates a copy of Data:** The user duplicates the data into the Public Cloud Application, uppermost he reads the data and creates a copy of it with a new DataKey and a new PartnerKey. The Application gives access to the Partner(a registered User). Therefore Partner has control on the data copy so he becomes the owner of that data copy.

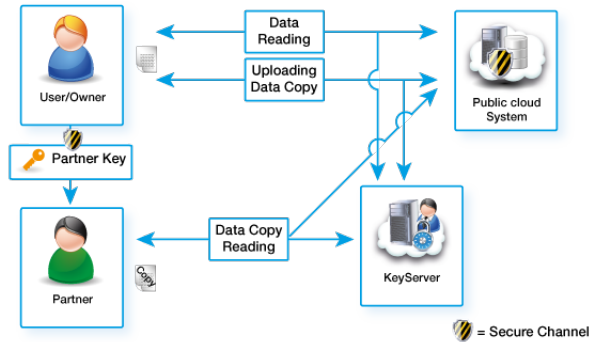


Figure 4.5: User create a copy of Data

- **User adds Partner into Data Owner:** In this case the User wants to share the data with the Partner, in addition to this he wants to extend the ownership of the data. In this way the User allows the Partner to read and update the data, preventing him to share with other Partners. To do this the User must send in a secure channel the OwnerKey to the Partner, and add into the Application the ownership extension. The only function that changes is uploading because the Partners and the Users can't change the OwnerKey. If the Users wants to delete the ownership extension he will block access of the Partner into the KeyServer and into the Public

Cloud Application. The User will remain the real data Owner, he has the control of sharing the data.

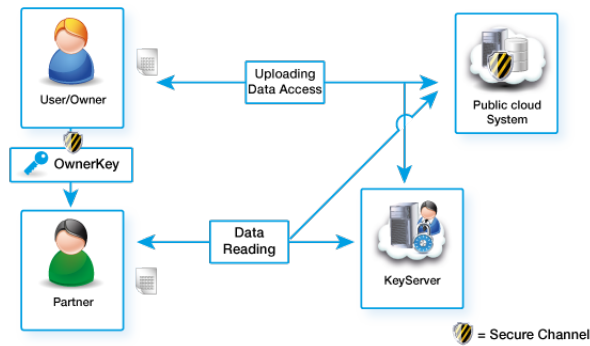


Figure 4.6: User add Partner into Data Owner

4.5.4 Company and Users groups

This scenario is really close to the previous case in fact the Company represents the Real Data Owner and can share it with a group of Users (Partners). The only difference is that the company has the complete control over data and User. The Users have no control of the data. The Company in fact, is a SuperUser that can control all the data content. In this situation in fact the owner of the data is property of the Company. The Company must send to the Users in a secure channel all the OwnerKeys.

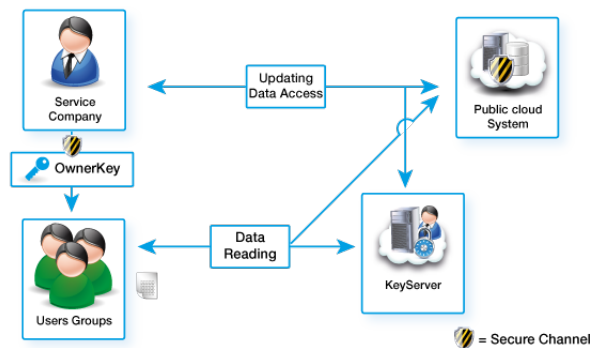


Figure 4.7: User add Partner into Data Owner

4.6 Advanced Features

The following functionalities are the main features with some changes. These advanced features can make the protocol more secure at the expense of performance because they increment the network communication between the actors.

4.6.1 Encrypted OwnerKey with MasterKey

This particular functionality is an update in the management of the OwnerKey. In this procedure we want to help the User save the OwnerKeys. It is one of the most important variable of the protocol, and the User is the only one that must remember it. In the case of the User having lots of data, it could be difficult to store it in local. With this new feature the system allows Users to store the OwnerKeys into the public Cloud Application. The User has to encrypt the OwnerKey with the MasterKey, a new Key that he has to use to encrypt all the OwnerKeys. In this way the User, in order to read and create, has to insert the MasterKey to decrypt/encrypt the OwnerKey and send it with the encrypted data into Public Cloud Application.

4.6.2 Checksum Store into KeyServer

In this functionality we want to make more secure the reading of the data because an important feature of the decryption is represented by the checksum. Like wikipedia[12]says “checksum or hash sum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. The integrity of the data can be checked at any later time by recomputing the checksum and comparing it with the stored one. If the checksums match, the data were almost certainly not altered (either intentionally or unintentionally).”

The checksum in the reading procedure can prove if the data is not altered, this allows the Company to know that also with a brute force over the encrypted data they can't be certain of the data content.

In this case the changes in the creation procedure are: User after encrypting the data sends the checksum to the Keyserver and encrypted data in the public Cloud Application.

