The Single Sign On and Unlinkability in Cloud Computing

Lalou Stavroula
Master of Computer Science

T.E.I. of Larissa

Staffordshire University

Technological Educational Institute of Larissa
August 2011

Technological Educational Institute of Larissa

The Single Sign On and Unlinkability in Cloud Computing

Lalou Stavroula

Master of Computer Science, 2011

Thesis Summary

This thesis discusses the available unlinkability protocols and mechanisms. Investigatesolutions in similar environments. Also propose suitable mechanism for cloudcomputing. The problem is that different CSP may share knowledge and determine aprofile of the transactions carried out by the same user. This may lead to disclosing,partially or fully, the user's identity or some attributes about it. This thesis goal is to improve and secure the communication between users anddistributed cloud service providers with the use of an intermediary entity, which iscalled infomediaries ,it is a business model, and enchases privacy. It is capable ofproviding anonymity, privacy and security, to customers and vendors of e commerce.This thesis uses the term of informediaries, and introduces a new protocol whichaims to enhance unlinkability and keep secret the personal information of users ina cloud environment. There are many types of attacks, such as man in the middle,traffic analysis, passive eavesdropping and trace back attacks that aim on users personalinformation in order to take advantage of them and use them for malicious purposes.

Keywords: Cloud computing, Single Sign On, unlinkability, authentication,anonymity, security.

Dedicated to my family

Acknowledgements

I would like to thank my supervisor, Dr. Alex Papanikolaou for his supervision and his valuable advice to my research.

I want to thank the TEI of Larisa and Staffordshire University for providing the great opportunity of study and facilitating me throughout the research.

I want to thank my parents, Anthoula and Apostolis Lalos and my brother George who never give up providing supports to my study. More than anyone else, I want to thank my life companion, Konstantinos, for his patience and his support during my study.

Table of contents

Chapter 1

Introduction

1.1 Motivation

In cloud computing users request many services from different Cloud Service Providers (CSP), in that case privacy issues are raised. The user should provide credentials in order to authenticate, these credentials are shared among different Cloud Service Providers CSP. The user's information should not appear or the information should be overshadowed. Unlinkability is defined as a property which ensures that a user may make multiple uses of resources or services without others being able to link those uses together. In unlinkability of two or more items is defined as a property that, within the system, those items are no more and no less related than they are related concerning the a priori knowledge ~\citeasnoun{bertino}.

Use of internet as a protocol for sharing data and information has become essential for international trading and businesses. This way of sharing data increases constantly. Nowadays businesses and organizations are using internet as one of their main portals for exchanging information with clients, co operational partners, employees etc. Exchanging data this way has made the work more efficient, mobile and made cooperation with own and other businesses around the world easier. An important factor is that this is very cheap, time and cost saving.

The 'internet revolution' doesn't only bring up positive aspects. Related to transferring data over internet and other networks, security risks have become a big issue. When

transferring information over an 'open' channel, as internet can be called, these data's can be picked up and used by any user. It has opened the doors for malicious activities, i.e. hacker attacks, exploits etc. Also dishonest employees can be the ones that want to get access to the data.

1.2 Aims of the Thesis

The aim of this research is to review and evaluate the most prevailing available unlinkability protocols and mechanisms. Also it mentions the existing techniques and how the problem of identity or personal information disclosure was faced in other environments and proposes a protocol that enhances unlinkability for the cloud environment.
The major contribution of this is the designing and implementation of a prototype framework with the following functions:

The first aim is to present an approach in which easily and safely users to communicate with distributed CSP over the cloud environment, also to protect user's unlinkability and identity disclosure in multiple transactions over the cloud requesting services through different CSPs. To present an approach that when a user requests a service from a CSP and then from another CSP and ensures that a user may make multiple uses of resources or services without others being able to link those uses together. Also to prove that the use of an Infomediary, enhances unlinkability and anonymity of users. The final aim is to evaluate the performance of the suggested protocol and suggest future improvements.

1.3 Research Methodology

1.3.1 What is the activity of research

Privacy is normally conceived as a problem of regulating the disclosure of identity information. One of the most common privacy issues is the linkability problem. Given anonymous dataset, which belongs to individuals, it is possible to associate the data of an individual with each other. Most current works are using suitable location privacy and secure routing mechanisms to hide the identity and/or location of a user, audit information of that user's accesses to various services is stored across various databases. It is possible for other users such as system administrators to correlate transaction information, including timing, across audit logs to expose identity, location, transaction history, and other sensitive attributes of a user.

The main theme of the research is allowing individuals to interact unlinkably with cloud service providers, and making multiple transactions while preventing them from finding out the individuals' identity.

It is quite vital to be able to maintain unlinkability among multiple transactions that have been carried out by the same user with the same or different CSPs. Unlinkability was also a major security issue in other situations, such as the Secure Electronic Transactions (SET) system, where any information regarding the user is disclosed on a need-to-know basis among the participating entities of the system. So far different techniques have been deployed, such as the use of pseudonyms.

1.3.2 Security issues

The already available protocols and models have the following vulnerabilities:

- Many of the already widely used systems and protocols are customized to provide encryption methods but do not offer anonymity and unlinkabilty because such systems are aware of all information about the participants, their behaviour, history of used services and content of messages. They are vulnerable in trace back or spoofing attacks.

- They provide encryption mechanisms as a means to provide anonymity, but fail to protect authenticity and integrity of the exchanged messages. Systems such JANUS encrypt URLs but do not provide confidentiality of data stream.

- Another crusial reason is that service providers gather identity information about users in order to provide services. A subject enjoys privacy when he can control, on his own, the dissemination of his private information. These technologies exploit confidentiality and encryption mechanisms only to control access to users personal data, but fail to provide anonymity, unlinkability to users.

- These technologies are more privacy-enhancing tools than models oriented to support global and anonymous online transactions. Thus, they do not integrate technologies, which help users explore and make use of the online transactions.

XML Digital Signatures is the encryption mechanism that is mostly used in the protocol, because this technique can protect users who request web services send and receive SOAP messages through insecure networks such as the Internet. The problem is that applications that communicate with external applications or users interchange sensitive data that may be read by unauthorized people while the messages with this data are in transit. XML Digital Signatures are introduced in this protocol due to the following factors:

- Messages may be captured while they are in transit, so there is the need to prevent unauthorized users from reading them by hiding the information of the message using encryption.

- The need to express encrypted elements in a standardized XML format to allow encrypted data to be nested within an XML message. Otherwise, different applications cannot interoperate.

- Different parts of a message may be intended for different recipients, and not all the information contained within a message should be available to all the recipients. Thus, recipients should be able to read only those parts of the message that are intended for them.

- For flexibility reasons, both symmetric and asymmetric encryption algorithms should be supported. Symmetric and asymmetric encryption use different algorithms to produce ciphertext. For example in symmetric encryption, the algorithm divides data into small chunks called blocks. It then switches the message form, changes the information in each block into numbers, compresses and expands the data, and runs those numbers through mathematical formulas that include the key. On the other hand an asymmetric encryption's algorithm, raises it to the power of another very large number, and then calculates the remainder after dividing it with a third very large number. Finally, the remainder number is converted back into text. Encryption programs can use the same algorithms differently, which is why the recipient needs to use the same application to decode the message that was used to encode it. When a user encrypts a message with a fast symmetric algorithm, then encrypts the secret key using asymmetric cryptography and then sends it to the recipient. Asymmetric cryptography is slow, but not too slow to encrypt such a small file as a symmetric encryption key. Upon receipt, the recipient can easily use his/her private asymmetric key to decrypt the symmetric key. Further that symmetric key can be used to quickly decrypt the message file.

- If a secret key is embedded in the message, it should be protected. The keys are the most important tools in keeping data secure. Keeping the private keys secure and large enough will make it very difficult to crack an encryption system. Otherwise, an attacker could read some messages.

  To protect the security of a message which carries sensitive data is to transform a message using some algorithm so that it can only be understood by legitimate receivers that possess a valid key. First, the data has to be serialized before encryption. The serialization process will convert the data

into octets. Then, this serialized data is encrypted using the chosen algorithm and the encryption key. The cipher data and the information of the encryption (algorithm, key, and other properties) are represented in XML format. XML Encryption supports both types of encryption: symmetric and asymmetric. The symmetric encryption algorithm uses a common key for both encryption and decryption. On the other hand, the asymmetric encryption algorithm uses a key pair (public key and private key). The sender encrypts a message using the receiver's public key, and the receiver uses its private key to decrypt the encrypted message. Thus, in both types of encryption, only recipients who possess the shared key or the private key that matches the public key used in the encryption process can read the encrypted message after decryption.

1.3.3 How the research is proceeding

The main topic that is tackled by this thesis is the need of some users to hide their personal information while interacting with cloud service providers. To overcome the problems of identity disclosure and profiling individuals, we suggest development of a model and the use of infomediary. Infomediary is a business entity supporting the development of anonymous business models. In current research on privacy is a protocol entity whose main role is to collect user's information, and secure users personal data from different service providers. I/M basic operation is based on taking users requests and finding the most suitable available service for them over the internet.