The changes in the reading procedure the changes are:

- KeyServer sends to the User the DataKey and the Checksum value.
- Only the User with the OwnerKey, DataKey and the checksum can see the data content.

Chapter 5

Developing

In this chapter we want to show and describe a demo application that respects our proposal. We have created this simple demo to prove all the main features and prove that the protocol works. Our scope was to show how to have total control over the data into a Public Cloud System. We took two different platforms, a PaaS and IaaS to develop and install the web-based demo application. The Paas solution is Google Appengine, a public cloud platform for developing and hosting web applications in Google-managed data centers. Google Appengine virtualizes applications across multiple servers. App Engine offers automatic scaling for web applications: as the number of requests increases for an application, it automatically allocates more resources for the web application to handle the additional demand. The Iaas solution is a VPS(Virtual private server) from a common hosting company.

5.1 Web-application side

5.1.1 Interface Overview

The web application side wants to show how an application could be on the public cloud. The demo that we made in the first implementation is on a LAMP platform installed in a Virtual Private Server. The second implementation is under Google Appengine. The demos display all the features that we described in the previous chapter.

We allow the User to register or login into the Application, after that he can add data, read data, update data and share data with other Users.

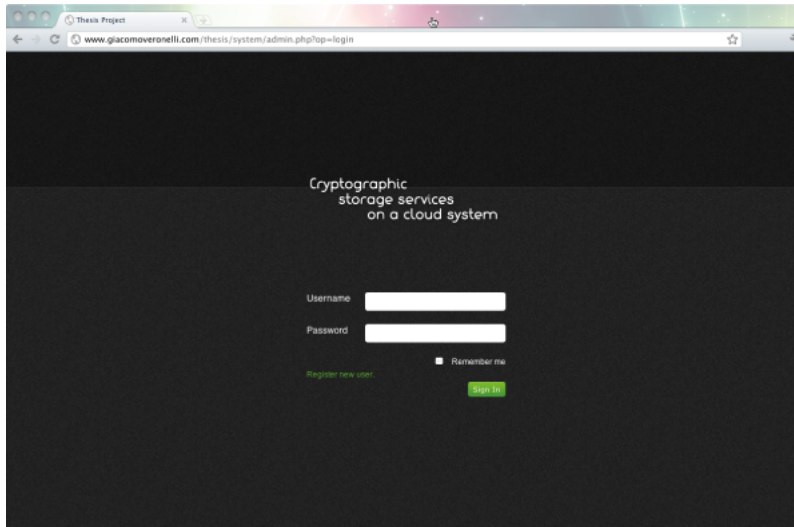


Figure 5.1: Application Login Screenshot

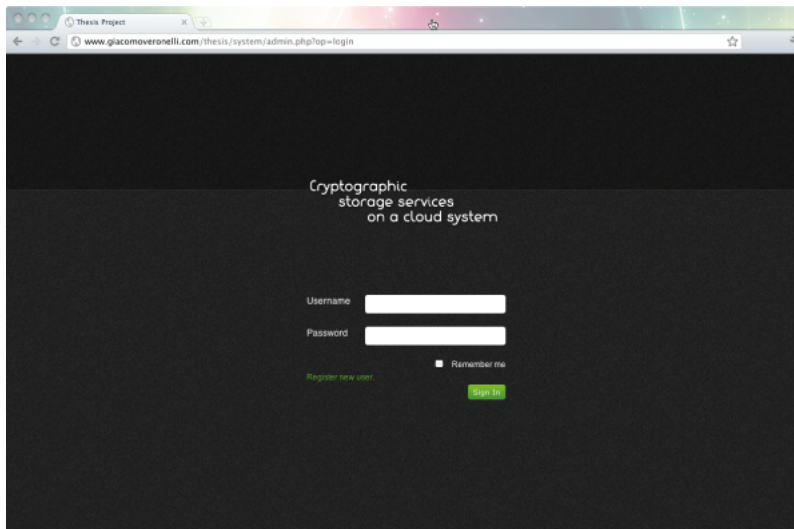


Figure 5.2: Application Registration Screenshot

In the previous figure we showed the web-based access to the Demo Application. The User after the login can access the menu of the functionalities that we can see in the next figure. The menu is allocated on the right side and it's always present in all the pages, for User is very simple to switch the application functionalities.

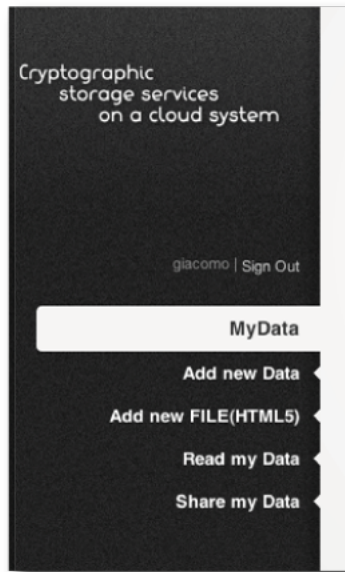


Figure 5.3: Functionality Menu

In the Demo Application, User can add data to store on the Public Cloud Application, the kind of data could be plain text or file. Now we will analyze step by step every functionality.

5.1.2 Add new Data

In this section we will describe all the procedure of data creation in the demo application. First the User has to be logged into the system with a valid account, in fact we can see the session code that allows the user to access all the functionalities of the system.

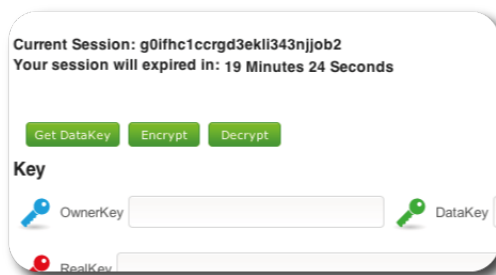
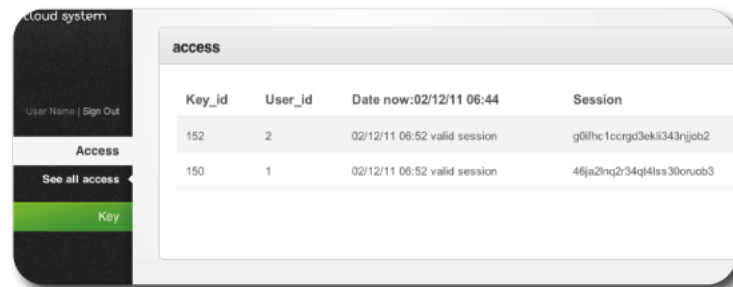


Figure 5.4: Session countdown

The current session is the session that the Application created after the login. At the same time the Application sends the same session to the KeyServer with the expiration time. This allow the Keyserver to control the DataKey request. In the admin panel of the KeyServer we can see the actual session, the system shows only just the last one of User besides we can see if the session is valid or expired. In this demo application the session expiration time is set at 20 min.



Key_id	User_id	Date now:02/12/11 06:44	Session
152	2	02/12/11 06:52 valid session	g0lfhc1ccrgd3ek3343njjob2
150	1	02/12/11 06:52 valid session	46ja2lnq2r34q4lss30oruob3

Figure 5.5: KeyServer Access Admin Panel

The logged in and authorized User can insert a new OwnerKey, in the Demo it is all up to the User the type of OwnerKey however he could be leave it blank.



Key

OwnerKey DataKey

RealKey

Figure 5.6: Insert of OwnerKey

The User now can request a new DataKey that allows him to encrypt the data. With the button “Get DataKey” User sends an ajax asynchronous call to the KeyServer, the User in the request sent CurrentSession, UserID, TypeOfRequest(new dataKey). The KeyServer, if the CurrentSession is still valid and the User has the correct requisites, creates a new DataKey and a new DataID after that the KeyServer sends the data to the web-browser with a Json answer. Json is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects; the JSON format is often used

for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML.[?]

In the following figure we can note the requested procedure. In the Demo we wanted to show the current status of the asynchronous request.

```
function Get_DataKey() {

    $.getJSON( 'http://upload.waaa.it/thesis/receive.php', { session: 'bbj169152m51k3gcph4mh8eqi6 ', user_id: '1', o: 'get', key_id: '68' },
    function(data) {
        var items = new Array();    ii=0;
        $.each(data, function(key, val) {items[ii]=val; ii=i+1;});
        if (items[0]== "ERROR") {
            $("#datakeystatus").html( '<span_class="input-notification_error_png_bg">' + items[1] + '</span>' );
            alert (items [1]);
        } else {
            $("#datakeyrealid").val(items [1]);
            $("#datakey").val(items [0]);
            $("#datakeystatus").html( '<span_class="input-notification_success_png_bg">Success</span>' );};
    });
}
```

Figure 5.7: GetDataKey Javascript Code

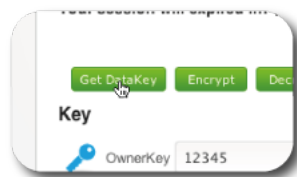


Figure 5.8: Get DataKey Procedure

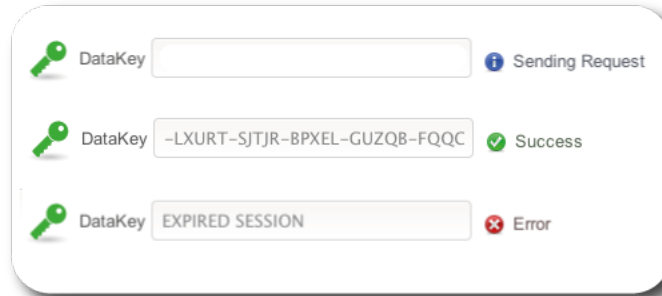


Figure 5.9: Get DataKey Request Status

The response time of the KeyServer in the tests that we have done is really quick, the average is around 1sec; nevertheless it will always depend on the network and the User connection.