Infomediary is the intermediatory entity in the proposed model. It is the entity that manages the users' personal information, also requests for services and the offering services from different cloud service providers, and matches them, so users do not interact directly with cloud service providers but only through the infomediary.

The objective is protecting the privacy of individuals, while maintaining their unlinkability in multiple transactions with different cloud service providers over the

cloud. To achieve described above, we designed a stand alone application with the use of XML digital signature in the system architecture that protects the user privacy for unlinkability and the digital content in transactions.

1.3.4 Measuring framework progress

- The proposed approach provides secure transactions and confidentiality to users over the network and protects them from different type of attacks such as replay, spoofing or eavesdropping attacks and trace-back attacks. These attacks are forms of network attacks in which unauthorized users try to capture users data in online transmissions.

- Also offer protection to users hiding their personal information and protects from identity disclosure.

- It is vital to offer data integrity and authenticity, meaning no one is able to modify data except the legal users.

- It is significant the proposed protocol being scalable and suitable in order to operate in cloud environment.

1.4 Novel Features of the Thesis

This thesis proposes an approach for the unlinkability in the Single Sign On environment. The use of Unlinkability in multiple transactions in cloud environment with the development of a framework is an original idea as far as we know.
Also the major novel element that is proposed is the use of the Infomediary (I/M). Infomediary is a trusted entity in the proposed protocol that manages the users credentials and the offering services which the distributed CSPs offer, and matches

the users request for a service and the offered services. Substantially is one intermediary between the user and the CSPs that offers a secure communication channel over the cloud environment and protects users identity disclosure. The element of Infomediary (I/M) was first introduced in Gritzalis, Moulinos and Kostis research for developing a business model.

1.5 Outline of the Thesis

Chapter 2 describes the background information and related work to this research. The chapter provides the needed definitions and terminology to understand the rest of the thesis. It surveys the different and the most prevailing approaches, protocols to identity disclosure and unlinkability and gives some detail description of those systems. The background and related work relevant to a speciffic chapter are discussed in the corresponding chapter.

Chapter 3 presents the proposed privacy protocol. The chapter provides a detailed description of the architecture, and operation of the model. Sample scenarios are used to illustrate the operation and progress of the protocol.

Finally, **Chapter 4** summarizes the contributions of this thesis. It highlights the limitations and possible enhancements of this work; and concludes the thesis.

Chapter 2

Background and Related work

2.1 Introduction

The term Cloud is been used as a metaphor for the Internet. This term was originally derived from the description of network diagrams as a cloud, used to represent the transport of data which are carrier across the network.

The network or cloud has no borders. Cloud computing is an Internet-based computing, where users have direct access to distributed computers and share software data, and other resources in the network. Users now can gain access to other users distributed data from everywhere in the network. Globalization of computing assets may be the greatest contribution the cloud has made to date. For this reason, cloud computing is a very important subject. There are applications that support services, mail filtering services, storage services, security of users and so on.

In cloud computing many users have access to distributed different systems, so there is the case of many users requesting many services from different Cloud Service Providers (CSP). Each CSP provides an authentication mechanism based upon their own system requirements so that users can authenticate themselves to the system.

User authentication is the basis for access control, keeping the intruders / attackers out while allowing authorized users to access to a service in the network. In the cloud environment, authentication and access control are more important than ever since the cloud and all of its data are accessible to anyone over the Internet. The authentication process can provide strong security and privacy than username and passwords.

Authentication needs are increased today due to many types of application that are hosted in the cloud, such as personal, financial, medical information. Also software applications that operate on cloud require access control mechanisms. Thus the need for higher authentication in cloud computing nowadays is an important issue. Cloud computing has many cloud service providers and many users who request multiple and different services in the cloud. Also because impersonation is easy in the cloud there is the need for the user to authenticate him only once and without giving much information about him in order to protect his privacy and anonymity. The above goal can be achieved with the use of anonymity and unlinkability techniques and protocols.

Nowadays, due to the heterogeneity of systems, vendors face the problem of how to provide single logon to their use community. Both users and administrators are affected by this problem. Users need to remember more than one id or password pair. Administrators face the problem of maintaining consistency and security for their varied user community across multiple platforms and policies. So there is the need of the use of Single sign-on (SSO) mechanism. Single sign-on (SSO) is mechanism where a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords, every time the user requests a service. Unlinkability is defined as a property which ensures that a user may make multiple uses of resources or services without others being able to link those uses together. In unlinkability of two or more items is defined as a property that, within the system, those items are no more and no less related than they are related concerning the a priori knowledge. This thesis will examine the available unlinkability protocols and mechanisms. Also it will analyze the existing techniques and how this problem was faced in other environments and propose a suitable unlinkability protocol and a framework for the cloud SSO environment.

2.2 Cloud computing

Internet is not only a communication medium but, due to the reliability, affordability, and ubiquitous broadband access that it offers, has become a powerful computing platform. Rather than running software and managing data on a desktop computer or server, users are able to execute applications and access data on demand from the cloud (the network / Internet), anywhere, anytime across the world. This is called "cloud computing". Examples of cloud computing applications are Amazon's Simple Storage Service (S3), Elastic Computing Cloud (EC2) for storing photos on Smugmug an on line photo service, and Google Apps for word-processing.

Cloud services make it easier for users to access their personal information from databases and make it available to services distributed across Internet. The availability of such information in the cloud is significant to provide better services to users and to authenticate users in case of services sensitive with respect to privacy and security.

Users have typically to authenticate themselves every time they use a new cloud service, usually by filling out an online form and providing sensitive personal information (e.g., name, home address, credit card number, phone number, etc.). This leaves a trail of personal information that, if is not properly protected, it may be misused. An important requirement is that users of cloud services must have control over the disclosure and use of their personal information, in order to minimize the risk of identity theft and fraud.

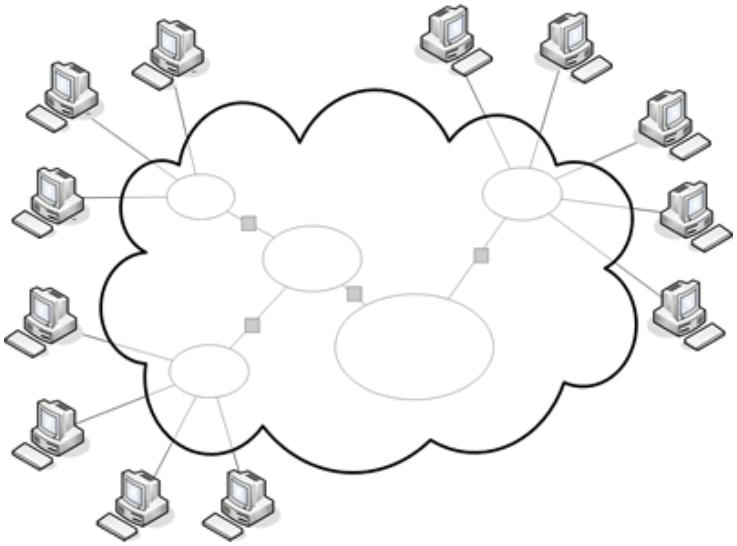


Fig. 2.1: Cloud

The problem is dealing with security and privacy in cloud environments. This issue makes it a great research area importance and challenge. Cloud computing has emerged as a powerful and cost-effective paradigm for provisioning computing power to users. In the cloud computing, users use an intranet or the Internet to access a shared computing cloud that consists of a large number (thousands or tens of thousands) of interconnected service providers organized as one or more clusters.

Cloud computing represents an important step of computation may some day be organized as a public utility service. It embraces concepts such as software as a service and platform as a service, which incorporate services for workflow facilities for application design and development, deployment and hosting services, data integration, and network-based access to and management of software. Customers of clouds, much like customers of utility companies, can subscribe to different services at different service levels to guarantee the desired quality of service.

Though not everyone agrees on the exact definition of cloud computing, most agree the vision encompasses a general shift of computer processing, storage, and software delivery away from the desktop and local servers, across the network, and into data centres hosted by large infrastructure companies such as Amazon, Google, Yahoo,

Microsoft, or Sun. cloud computing is reminiscent of the application service provider (ASP) and database-as-a-service (DaaS) paradigms. In practice, cloud computing platforms, like those offered by Amazon Web Services, ATT's Synaptic Hosting, AppNexus, GoGrid, Rackspace Cloud Hosting, and to an extent, the HP/Yahoo/Intel Cloud Computing Testbed, and the IBM/Google cloud initiative, work differently than ASPs and DaaS.

2.3 Cloud Characteristics

Cloud computing has generated significant interest, but it's still an evolving area. Essentially, it aims to consolidate the economic utility model with the evolutionary development of many existing approaches and computing technologies, including distributed services, applications, and information infrastructures consisting of pools of computers, networks, and storage resources. Cloud computing can significantly reduce costs through optimization and increased operating and economic efficiencies. Furthermore, cloud computing could enhance collaboration, agility, and scale, thus enabling a truly global computing model over the Internet infrastructure. However, without appropriate security and privacy solutions designed for clouds, this could become a huge failure. The main characteristics of cloud computing include on-demand self-service, ubiquitous network access, location-independent resource pooling, rapid elasticity, and measured service.