After the insertion of DataKey and OwnerKey, the User can insert the Data that he wants to store into the public Cloud Application.

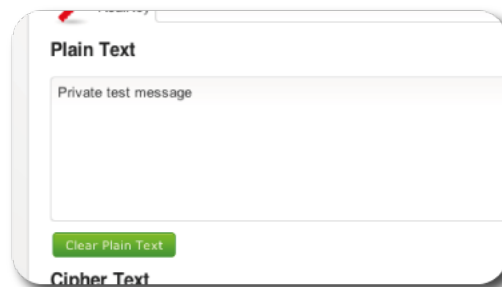


Figure 5.10: Data insertion

Now the User can encrypt his data into the Application, with the Encrypt button (a Javascript function is called). The Encrypt function with the RealKey and the PlainText creates a Cipher Text. The User also can insert a DataName or Description, that will help him to recognize his data.



Figure 5.11: Encrypt button

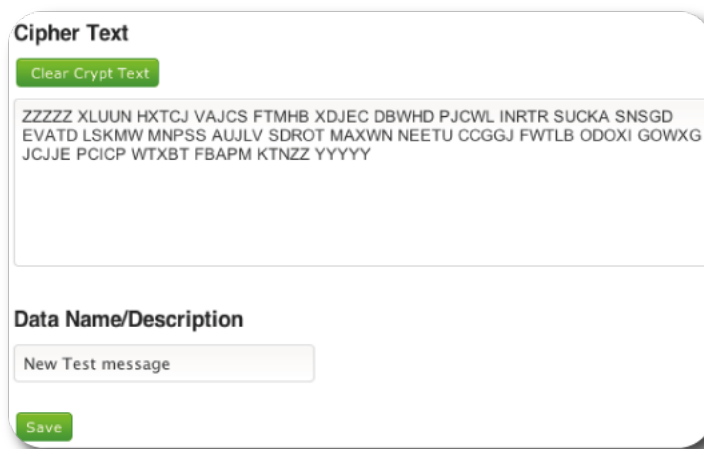


Figure 5.12: Cipher Text

The User can now store the data with the “Save” button. After the procedure, user will be redirected into myData pages, that allows him to see all the data and the recently inserted one.

ID	Data	Titolo	EDITA
22	02/12/11	New Test message	
21	30/11/11	60-proiect.ppt	

Figure 5.13: My data panel

5.1.3 Read my data

The user in the read my data page can see all his data. When the User select a data, the application redirects him to the view page of the specific data.

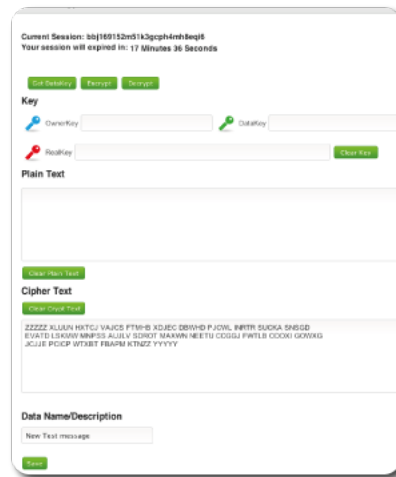


Figure 5.14: Read data panel

In this procedure the User has to do the same step that we saw in the creation procedure. The only difference is the `GetDataKey` function, because this time in the request we insert the `DataId`, and the `typeOfRequest` is different (reading).

The User, when having all the keys, can press the “Decrypt” button that allows him to see the data content.

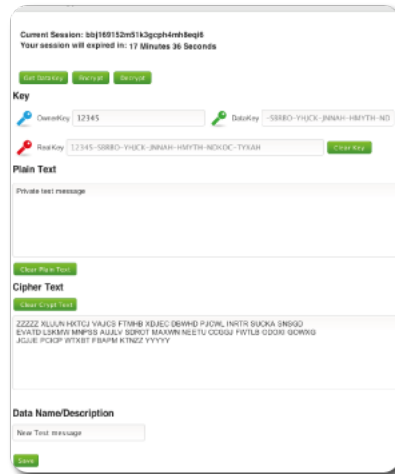


Figure 5.15: Read data panel

After the decrypt procedure the Application allows the User to update the plain text and he can encrypt again with a new DataKey and OwnerKey if he wants and then he can save the updated data into the public Cloud.

5.1.4 Add new File

In this section we show how to store files, not simple plain text. For this feature User must use an updated Web-Browser, in the tests we always use Google Chrome that supports the new features of HTML5. The most important feature that we used is the local filesystem. HTML5 provides very powerful APIs to interact with binary data and a user's local file system. The File APIs give web applications the ability to do things like read files synchronously, create arbitrary Blobs, write files to a temporary location, recursively read a file directory, perform file drag and drop from the desktop to the browser, and upload binary data using XMLHttpRequest.[10]

In the add file page, all the Users can drag and drop into the panel the file that they want to encrypt.