- On-demand self service, allows business units to get the computing resources they need without having to go through IT for equipment. For example a user can quickly and easily configure the computing resources he needs all by himself, without filling out forms or emailing the service provider.

- Broad network access enables applications to be built in ways that align with how businesses operate today - mobile, multi-device, etc. For example users can access these resources from anywhere they can access the Internet, and can access them from a browser, from a desktop with applications designed to work with them, or from a

mobile device. One of the most popular application models is a mobile application that communicates with a cloud-based back end.

• Resource pooling, allows for pooling of computing resources are to serve multiple consumers. The cloud service provider, whether it's a certain server or his own IT department, manages all of its cloud's physical resources; creates a pool of virtual processor, storage, and network resources; and securely allocates them between all of its customers.

• Rapid elasticity allows quick scalability or downsizing of resources depending on demand. The capacity can grow and shrink (processing power, storage, network) very quickly, in minutes or hours. Self-service and resource pooling are what make rapid elasticity possible. Triggered by a customer request, the service provider can automatically allocate more or less resources from the available pool.

• Shared Infrastructure Uses a virtualized software model, enabling the sharing of physical services, storage, and networking capabilities. The cloud infrastructure, regardless of deployment model, seeks to make the most of the available infrastructure across a number of users.

• Dynamic Provisioning Allows for the provision of services based on current demand requirements. This is done automatically using software automation, enabling the expansion and contraction of service capability, as needed. This dynamic scaling needs to be done while maintaining high levels of reliability and security.

• Network Access Needs to be accessed across the internet from a broad range of devices such as PCs, laptops, and mobile devices, using standards-based APIs. Deployments of services in the cloud include everything from using business applications to the latest application on the newest smartphones.

• Managed Metering Uses metering for managing and optimizing the service and to provide reporting and billing information. In this way, consumers are billed for services according to how much they have actually used during the billing period. In

short, cloud computing allows for the sharing and scalable deployment of services, as needed, from almost any location, and for which the customer can be billed based on actual usage.

• Compute power is elastic, but only if workload is parallelizable. One of the oft-cited advantages of cloud computing is its elasticity in the face of changing conditions. For example, during seasonal or unexpected spikes in demand for a product retailed by an e-commerce company, or during an exponential growth phase for a social networking Website, additional computational resources can be allocated on the fly to handle the increased demand in mere minutes (instead of the many days it can take to procure the space and capital equipment needed to expand the computational resources in-house). Similarly, in this environment, one only pays for what one needs, so increased resources can be obtained to handle spikes in load and then released once the spike has subsided. However, getting additional computational resources is not simple as, more powerful machine on the fly (with commensurate increase s in CPUs, memory, and local storage); rather, the additional resources are typically obtained by allocating additional server instances to a task. For example, Amazon's Elastic Compute Cloud (EC2) apportions computing resources in small, large, and extra large virtual private server instances, the largest of which contains no more than four cores. If an application is unable to take advantage of the additional server instances by offloading some of its required work to the new instances which run in parallel with the old instances, then having the additional server instances available will not be much help. In general, applications designed to run on top of a shared-nothing architecture where a set of independent machines accomplish a task with minimal resource overlap are well suited for such an environment. Some cloud computing products, such as Google's App Engine, provide not only a cloud computing infrastructure, but also a complete software stack with a restricted API so that software developers are forced to write programs that can run in a shared-nothing environment and thus facilitate elastic scaling.

• Data is replicated, often across large geographic distances Large cloud computing providers with data centres spread throughout the world have the ability to provide high levels of fault tolerance by replicating data across large geographic distances.

Amazon's S3 cloud storage service replicates data across 'regions' and 'availability zones' so that data and applications can persist even in failure.

2.4 Service Models

Once a cloud is established, the cloud computing services are deployed in terms of business models~\citeasnoun{dialogic}, but they can differ depending on requirements. The primary service models being deployed (see Figure 2.2) are commonly known as:

- Software as a Service (SaaS) Consumers purchase the ability to access and use an application or service that is hosted in the cloud. In this layer the necessary information for the interaction between the consumer and the service is hosted as part of the service in the cloud. In SaaS the end user application is delivered as a service. Platform and infrastructure is abstracted, and can be deployed and managed with less effort.

- Platform as a Service (PaaS) Consumers purchase access to the platforms, enabling them to deploy their own software and applications in the cloud. The operating systems and network access are not managed by the consumer, and there might be constraints as to which applications can be deployed. PaaS is an application platform onto which custom applications and services can be deployed. Can be built and deployed more inexpensively, although services need to be supported and managed.

- Infrastructure as a Service (IaaS) Consumers control and manage the systems in terms of the operating systems, applications, storage, and network connectivity, but do not control the cloud infrastructure. Physical infrastructure is abstracted to provide computing, storage, and networking as a service, avoiding the expense and need for dedicated systems.

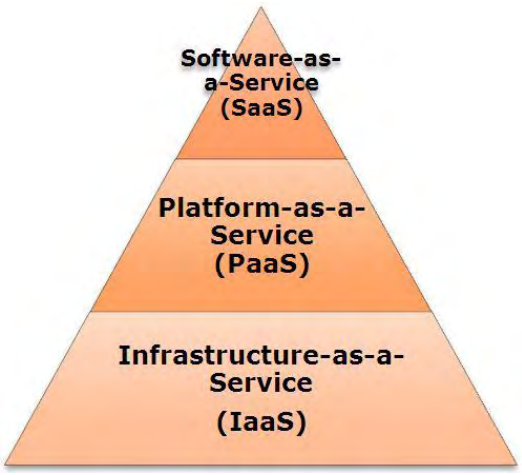


Fig. 2.2: Service Models

2.4.1 Benefits of Cloud Computing

The following are some of the benefits of cloud computing based services and applications:

- Cost Savings Companies can reduce their capital expenditures and use operational expenditures for increasing their computing capabilities. This is a lower barrier to entry and also requires fewer IT resources to provide system support.
- Scalability and Flexibility Companies can start with a small deployment and grow to a large deployment fairly rapidly, and then scale back if necessary. Also, the flexibility of cloud computing allows companies to use extra resources at peak times, enabling them to satisfy consumer demands.

- Reliability Services using multiple redundant sites can support business continuity and disaster recovery.

- Maintenance Cloud service providers do the system maintenance, and access is through APIs that do not require application installations onto PCs, thus further reducing maintenance requirements.

- Mobile Accessible Mobile workers have increased productivity due to systems accessible in an infrastructure available from anywhere ~\cite{dialogic}.

2.5 Challenges in Cloud Computing

As it was mentioned the security and privacy risks in cloud computing and developing efficient and effective solutions are critical for its success. Although clouds allow customers to avoid start-up costs, reduce operating costs, and increase their agility by immediately acquiring services and infrastructural resources when needed, their unique architectural features also raise various security and privacy concerns.

Cloud computing provides access to data, but the challenge is to ensure that only authorized entities can gain access to it. When we use cloud environments, we rely on third parties to make decisions about our data and platforms in ways never seen before in computing. It's critical to have appropriate mechanisms to prevent cloud providers from using customers' data in a way that hasn't been agreed upon. It seems unlikely that any technical means could completely prevent cloud providers from abusing customer data in all cases, so we need a combination of technical and nontechnical means to achieve this. Clients need to have significant trust in their provider's technical competence and economic stability. Cloud providers and customers must share the responsibility or security and privacy in cloud computing environments, but sharing levels will differ for different delivery models.

The following are some of the notable challenges associated with cloud computing, and although some of these may cause a slowdown when delivering more services in the cloud, most also can provide opportunities, if resolved with due care and attention in the planning stages.

- Security and Privacy Perhaps two of the more critical issues surrounding cloud computing relate to storing and securing data, and monitoring the use of the cloud by the service providers. These issues are generally attributed to slowing

the deployment of cloud services. These challenges can be addressed, for example, by storing the information internal to the organization, but allowing it to be used in the cloud. For this to occur, though, the security mechanisms between organization and the cloud need to be robust and a Hybrid cloud could support such a deployment.

- Lack of Standards Clouds have documented interfaces; however, no standards are associated with these, and thus it is unlikely that most clouds will be interoperable.

- Access Control and Accounting Heterogeneity and diversity of services, as well as the domains' diverse access requirements in cloud computing environments, demand fine-grained access control policies. In particular, access control services should be flexible enough to capture dynamic, context, or attribute- or credential-based access requirements and to enforce security principles.