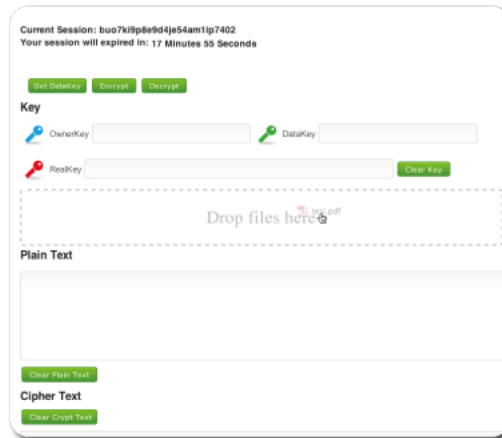


Figure 5.16: Drop File

After he has dropped the file, the web-browser will translate from binary to base64. Base64 is a group of similar encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. After the translation in the demo application, the User can see the base64 string that represents the file content.



Figure 5.17: Base64 String

The User after the insertion of the Keys, like in the plain text creation, can encrypt the file and store it into the system. In the following figure we can see that after cipher text the name of the data is already filled with the original file name attached at the DataId.

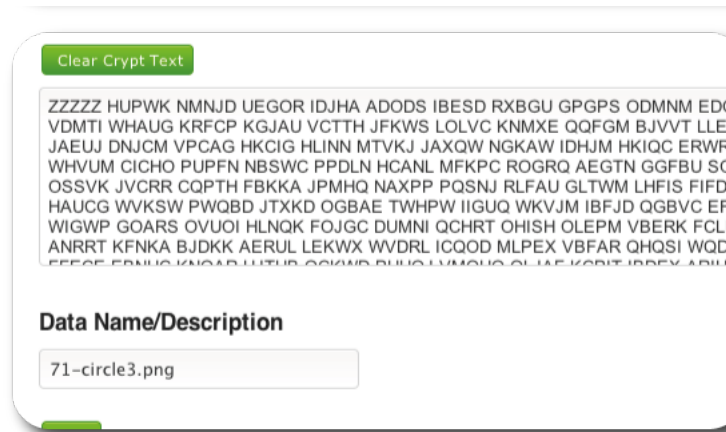


Figure 5.18: Encrypted file

This functionality could be not supported in the same way in older web-browsers. We hope that html5 and the web-browsers in general will improve and that also all the functionalities will be more portable to build applications.

5.1.5 Share my Data

One of the most important features is the file sharing with Partners. In the demo we develop the Users sharing, with the two scenarios that we describe in the proposal chapter, which are: User creates a copy of Data and User adds Partner into Data Owner. In the next picture we can see the share page for a data. Before the sharing page User has to choose the data that they want to share.

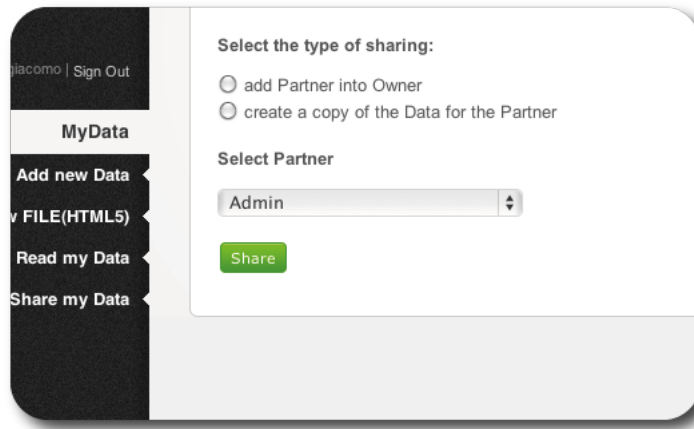


Figure 5.19: Data sharing panel

The next step is to select the kind of sharing and the Registered User that he wants to share data with.

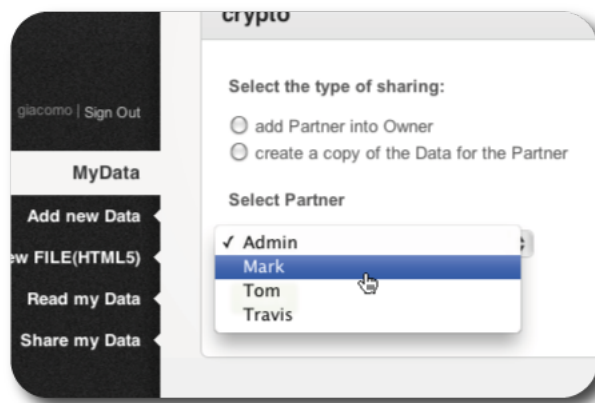


Figure 5.20: Registered User list

If the User wants to add the partner to the Owner the procedure is over, the application will add the Partners in the Owner of that Data. Now Partner can read the shared data, the application and the Keyserver will allow him all the operations that belong to the shared data.

If the User chooses the copy creation for the partner, the system will redirect the user in a page similar to the reading page, and after the decryption of data,

User has to insert the PartnerKey and encrypt again the data to store it in order to share with the partner. The last step for the User is to send the PartnerKey to the Partner through a secure channel.

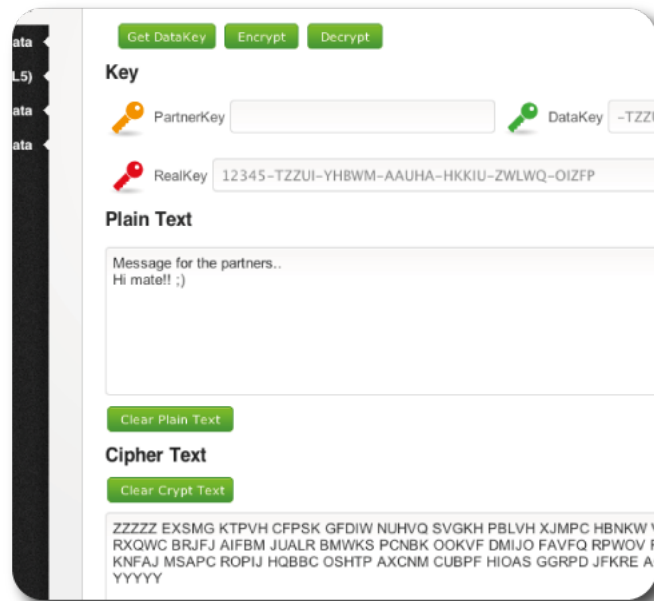


Figure 5.21: User create a copy of Data

5.1.6 LAMP platform

LAMP is an acronym for a solution stack of free, open source software, referring to the first letters of Linux (operating system), Apache HTTP Server, MySQL (database software) and PHP (or sometimes Perl or Python), principal components to build a viable general purpose web server.[13] The GNU project is advocating people to use the term "GLAMP" since what is known as "Linux" is known to GNU as the GNU/Linux system.



Figure 5.22: Lamp Solution

The exact combination of software included in a LAMP package may vary, especially with respect to the web scripting software, as PHP may be replaced or supplemented by Perl and/or Python. Similar terms exist for essentially the same software suite (AMP) running on other operating systems, such as Microsoft Windows (WAMP), Mac OS (MAMP), Solaris (SAMP), iSeries (iAMP), or OpenBSD (OAMP). Though the original authors of these programs did not design them all to work specifically with each other, the development philosophy and tool sets are shared and were developed in close conjunction. The software combination has become popular because it is free of cost, open-source, and therefore easily adaptable, and because of the ubiquity of its components which are bundled with most current Linux distributions. When used together, they form a solution stack of technologies that support application servers.

5.1.7 Google Appengine

We decided to develop on one of the best PaaS, Google App Engine. Google App Engine is a platform that lets you run web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users. You can serve your app from your own domain name using Google Apps. Or, you can serve your app using a free name on the appspot.com domain.



Figure 5.23: Google App Engine and GWT

You can share your application with the world, or limit access to members of your organization. Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features two dedicated Python runtime environments, each of which includes a fast Python interpreter and the Python standard library. Finally, App Engine provides a Go runtime environment that runs natively compiled Go code. These runtime environments are built to ensure that your application runs quickly, securely, and without interference from other apps on the system. With App Engine, you only pay for what you use. There are no set-up costs and no recurring fees. The resources your application uses, such as storage and bandwidth, are measured by the gigabyte, and billed at competitive rates. You control the maximum amounts of resources your app can consume, so it always stays within your budget. App Engine costs nothing to get started. All applications can use up to 1 GB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.[11]

We chose the Java developing. Java runtime environment using common Java web development tools and API standards. Your app interacts with the environment using the Java Servlet standard, and can use common web application technologies such as JavaServer Pages (JSPs). The Java runtime environment uses Java 6. The App Engine Java SDK supports developing apps using either Java 5 or 6. The environment includes the Java SE Runtime Environment (JRE) 6 platform and libraries. The restrictions of the sandbox environment are implemented in the JVM. An app can use any JVM bytecode or library feature, as long as it does not exceed the sandbox restrictions. For instance, bytecode that attempts to open a socket or write to a file will throw a runtime exception. Your app accesses most App Engine services using Java standard APIs. For the App Engine datastore, the Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces. Your app

can use the JavaMail API to send email messages with the App Engine Mail service. The java.net HTTP APIs access the App Engine URL fetch service. App Engine also includes low-level APIs for its services to implement additional adapters, or to use directly from the application.

5.1.8 Client browser side

In the client browser we developed the most important feature of the protocol. We used Javascript for the runtime procedure on the web-browser, Html for the interface and Ajax for the asynchronous request to the KeyServer that allow us to use encrypt data without the Public Cloud Provider.