  It's important that the access control system employed in clouds is easily managed and its privilege distribution is administered efficiently. Also it must be ensure that cloud delivery models provide generic access control interfaces for proper interoperability, which demands a policy-neutral access control specification and enforcement framework that can be used to address cross-domain access issues.The utility model of clouds demands proper accounting of user and service activities that generates privacy issues because customers might not want to let a provider maintain such detailed accounting records other than for billing purposes. The outsourcing and multi-tenancy aspects of clouds could accelerate customers' fears about their personal data, for example accounting logs. Hence, utilizing a privacy-aware framework for access control and accounting services is crucial.

- Data is stored at an untrusted host. Although it may not seem cloud computing host to violate the privacy of its customers and access data without permission, such a possibility nevertheless it occurs risks of disclosure. The cloud

computing uses computing and storage resources are being delivered from a celestial location, the fact is that the data is physically located in a particular distributed service provider (e.g. another country and is subject to local rules and regulations). Most cloud computing vendors give the customer little control over where data is stored such as, Amazon S3 which only allows a customer to choose between US and EU data storage options. The data should be encrypted using a key not located at the host, the data should not be accessed by a third party without the customer's knowledge.

- Continuously Evolving User requirements are continuously evolving, as are the requirements for interfaces, networking, and storage. This means that a cloud, especially a public one, does not remain static and is also continuously evolving ~\cite{dialogic} ~\cite{onLineSource}.

2.6 Single Sign-On

The real problem is that each user does not have the ability to access many service providers simultaneously, without authenticate themselves every time they access another service. Access to simultaneous services and sharing of resources are, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.

Historically each system provided an authentication mechanism based upon its own system requirements which were adequate when users only needed to authenticate to one system. Today due to the heterogeneity of systems, vendors face the problem of how to provide single sign on to their use community. Single sign-on mechanisms already exist, and they are widely used, like the Active Directory for Microsoft Windows or eDirectory for UNIX systems.
Single sign-on (SSO) is mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords. Single sign-on

reduces human error, a major component of systems failure and is therefore highly desirable but difficult to implement.

Single Sign On is an advantageous technique. In a typical client-server environment today the average user signs on many services, workstations, servers and applications. Using Single Sign On users do not need to remember more than one id or password pair. Moreover in cloud computing where there are many threats due to the large amount of users, the use of many passwords and many times the same for different applications, leads to a big security issue. With Single Sign On administrators do not face the problem of maintaining consistency and security for their varied user community across multiple platforms and policies. They do not have to manage and change a large number of passwords. Also is a system of comprising multiple web applications/services and a single administrative domain that holds Identity Provider service. All the applications/services are registered in the Identity Provider service database. The system provides SSO to users of the web applications/services.

Three logical components:

- The user is a person who assumes a particular digital identity to interact with an online network application. The user agent is a browser or other software application that runs on anything from a PC to a mobile phone to a medical device. A user's online interactions always take place through an agent, which can passively allow identity information flow or actively mediate it.

- The service provider (SP) site is a Web application such as an expense-reporting application or an open source community that offloads authentication to a third party, which might also send the SP some user attributes. Because the SP relies on external information, it's often called a relying party (RP).

- The identity provider (IdP) is a Web site that users log in to and that sometimes stores attributes of common interest to share with various SPs.

2.6.1 The Basic operation of Single Sign-On

The main problem with using most common security architectures with Web services is that the security infrastructure is very distributed and these architectures usually require that key security features and algorithms are implemented by all parts of the system. The main weakness of all security systems is that the level of security of the whole system is the same as the level of the security of its weakest part. This obviously leads to inflexibility because we have to either avoid certain technologies or compromise the security of a whole system. Avoiding technologies is generally impractical and negates the very reason Web services are so popular: they are able to bridge between any technology. One possible solution to this problem is the single sign-on (SSO) architecture.

The basic idea of the single sign-on security architecture is to shift the complexity of the security architecture to the so-called SSO service and thus release other parts of the system from certain security obligations. In the SSO architecture, all security alghorithms are found in the single SSO server which acts as the single and only authentication point for a defined domain. Thus, there is a second benefit to an SSO approach to autnentication/registration: a user has to sign-on only once, even though he may be interacting with many different secure elements within a given domain. The SSO server, which can itself be a Web service, acts as the wrapper around the existing security infrastructure that exports various security features like authentication and authorization.

In the simple SSO scenario the authenticated party first calls the SSO server and requests the authentication token that identifies it in the particular domain. In order to get the token, it must first provide the correct authentication credentials. There are various forms for those credentials. For example, it could be a simple username/password, email or certificate. The SSO server performs a verification of the users' credentials using the underlying security infrastructure, and only then issues a ticket that the calling application uses for authentication against other applications. In the simple scenario, the token doesn't impose any specific information, it is simply the unique identification of the user. After the invoked application receives the token,

it validates it by passing it to the SSO server that then performs the actual checks. However when the SAML is used the token itself contains valuable security information that allows validation without having to call the SSO server each time. The token contains the authentication or authorization information. This information is signed by the SSO server, so the recipient trusts this server, it doesn't have to do any further verification.

The main advantages of the SSO approach are the encapsulation of the underlying security infrastructure in the SSO server. Implementation, deployment and maintenance are much easier as all communicating parties in the distributed system don't need to individually implement all of the security features and mechanisms. The SOAP interface to the SSO server makes the SSO architecture universal. The SSO is itself a Web service. The SSO server enhances security of the whole system as the security credentials don't need to be passed around.

2.7 Related work

Many Protocols and mechanisms were proposed and developed over the years in order to provide unlinkability and protect users from different types of attacks which aimed in their personal information, such as credit cards numbers, emails, log-in usernames and passwords etc. for malicious purposes. As was mentioned earlier different CSPs share data and users information in order to provide them services, or access to certain services so it is very common and possible the disclosure of their identity full or partial.

The issue of unlinkability also was introduced by Elisa Bertino, Federica Paci, Rodolfo Ferrini, Ning Shang who proposed a privacy-preserving multi-factor identity attribute verification protocol supporting a matching technique based on look-up tables, dictionaries, and ontology mapping techniques to match cloud service providers and clients vocabularies. The protocol that they proposed uses an aggregate zero knowledge proofs of knowledge (AgZKPK) cryptographic protocol to allow

clients to prove with a single interactive proof the knowledge of multiple identity attributes without the need to provide them in clear. Their approach allows the user to use pseudonyms when interacting with the CSPs, if the CSP policies allow the use of pseudonyms and the user is interested in preserving his/her anonymity. In the case that multiple transactions are carried out by the same user with the same CSP, or this CSP can determine that they are from the same user, even if the CSP does not know who this user is nor the identity attributes of the user. Different CSPs may also collude and determine a profile of the transactions carried out by the same user. Such information when combined with other available information about the user may lead to revelling the identity of the user or values of some of his/her identity attributes, thus leading to privacy breaches. We are going to address this problem by investigating techniques that maintain unlinkability among multiple transactions carried out by the same user with the same or different CSPs.

Zero-Knowledge Proof Protocols

The use of Zero-knowledge proof protocols is another technique that preserves the privacy of the user identity attributes by assuring that the participants CSP do not learn the values of these attributes, thus the CSP has no knowledge of the user's information. It is a cryptographic protocol between two parties. In that interactive method one party wants to prove to another that a statement is true, without revealing any other information. A zero-knowledge proof is a way that a prover can prove possession of a certain piece of information to a verifier without revealing it. Zero-knowledge proofs can be applied where secret knowledge is sensitive to reveal and it needs to be verified, such as in Key authentication, PIN numbers, Smart cards.

Zero-knowledge proof has three main properties of identity:
Completeness: The honest prover convinces the honest verifier that the secret statement is true.
Soundness: Cheating prover can't convince the honest verifier that a statement is true (if the statement is really false).
Zero-knowledge: Cheating verifier can't get anything other than prover's public data sent from the honest prover.

Anonymizer

Anonymizer is a widely used Web anonymity tool. It is a service that submits HTTP requests on behalf of its users. The main concept introduced by the Anonymizer tools is the existence of a third-party Web site. The third party acts as a middle layer between the user and the site to be visited. When the user wishes to visit a specific Web page, instead of establishing a direct link to the required Web server, he/she does so through the Anonymizer Web site. Having established the connection, Anonymizer forwards the information received from the desired Web site back to the user.
Anonymizer is extremely easy to use but it has some vulnerabilities. The most important is that users should trust the Anonymizer service because the Anonymizer server monitors all the Web sites visited by the user, thus collecting information about user's behaviour.

Mix –Nets

Mix networks, protect the unlinkability of the messages that enter the network with respect to their recipients, the messages that leave the network with respect to their senders, and, hence, the identifiers of communicating parties with respect to communication sessions. By using only one mix, malicious users cannot identify the relation between sender and recipient but the mix and the sender can. A chain of independent mix stations is used for improving security.
Using a chain of mixes an attacker, who has access to all communication lines, can only trace a message through the mix network, if the user has the cooperation of all mix nodes on the path or if he/she can break the cryptographic operations that mix stations use. Ensuring the trustworthiness of at least one mix, unlinkability of sender and recipient is ensured.