In particular Ajax is a group of interrelated web development methods used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data is usually retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed (JSON is often used instead), and the requests do not need to be asynchronous.[2] Ajax is not one technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

We also used JQuery for the developing of the javascript. JQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. Used by over 49% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today. JQuery is free, open source software, dual-licensed under the MIT License or the GNU General Public License, Version 2. JQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. JQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery framework allows the creation of powerful and dynamic web pages and web applications.

For the user interface we used Html and for the file creation most of the new feature of Html5.

5.1.9 Server side on VPS solution

In the server side of Lamp solution we developed all the web-dynamic code in Php. With a self-made Cms we built the Demo Application. We will see the structure of the Cms in the next section. In the server we used MySQL to store the data in the database. In the public cloud database we created tree tables: data, user, ownership.

In the next Figure we can see the structure of the tables.

```

CREATE TABLE IF NOT EXISTS `data` (
  `data_id` int(11) NOT NULL AUTO_INCREMENT,
  `data_name` text NOT NULL,
  `data_cipher` longtext NOT NULL,
  `data_user_id` int(11) NOT NULL,
  `data_date_creation` bigint(20) NOT NULL,
  `data_date_last_update` bigint(20) NOT NULL,
  `data_real_id` int(11) NOT NULL,
  `data_type` varchar(5) NOT NULL,
  PRIMARY KEY (`data_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

CREATE TABLE IF NOT EXISTS `user` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(20) NOT NULL,
  `user_email` varchar(30) NOT NULL,
  `user_pass` varchar(100) NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `user_name` (`user_name`,`user_email`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

CREATE TABLE IF NOT EXISTS `share` (
  `user_id` int(11) NOT NULL,
  `data_id` int(11) NOT NULL,
  PRIMARY KEY (`user_id`,`data_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;

```

Figure 5.24: Sql Creation code

5.1.10 CMS with MVC

In the demo application we used a self-made cms in php. This simple cms respected the MVC (Model–view–controller) that is a software architecture, currently considered an architectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from the user interface (input and presentation), allowing independent development, testing and maintenance of each (separation of concerns). Model View Controller (MVC) pattern creates applications that separate the different aspects of the application, while providing a loose coupling between these elements.

In the demo application we create a tree controller: data, admin, user. The admin is a controller that allows only the admin users to see all the data and ownership. The data controller has these functionalities:

- Insert new data

- Read all
- Read a specific data
- Update a specific data
- Delete a specific data
- Share a copy of a specific data
- Add partner into data owner for a specific data

Not all of these functions have the view. In fact only read and read all have a specific view, while the other redirect the user in different views.

The Cms was really useful for the demo implementation, nevertheless on Google app engine we built a copy of the php version in java. We converted all the controller and most of the views.

5.2 Datakey service

5.2.1 Overview

KeyServer is a server with some services that allow the user to get DataKey. The services are two: send and receive.

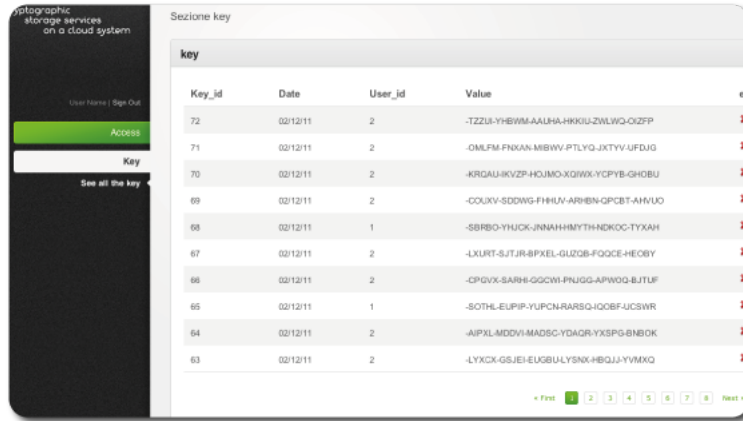
The send service has the feature to receive into the KeyServer only from Public Cloud platform the new session key for the User that enters into the Application or updates the expired session. In input the send service wants UserId, SessionCode, ExpiredTime. The send service returns to the Public Cloud platform only the confirmation of success or otherwise the type of error.

The receive service has these functionalities:

- Receive a new DataKey and DataId
- Receive the DataKey for a specific Data

These two functionalities respect the protocol; to allow the User to receive the DataKey, he has to send the Session and after a check, if he has the right requisites, the KeyServer sends the DataKey.

Besides, the KeyServer has an admin panel built on the same cms of the demo. The panel allows only the Admin to monitor the access and the keys.