Hordes

Hordes is a protocol that engages multiple proxies for routing and it uses multicast services for routing the reply to the initiator in an anonymous way. Hordes architecture is based on several proxies. Hordes has been designed for gathering many participants in the same reception group. This is succeeded because Hordes takes

advantage of the anonymity characteristics and performance benefits inherent in the IP multicast routing. Hordes utilises multicast communication for the reverse path of anonymous connections, achieving not only anonymity but also sender unlinkability and unobservability.

GAP

GNUnet's Anonymity Protocol GAP is a recently presented protocol claiming to achieve anonymous data transfers. GNUnet is a peer-to-peer network and its framework provides peer discovery, link encryption and message batching. GAP has been designed for securing GNUnet. It aims to hide the identity of an initiator and a responder from all other entities, including GNUnet routers, active and passive adversaries and the responder or initiator respectively. The communication between all network nodes is confidential; no host outside the network can observe the actual data travelling across the network. Data types are not identifiable since all packets are padded in order to have identical size. However, GAP's greater vulnerability is that it has been customised to the functionality of a peer-to-peer network.

Secure Electronic Transactions (SET)

The extended use of modern IT systems and communication networks in daily activities such as e-banking, shopping, communicating between users, collaborating with government authorities, etc, sets additional requirements for protecting the electronic privacy of participants. The Secure Electronic Transactions Protocol (SET) is a set of security protocols and ensures the privacy and protection requirements in electronic transactions. The unlinkabilty in multiple transactions is a critical issue, because in transactions information about the buyer must not be revealed among the participant users.

SET is an open encryption and security specification designed to protect credit card transactions in the Internet. It was developed by Visa and MasterCard. It relies on two different encryption mechanisms, and an authentication mechanism. SET uses symmetric encryption, in the form of the Data Encryption Standard (DES) and asymmetric or public-key encryption.

The SET protocol provides three services:
1) Provides a secure communication channel between all parties that are participate in the transaction. All sensitive information sent between the three parties is encrypted.
2) All three parties are required to authenticate themselves with certificates.
3) The information is available only when is needed. The merchant never sees the customer's card number in plaintext.

SET protocol consists of four agents. These agents are Cardholder, Merchant, Payment Gateway and Certificate Authority. Each one has a specific role in the transaction process. The participants need to know the basis and do not need to gather or disclose other information about the other participants in the transaction. For this reason SET offers the Dual Signature.

The goal of Dual Signature is to link two messages that are intended for two different recipients. Also it links the different recipients without disclosure their identity to each other. For example the customer wants to send an order information to the merchant and payment information to the bank. Merchant does not need to know customer's credit card number and the bank does not need to know the order. In the process of Dual Signature customer takes the hash of the Payment Information and the hash of Order Information. These two hashes are concatenated and the hash of the result is taken. Then the customer encrypts the final hash with his own private signature and creates a Dual Signature.

Blind Signatures is another technique for protecting electronic transactions. User's information must not be acquaintances in all the participants, only information that are needed for the transaction. For example a bank signs the user in every transaction but this seems to immediately violate anonymity because the Bank will be able to recognize the signatures it computed when the payment information come back and are deposited. But this can be solved by so called blind signature schemes, where the Bank does a protocol with the User, and as a result the User obtains the Bank's signature on some message, nevertheless the Bank has no information about this message.

PPay is another P2P payment protocol, it is a micro-payments, offline protocol that uses floating, self-managed coins. In this protocol security is sacrificed to reduce the broker's involvement and as a result the brokers load. Though a significant improvement of the systems performance is achieved this protocol is inappropriate for medium and large payments as the proposed peer-to-peer protocol. The secure version of PPay is called WhoPay.

WhoPay provides a secure infrastructure for electronic commerce and anonymity between the parties involved in a transaction, though it requires a big database for storing the scripts and does not consider that the P2P environment is an environment of unstable connectivity. PPay's and WhoPay's scalability is based on the domination of the system by the transactions of transfer and renewal of scrips. These transactions require the presence not only of the two parties doing business but also of a third party that "substitutes" the broker. If this party is offline the broker is the one that has to take part to the transaction, so in this case the broker's load is increased.

Despoina Palaka, Petros Daras Kosmas Petridis, and Michael G. Strintzis proposed an electronic payment protocol, three parties are involved in this protocol: the customer (who makes the actual payment), the merchant (who receives the payment) and the acquirer gateway (that acts as an intermediary between the electronic payment world and the existing payment infrastructure and authorizes transactions by using the latter). The acquirer gateway is addressed as "the broker". This broker is used to secure the transactions and to enable a trust relationship between the parties, introduces the problem of "single point failure". This problem is typical in any client/server payment system, but the role of the broker is essential for security and financial reasons and it cannot be omitted. In this protocol the broker's participation in the transactions has been minimized in order to minimize the effect of the problem that it introduces. We consider that approach will help us develop a framework that will provide unlinkability and security in online transactions and communication between peers.

2.7.1 Other Approaches

Bao and Deng were concerned about the anonymous transaction and commensurate with the general problem of the Private Information Retrieval (PIR). They introduced the system that allows a customer to disclose identity information to the web site in exchange for a digital item, but prevents the web site from learning which specific item a customer buys. The potential customer pay for the desired content on the Internet but also his purchasing information is hidden from the web site. Therefore, it is difficult to get the necessary sales information for service provider's business activities. To do this, the merchant generates the secure package including item information, encrypted item and an encryption key. The customer downloads this secure package and gets an encryption key for item decryption. The user must obtain the key from the transaction server which is independent entity in the system architecture. However, the static number of downloaded item can be gathered at the service provider. The main weakness of this system is that identity information must be given to the service provider.

Gritzalis, Moulinos and Kostis introduce a new infomediaries-based, privacy enhancing business model, which is capable of providing anonymity, privacy and security, to customers and vendors of e-commerce. In order to identify customer preferences, and customize products and services, marketers are looking for new ways of capturing, processing and exchanging customer data. They collect data every time customers visit their web sites; they also use numerous techniques to analyze that data and create mature user profiles. A user profile includes personal data, which may identify in a unique way a customer consuming behavior. Such a collection and processing of personal data may lead to privacy violation.

Using this model, customers of e-commerce can buy goods or services, without revealing their real identity or preferences to vendors, and vendors can sell or advertise goods or services without violating the privacy of their customers. This system is based on the informediaries. The informediary (I/M) is a business entity supporting the development of anonymous business models and its role is to gather user information, and deals with products and services on behalf of them. The informediaries provide the user information that is not enough to identify each user to

suppliers and maximize the value of customers profile while they prevent suppliers or commercial web servers from collecting user profile. Therefore, the use of I/M enables customers to increase their bargain capability without revealing personal data and, at the same time, enables vendors to promote products and service without violating customers' privacy.

Enzman, Kunz and Schneider, proposed a system that prevents the vendor (or supplier) from linking the user information which is gathered while searching with identifying information. In order to do this, the system requires asymmetric algorithm for data encryption by using public key of the vendor. If the user wants to buy some products or services, the user generates agents which contain the desired product information, and sends it to the base station which is in the middle of communication between a user and vendors. That is, all agents from the user are sent to the destined vendor via the base station. The base station dispatches these agents and plays a role of proxy. Thus, the vendor cannot gather users' IP addresses and cookies for linking the received order. In these approaches generate the pseudonyms for customers and employ the TTP between the customer and the service provider. The service provider is able to get the necessary information related with its customers, but the customer must trust the TTP.

Seok-kyu Kang Tomoyuki Asano Kwangjo Kim system consists of three components: Customer, Service Provider (SP) and Payment Server (PS). The customer blinds his desired Content's sample Information (CI), and Payment Server (PS) generates a signature for blinded CI, PS and cannot learn which content the customer intends to purchase. Customer reveals information only for the payment process. SP provides digital contents to the customer and this participant should have user-profile information for its marketing activities. However, SP is not permitted to get customers' identities. PS is an entity performing the payment process for the customer. In this approach, PS cannot know about the customer's purchasing information, such as what the customer bought or which content he/she expects to buy. Simply, PS only deals with the payment information received from a customer and there is no way for PS to know which customer intends to purchase what kinds of contents. PS does not reject SP's request to transfer money after finishing the transaction of the customer. In conclusion communication between SP and the

customer is achieved with anonymity and SP cannot trace the specific customer's identity.