The screenshot shows the KeyServer Admin Panel. On the left is a dark sidebar with the text "cryptographic storage services on a cloud system", a "Sign Out" link, and buttons for "Access" and "Key". The main area is titled "Sezione key" and contains a table with the following data:

Key_id	Date	User_id	Value	ed
72	02/12/11	2	-TZZU-YHBWM-AALHA-HKKIU-ZWLWQ-OIZFP	X
71	02/12/11	2	-OMLFM-FKXAN-MBWV-PTLYQ-JXTYV-UFDJG	X
70	02/12/11	2	-KRDJU-RKVZP-HJUMO-XQIWX-YCPYB-GHOBV	X
69	02/12/11	2	-COUXV-SDDWG-FHHLV-ARHBN-GPCBT-AHVUO	X
68	02/12/11	1	-SBFBO-YHUCK-JNNAH-HMYTH-NDKOC-TYXAH	X
67	02/12/11	2	-LXURT-SJTR-8PXEL-GLZOB-FOOCE-HEOBY	X
66	02/12/11	2	-CPGVX-SARH-GGCWI-FNUGG-APWOG-BJTUF	X
65	02/12/11	1	-SOTHE-EUPIP-YUPCN-RARSG-IOCBF-UJCSWR	X
64	02/12/11	2	-AIPXL-MDDVI-MADSC-YDAQR-YXSPG-BNBOK	X
63	02/12/11	2	-LYXCX-GSJEI-EUGBU-LYSXN-HBQJA-YVMXQ	X

At the bottom of the table, there is a pagination control: "First" (highlighted), "2", "3", "4", "5", "6", "7", "8", "Next".

Figure 5.25: KeyServer Admin Panel

In the next Figure we can see the structure of the tables.

```
CREATE TABLE IF NOT EXISTS 'access' (  
  'access_id' bigint(20) NOT NULL AUTO_INCREMENT,  
  'access_user_id' int(11) NOT NULL,  
  'access_session' text NOT NULL,  
  'access_time_exp' bigint(20) NOT NULL,  
  PRIMARY KEY ('access_id')  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;
```

```
CREATE TABLE IF NOT EXISTS 'key' (  
  'key_id' bigint(20) NOT NULL AUTO_INCREMENT,  
  'key_user_id' bigint(20) NOT NULL,  
  'key_value' text NOT NULL,  
  'key_date' bigint(20) NOT NULL,  
  PRIMARY KEY ('key_id')  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;
```

```
CREATE TABLE IF NOT EXISTS 'share' (  
  'user_id' int(11) NOT NULL,  
  'data_id' int(11) NOT NULL,  
  PRIMARY KEY ('user_id', 'data_id')  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;
```

Figure 5.26: Sql Creation code

5.3 Encryption

5.3.1 Browser-Based Cryptography Tools

In the web browser side, we used a javascript algorithm to encrypt and decrypt the data. This algorithm was written by John Walker[11]. In the demo version we used that algorithm but the protocol can work with any symmetric cryptographic algorithm.

In the demo application we used the AES 256 symmetric cryptographic algorithm, that is a secure and solid choice. With the algorithm we have also the checksum value inside the cipher text (first 25 bytes); this is really useful because with that we can always know if the decryption is correct or not. One of the protocol scenario has the checksum into the KeyServer, this feature is really useful but needs more callback between User side and KeyServer.

5.4 Conclusion

In this project we wanted to design a protocol able to, with a series of policies and rules, resolve the security concern problem of control over data. With the demo application we prove that it works. The demo application respects every single part of the protocol. In the development we made a lot of choices that respected the protocol such as the encryption algorithm, the merge algorithm for OwnerKey and DataKey, the type platform and the programming languages. All these choices were free constraints that is up to the developers.

The protocol can be integrated in all the Applications. The limit of the protocol is data mining over the encrypted data. However it could be implemented if we use homomorphic encryption, but we also have to change the policy about the Keys because the data has to encrypt with the same key, but that's in contrast with the protocol.

Bibliography

- [1] Mell P, Grance T. The NIST Definition of Cloud Computing Version 15; 2009, National Institute of Standards and Technology, Information Technology Laboratory.
- [2] Securing the Cloud: Cloud Computer Security Techniques and Tactics, Graham Speake, Patrick Foxhoven, Elsevier Science & Technology, May 1, 2011
- [3] Cloud Security: A Comprehensive Guide to Secure Cloud Computing By Ronald L. Krutz, Russell Dean Vines
- [4] Code in the Cloud, Pragmatic Programmers, LLC, The, Jan 25, 2011
- [5] A Quick Start Guide to Cloud Computing: Moving Your Business Into the Cloud, Kogan Page, 28/nov/2010
- [6] <http://en.wikipedia.org/wiki/Checksum>
- [7] [http://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](http://en.wikipedia.org/wiki/LAMP_(software_bundle))
- [8] <http://en.wikipedia.org/wiki/JSON>
- [9] <http://www.bitsandbuzz.com/article/cloud-portability-force-com-vs-google-app-engine-vs-amazon/>
- [10] <http://www.html5rocks.com/en/features/file>
- [11] <http://code.google.com/appengine/docs/whatisgoogleappengine.html>
- [12] <http://www.fourmilab.ch/javascript/>
- [13] <http://research.microsoft.com/en-us/projects/cryptocloud/>