OpenID Brad Fitzpatrick developed OpenID for use in the LiveJournal online community as a lightweight, decentralized way to authenticate commenter and avoid blog- comment spam. OpenID is a popular federated login system that focuses on for a proof of concept implementation. In OpenID, users can claim identifiers in the form of URIs. To login to a website that supports OpenID, the user enters his OpenID URI and the relying party redirects him to his identity provider's page. The identity provider authenticates the user through its choice of authentication system (e.g. passwords, smart cards, etc.) and then returns the user to the relying party with either a positive or negative assertion that the user owns the claimed identifier. If the relying party receives a positive assertion from the identity provider, it may allow the user to enter the site under the name of the claimed identifier.
With the advent of OpenID 2.0, the protocol also began to support the concept of directed identity or private digital addresses through a new feature called identifier select. This allows the user to just specify the URI of his identity provider instead of claiming a personal identifier when logging into a website. The site then redirects the user to the identity provider as before, but the identity provider now has the opportunity to select an identifier for the user. Authenticating successfully the user, the identity provider returns the selected identifier to the site. This allows the identity provider more flexibility in selecting identifiers for its users. On the other hand OpenID enables a third party to gather information about user behaviour over multiple web sites without user consent more easily than using "Web Bugs". Web Bugs are invisible third party images added to a web page so that the third party receives notice of the page viewing event. In order to solve the problem, authentication systems should have the property that it is difficult to determine whether two transactions in distinct service providers are related to the same user or not (weak-unlinkability). There are some authentication systems which satisfy weak-unlinkability, such as Janus, anonymous credentials, and authentication systems based on group signatures.

PseudoID is designed to be a one way, consistent, and unlinkable federated login system. It consists of a token service used during setup, and a private identity provider used for sign on. The user has an account with the token service, which may be a

persistent, real identity like an email address. During setup, the user logs on to the token service using a familiar authentication scheme, such as entering a user name and password. The user then requests an access token from the token service that is bound to a desired pseudonym. When logging into a relying party, the user presents this token to an identity provider. The identity provider will verify the authenticity of the token and return the user's pseudonym to the relying party. To provide unlinkability, the access tokens must be generated such that even if both the token service and identity provider are compromised, the user's real identity with the token service cannot be linked to their pseudonyms on different relying parties. PseudoID achieves this property by using blind signatures.

CardSpace, was designed by Microsoft, is based on WS-Federation. CardSpace focuses on maximizing the ease of use and the individuals' control over their identities. This is done by allowing the identities, called InfoCards, to be managed at the individuals' machines. Info cards do not store identity information. Instead, these cards point to the identity providers from which the identity information can be requested. For a service provider to access identity information, the corresponding info card must be handed to the identity provider. Thus, individuals need not to worry about protecting their identities, while at the same time, they control which service providers get to use which info cards, and at which conditions.

Lucent Personalized Web Assistant (LPWA) is used at a server that is remote from the user application like the Anonymizer. It enables users to browse web sites using aliases that are secure, consistent and pseudonymous. Each alias presents a different persona - personal information, such as user name, password and e-mail address to each web site. The personas for different web sites belong to the same user and they are independent and unrelated among them. To provide anonymous personalized service for user, LPWA has three functional components: Persona generator, Browsing proxy and E-mail forwarder. The generated alias in the Persona generator is used for a specific web site where a user wants to visit anonymously. The E-mail forwarder sends mail with alias e-mail address to the corresponding user. First of all, these aliases have to satisfy anonymity, consistency and uniqueness properties. To generate different aliases and meet those alias properties, LPWA employs the proxy

using a collision resistant hash function called Janus function. The alias is computed in this function which takes user-ID and secret phases as an input.

JANUS is a cryptographic engine that helps clients establishing and maintaining secure and pseudonymous relationship with a server. JANUS is hiding the identity of recipients but not the identity of the senders. It employs a proxy, like an Anonymizer server, for the web browser's anonymity and encrypts the web server's URL with asymmetric encryption algorithm. The URL of web server is opened to the public encrypted. The sender requests the access to the server through the JANUS system with an encrypted URL. After JANUS system decrypts the server's encrypted URL with its private key, after the user is able to access the desired web site via JANUS. In this case, the unlinkability between sender and receiver is provided. However, like an Anonymizer, the JANUS system knows all information and both participants in the interaction also have to trust the JANUS system. Moreover, the system only encrypted URLs but does not provide the confidentiality of data stream on the network.

Onion Routing provides anonymous connections using different layers of encryption. It is more robust than single-proxy systems for anonymous communication. In Onion Routing, a series of proxies communicating over encrypted channels cooperate to forward data to a responder. Data is wrapped in a series of encrypted layers that are peeled at a series of proxies (onion routers) along a path towards the responder.

Each onion router knows the identities and public keys of each other onion router. The initiator begins by choosing a route through the other onion routers to the responder. For each onion router on the path, the initiator constructs a layer of a connection setup packet consisting of the IP address of the next onion router, the encryption key seed information shared with the next onion router k, and the successor's layer. The inner-most layer of the onion contains the identity of the responder and the data to be sent. Each layer is encrypted with the public key of the corresponding router. Each onion router pair uses a locally unique anonymous connection identifier so that subsequent communication does not require sending another onion. As the packet is forwarded through the path of onion routers, the layers are peeled. When the packet reaches the last onion router in the path, the data is forwarded directly to the responder. All requests from the initiator are sent along the same path of onion routers. Replies are

sent to the last onion router on the path, which in turn forwards the data along the reverse path of onion routers towards the initiator. In implementation, Onion Routing is not typically deployed at every host. Instead, a number of dedicated onion routers are available for use, and an initiator must connect to one of these to contact the receiver. The first onion router thus knows all initiators it is servicing. Should an onion router be corrupted, all initiators that use that router could be exposed.

In Federated Identity Management Systems (FIMSs), identity providers may supply individuals with anonymous credentials. Anonymous credentials allow individuals to interact with service providers in an anonymous fashion. However, since the pseudonyms in these credentials do not change, the actions of an individual are linkable to each other. To overcome the linkability of the actions of an individual, Anonymous Credential Systems (ACS) allow individuals to prove the possession of credentials, without showing them to organizations. For example, this enables an individual, to visit a service provider, multiple times, while that provider is unable to link the individual's actions.

U-Prove and Idemix are examples of anonymous credential systems. These systems utilize zero-knowledge proofs to enable individuals convince Service Providers (SPs) that those individuals possess credentials, granted by IPs, without disclosing the actual credentials. Zero-knowledge property allows a prover, who knows a correct solution for a problem, to convince a verifier that it knows that correct solution. The verifier learns nothing beyond the solution's correctness. Originally developed by Credentica, U-Prove has been recently acquired by Microsoft. U-Prove focuses on security and privacy aspects of identity management. Idemix, developed at IBM, not only targets privacy, but also tackles the problem individuals sharing their credentials. Idemix achieves multi-show unlinkability of credentials; that is, an individual may interact with same SP multiple times, without the SP being able to link these interactions together.

Prime is a project funded by the European Union and several corporations to support privacy enhanced identity management. European regulations regarding privacy are the core requirements being incorporated into Prime. Prime uses Idemix protocols for

issuing and verifying credentials. Whenever an individual desires to authenticate at an SP, the individual uses a credential obtained previously from his IP. The system uses zero-knowledge proofs to convince the SP that the individual has a valid credential from the IP. Note that identity providers are not required to be online, while individuals are interacting with service providers.

Gennaro and Rohatgi describe two methods for authenticating streams using one-way functions. The on-line scheme uses one-time signatures. Each block of the stream contains a public key, and is signed with the private key corresponding to the public key contained in the previous block. The first block carries a conventional asymmetric signature. One-time signatures are large, so the on-line scheme has a considerable bandwidth overhead.

Guy Fawkes protocol also uses chained hashes. The sender does not need to know the entire stream in advance, but each block must be known before the previous block is sent. Each block Ai carries a preimage Xi-1 and a hash H(Xi) that are used to verify the previous block, and a hash H(Ai+1,H(Xi+1),Xi) that commits to the contents of the next block. The first block carries a conventional signature.

Rohit Gupta, Souvik Ray, Arun K. Somani and Zhao Zhang, proposed an anonymous lookup protocol that provides high degree of client and server anonymity (and hence also unlinkability) in P2P networks. This protocol tries to provide requestor and service (Client - Server) provider anonymity and unlinkability in order their Identity will not be traceable. Otherwise in P2P systems server identity is always known to terminal nodes. The protocol builds a distributed anonymity infrastructure by implementing an incentive scheme that motivates nodes to participate in the system and also maintains secrecy of the identities of nodes they interact with as part of any transaction. The protocol uses an auction protocol for trading of network resources and ensures that the rewards received by network nodes are maximized if they truthfully follow the protocol steps. Moreover, the protocol is light-weight as it does not rely on any trusted centralized entity or require specialized encryptions to be performed by the nodes.

With the increase of the number of services which a user would like to use, it is becoming more and more tedious for the user to establish and manage pairs of a user name (pseudonym) and a password of multiple services. Hence much attention is recently paid to authentication systems which enable users to use multiple services after they register or sign on only once at a registration manager. For example, credit card systems and single-sign-on systems such as OpenID. Such a system is called an authentication system in multi-service environment. In multi-service environment raises a new privacy problem for users, the daily activity of a user can be revealed by gathering information about his/her usage of multiple service providers. Service providers usually maintain service logs of the transactions for the purpose of the detection of abuse, audit, and diagnosis of problems, and they can collect their log files and trace actions of a user from his/her transactions.

Service logs of a user in a service provider can be associated with those in other service providers if a pseudonym of the user is used among multiple service providers. In fact, credit card systems and OpenID are typically based on such an implementation, hence much more information in various service providers can be collected due to leakages of the service logs or illegal coalitions among multiple service providers.

Privacy protocols have attempted to ensure anonymity, unlinkability and deniability. On one hand, service providers want to authenticate legitimate users and make sure they are accessing their authorized services in a legal way. On the other hand, users prefer not to expose any sensitive information to anybody. They want to have complete control on their personal data, without being tracked down for wherever they are, whenever and whatever they do. User Privacy Attributes must be pointed out, such issues are:

Anonymity: Anonymity is the state of being not identifiable within a set of subjects (users), known as the anonymity set. To enable anonymity of a subject (user), there always has to be an appropriate set of subjects with potentially the same attributes.

Pseudonymity: Being pseudonymous is the state of using a pseudonym as identification. Therefore, users can be identified through their pseudonym but they

remain anonymous as far as their real identity is concerned. Clearly, it is assumed that each pseudonym refers to exactly one holder, invariant over time, and is not transferred to other users.

Unlinkability: Ensures that a user may make multiple uses of resources or services without others being able to link these uses together. It requires users to be unable to determine whether the same user caused certain specific operations in the system.

Undetectability: Undetectability of a user, from an attacker's perspective, means that the attacker cannot sufficiently distinguish whether it exists or not.

Unobservability: Ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used. It requires users to be unable to determine whether an operation is being performed.

2.8 Vulnerabilities of the existing protocols and techniques

The aforementioned technologies have the following disadvantages:

1. Many of the already widely used systems and protocols are customized to provide encryption methods but do not offer anonymity and unlinkabilty because such systems are aware of all information about the participants, their behaviour, history of used services and content of messages. They are vulnerable in trace back or spoofing attacks.

2. They provide encryption mechanisms as a means to provide anonymity, but fail to protect authenticity and integrity of the exchanged messages. Systems such JANUS encrypts URLs but do not provide confidentiality of data stream.

3. Another crucial reason is that service providers gather identity information about users in order to provide services. A subject enjoys privacy when he can control, on his own, the dissemination of his private information. These technologies exploit confidentiality and encryption mechanisms only to control access to users' personal data, but fail to provide anonymity, unlinkability to users.

4. These technologies are more privacy-enhancing tools than models oriented to support global and anonymous online transactions. Thus, they do not integrate technologies, which help users explore and make use of the online transactions.

The main concern is data security, which has become one of the most important concerns for almost every user, such as governments, financial institutions, hospitals, and private businesses. An important security risk is that user information can be captured and read during its transmission. How do users protect this information from being read by intruders? Encryption provides message confidentiality by transforming readable data (plaintext) into an unreadable format (ciphertext) that can be understood only by the intended receiver.

The encrypted messages may be intercepted and be the object of attacks, including illegal reading, modification, and replay. The problem is that many web applications exchange sensitive information over insecure channels. Applications that communicate with external applications interchange sensitive data that may be read by unauthorized users while they are in transit. The solution for this problem is affected by the following factors:

Confidentiality: Messages may be captured while they are in transit, so it is crucial to prevent unauthorized users from reading them by hiding the information of the message.
Reception: The hidden information should be revealed conveniently to the receiver.
Protocol: The need to apply the solution properly or it will not be able to stand attacks, there are several ways to attack a method to hide information.

Performance: The time to hide and recover the message should be reasonable. So there is the need to prevent unauthorized users from reading them by hiding the information of the message using an encryption method.

2.9 Attacks

Many techniques and protocols have been proposed in order to provide unlinkability and protect user identity. However the existing techniques suffer from vulnerabilities. For example the history of downloaded item can be gathered at a service provider. And a third party may identify the user from the information that is given to the service provider.

Attacks may be raised and is possible attackers to breach the security, to revile and discover information about the user. There are two main types of attacks the passive and active attacks. The following attacks are among the most common attacks, their goal is to reveal information about the users and take advantage of them.

2.9.1 Man in the middle

Man-In-The-Middle attack is the type of attack where attackers intrude into an existing connection to intercept the exchanged data and inject false information. It involves eavesdropping on a connection, intruding into a connection, intercepting messages, and selectively modifying data.

In cryptography, the man-in-the-middle attack or bucket-brigade attack, or sometimes Janus attack, is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection when in fact the entire conversation is controlled by the attacker.

Also Man-in-the-Middle attacks are often referred to as "session hijacking attacks", in this attack the intruder aims are to gain access to a legitimate user's session and tamper it. The attack usually starts with sniffing and eavesdropping on a network stream, and ends with trying to alter, forge or reroute the intercepted data. Man-in-the-Middle attacks are usually selected by hackers against public-key cryptosystems. For example in a public key scenario, hackers may substitute the intercepted public key with their forged public keys. Quite often in such cases, the victim parties are made to believe that they remain safe in communicating with each other.

2.9.2 Traffic Analysis

Traffic analysis is a type of passive attack. The first step to any type of hacking is footprinting , and wireless footprinting is done by carrying out traffic analysis. The attackers before mounting an active attack have to obtain sufficient information about the network. This operation of traffic analysis gives the attacker some basic information about the network, like, the network activity going on, the protocols being used by the network and also the active access points (AP's) of the network. The determination of the active AP's is employed by a process known as Wardriving. This is a process where the attacker physically moves around to find out about the active AP's of the network to be attacked. By finding out about the active AP the attacker can use the AP as a starting point to mount his/her attack. This is done by detecting and capturing the beacon frames sent out periodically by the AP's.

2.9.3 Passive eavesdropping attacks

Passive eavesdropping is a passive form of information violation or theft. A passive eavesdropping attack occurs when a malicious user manages to listen in on traffic exchanged between two users. For this to happen an attacker needs access to the

traffic; this can occur in the following ways. This attack discloses the information about the network but at the same time the attacker can access and read the message contents. If the Message is encrypted, this will require the attacker to break the encryption and read the message. The attacker can mainly gain two types of information by mounting this sort of an attack, the attacker can read the data transmitted in the session and also get variable information about the packet characteristics.

2.9.4 Trace-back attack

Another risk is that the identity of the user can be revealed through a trace-back attack, whereby someone traces the path back to the initiator along the forward or the reverse path. Furthermore, by linking specific communication channels, sequences, sessions with certain client-server, pairs can be achieved by tracing the contents and or the size of a message travelling over a communication link or by attempting to detect and analyse periodically transmitted packets, aiming to discover their source through specific time correlations. Many security protocols have recently been proposed for peer-to-peer and ad hoc networks, to address these issues.

Because the service provider may require some sensitive information from user, such as email address to send back to user some information later and if user show the link to database on the server, the system will easily know the true identity of user. (Note that, these data only consist of sensitive information like user preferences or personal information, so it is not much.) And suppose that user sends some sensitive information to the service; if that data in database on server is not encrypted, the system can easily link them to the user's true identity because those data is often unique. Some nodes in the network can be malicious, whose goal is to compromise the identity of the nodes that request and/or provide a service. Malicious nodes do not aim to maximize their profits (and in fact are prepared to incur loss), and can work in collusion, so as to identify who originated a request and who served it.

2.10 Cryptographic background

2.10.1 Symmetric Encryption

Cryptography is a security related technology, it prevents unauthorized entities from accessing the information, and cryptography prevents them from understanding what the data means. Notice that encrypted data is still data, but without meaning in relation to its purpose. So, the encryption process transforms data into other data form that loses the prior meaning.

Symmetric encryption is the oldest cryptographic technique used to transmit or store digital data securely. Data that has been encrypted with symmetric encryption is considered to be confidential, in the sense that only those entities who have the key can understand what the data means. In the symmetric encryption there are two parties who share a key and will use this key to communicate and share data safely. The main security goals are privacy and authenticity of the communicated data. This chapter presents the most popular encryption methods.

With symmetric encryption, confidentiality is guaranteed by the use of a secret key (private key). This system uses only private keys for exchanging messages between certain users. With this key, and a mathematical algorithm, data is converted to an unintelligible form. By using the same key, and applying the mathematical algorithm in the reverse way, the original data can be reconstituted. The algorithm is said to be a two-way algorithm which specifies an encryption. The algorithm, also is called encryption scheme, it tells the sender how to process the plaintext using the key, thereby producing the ciphertext that is actually transmitted. An encryption scheme also specifies a decryption algorithm, which tells the receiver how to retrieve the original plaintext from the transmission while possibly performing some verification, too. Finally, there is a key-generation algorithm, which produces a key that the parties need to share.

Symmetric algorithms can be divided into two main classes: stream algorithms and block algorithms. Stream Algorithms operate directly on a stream of bytes whereas

block algorithms operate by transforming fixed-length groups of bits, a block, at a time Block algorithms are most commonly used in the IT world today.

2.10.2 Asymmetric or Public key cryptography

Asymmetric or Public key cryptography was introduced by Diffie and Hellman. Two keys are used a public and a private key. The setting of public-key cryptography is also called the asymmetric setting due to the asymmetry in key information held by the parties. Namely one party has a secret key while another has the public key that matches this secret key. This is in contrast to the symmetry in the private key setting, where both parties had the same key.

For example Alice generates a public and a private key. The Public Key is given to anyone who would like to securely communicate with Alice. Alice keeps the private or secret Key private and she may decrypt messages encrypted with the public key. Asymmetric cryptography is not a replacement of symmetric cryptography

Asymmetric encryption schemes

An asymmetric encryption scheme is just like a symmetric encryption scheme except for an asymmetry in the key structure. The key Public Key used to encrypt is different from the Private Key used to decrypt. Furthermore Public Key is public, known to the sender and also to the adversary. So while only a receiver who has the secret key can decrypt, anyone who have the corresponding public key can encrypt data to send to this one receiver.

Asymmetry allows for easier key distribution, so there is not need for a trusted third party to distribute keys. Asymmetry provides originality because the secret key is something that only one entity knows. However Asymmetric Encryption is not as fast or computationally efficient as symmetric cryptography. The Communication

Overhead, PKIs typically require significant communication overhead (frequent and large messages).

The two main branches of public key cryptography are:

- In Public key encryption a message is encrypted with a recipient's public key, cannot be decrypted by anyone except a possessor of the matching private key, this entity will be the owner of the key and the person that is associated with the public key used. This procedure is used for confidentiality reasons.

- Digital signatures: a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, proving that the sender had access to the private key so is the person who is associated with the public key used and the part of the message that has not been tampered with.

2.10.3 Hash Functions

With the advent of public key cryptography, the sender of an encrypted message could no longer be authenticated simply based on the fact that he knows the encryption key. When encrypting messages with public keys, it is therefore necessary to separately authenticate the sender. Of course, this problem is not limited to encrypted messages. Methods of authentication became important with the rise of digital communication.

It would be possible to encrypt the message with the sender's private key. Either the message is encrypted with the recipient's public key once more, or attached to the message, to encrypt both. Then, to authenticate the sender, it can be decrypted using his public key. However, apart from either complicating handling, or unnecessarily doubling the length of a secret message, it has several other disadvantages. For example, the message always has to be decrypted before the authenticity can be verified, and thus can only be verified by the recipient. Also, the authenticity

information cannot be detached from the message. A lot of such problems can be eliminated by not using the whole message as the signature, but instead calculating a form of checksum of the message, the message digest. A signature scheme can be devised with this digest and public key cryptography. Thereby, a small, detachable signature can be produced, conveniently handled, and independently confirmed.

The security of such a signature scheme is naturally limited by the strength of the underlying method of encryption. However, the signature has to be able to withstand several kinds of attack. It has to be impossible to forge a signature: nobody should be able to sign any message without the signers' private key. Finding a message that a valid signature can be attached to must be impossible as well. Additionally, it has to be impossible to alter a signed message in any way, without invalidating the signature. Of course, gaining any information about the message out of the signature should not be possible as well. All these demands require the generation of the message digest to have specific properties. In addition, it has to be infeasible to generate two messages with the same digest, as well as find a message that has a given digest.

A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string the cryptographic hash value, any change to the data will change the hash value. The cryptographic hash function has four main properties, it is easy to compute the hash value for any given message, it is infeasible to find a message that has a given hash, also it is infeasible to modify a message without changing its hash, and it is infeasible to find two different messages with the same hash. Hashing can be something like a 'fingerprint' of data.

2.11 Uses of Cryptographic Functions

Cryptographic hash functions are necessary for useful authentication protocols and digital signatures. There are many signature standards, one of the most popular is the digital signature algorithm (DSA). However, because of their various properties,

cryptographic hash functions are well-suited for many types of applications. In the digital world, they are in everyday use.

The hash value of a set of data should always be unique. Consequently, any file can be easily identified by its hash value. Additionally, the integrity of data can be verified this way. This section mentions the most well known hash functions, such as MD-4, MD5, SHA-1 and SHA-2.

A hash function is irreversible. Therefore, it can be used to prove knowledge of a secret without revealing it. Additionally, the irreversibility is used by saving only the hash of a password. To gain access to a system, the provided password is then hashed and the result is compared to the saved hash. This scheme ensures that even the system administrator has no knowledge of the password, and is, slightly modified at times, widely used for user authentication. In addition, a hash function can be utilized to transform a password into a key with specific properties. For example, keys for symmetric ciphers have to have a certain size, like 128 bit. Another excellent and important use of cryptographic hash functions is the generation of pseudo-random numbers. A simple setup is the hash of a combination of a counter with a seed. This random number generator will have an infinite period, if the counter is allowed to become arbitrarily large. Session identifiers are usually created this way. The same aspect can be used for encryption. The output of hash functions can be used either to create block ciphers, or even as a stream of bits for stream ciphers. Due to the properties of cryptographic hash functions, they have many more uses throughout all areas of computing.

A cryptographic hash function h is a mapping from the set of input data, binary strings of arbitrary length, {0, 1} = N, to the set of hash values {0, 1}d which are fixed in size, and considerably small, d is usually a value between 128 and 512. Computation of the hash function should be very fast. The complexity of a hash calculation should be h(n) for a string of length n.

Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value. A cryptographic hash function must be able to

withstand all known types of cryptanalytic attack. It must have the following properties:

Preimage resistance
Given a hash h it should be difficult to find any message m such that h= hash (m). This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks. In other words is a type of attacks on hash functions for finding a message that has a specific hash value.

Second preimage resistance
Given an input message m1 it should be difficult to find another input message m2 where m1 = m2 such that hash (m1) = hash (m2) some times referred to as weak collision resistance, and functions that lack this property are vulnerable to second preimage attacks.

Collision resistance
It should be difficult to find two different messages m1 and m2 such a pair hash (m1) = hash (m2) is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for preimage resistance, otherwise collisions may be found.

2.12 Types of cryptographic Hash Functions

SHA-0 and SHA-1

The Secure Hash Algorithm (SHA) was initially approved for use with the Digital Signature Standard (DSS) later the standard was updated to become what is currently known as SHA-1. The first version of SHA is referred in the cryptographic literature as SHA-0, although it has never been its official designation. SHA-1 differs from SHA-0 by exactly one additional instruction, which is extremely important from the

cryptanalytic perspective. Since there were no reasons to prefer the initial version of the standard, SHA-1 replaced SHA-0 in all but most antiquated applications.

SHA-1 is closely modelled after MD4, taking some cues from MD5. It uses the same padding algorithm, breaking the message into 512-bit blocks and encoding the length as a 64-bit number. The size of its internal state and its output length are 160 bits, which is substantially longer than MD5's 128 bits. Although its round functions are simpler and less varied than those of MD5, there are more of them 80 instead of 64. SHA-1 uses a more complex procedure for deriving 32-bit subblocks from the 512-bit message. If one bit of the message is flipped, more than a half of the subblocks get changed.

SHA-224, 256, 384, 512, 1024

The new standard issued by NIST and adds three members to the SHA family functions. The connections between the NIST approved functions are the SHA-256 and SHA-512 which have similar designs, with SHA-256 operating on 32-bit words and SHA-512 operating on 64-bit words. Both designs have strong resemblance to SHA-1. SHA-384 is a trivial modification of SHA-512, which consists of trimming the output to 384 bits and changing the initial value of the chaining variable. SHA-224 is defined as a truncated version of SHA-256 with a different initial value. This hash function provides 112-bit level of security, on par with triple DES. The most important difference between the three new functions and SHA-1 is the new message schedule (procedure for deriving sub blocks from one block of the message).

2.12.1 Hash Functions Applications

Hash functions are one of the most popular techniques of encryption and are used in several applications. An important application of secure hashes is verification of message integrity. Determining whether any changes have been made to a message, for example can be accomplished by comparing message digests calculated before,

and after, transmission. A message digest can also serve as a means of reliably identifying a file, several source code management systems, use the SHA-1 sum of various types of content to uniquely identify them.

Another related application is password verification. Passwords are usually not stored in cleartext, for security reasons, but instead in digest form. To authenticate a user, the password presented by the user is hashed and compared with the stored hash. This is sometimes referred to as one-way encryption.

Digital signatures were the first application of cryptographically secure hash functions. Hash functions serve a dual role in practical signature schemes, they expand the domain of messages that can be signed by a scheme and they are an essential element of the scheme's security. For both security and performance reasons, most digital signature algorithms specify that only the digest of the message be signed, not the entire message. Hash functions can also be used in the generation of pseudorandom bits. Hashes are used to identify files on peer-to-peer where users share files over networks.

2.13 Digital Signatures

A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document, and possibly to ensure that the original content of the message or document that has been sent is unchanged. Digital signatures are easily transportable, cannot be imitated by someone else, and can be automatically time-stamped. The ability to ensure that the original signed message arrived means that the sender cannot easily repudiate it later. A digital signature can be used with any kind of message, whether it is encrypted or not, simply so that the receiver can be sure of the sender's identity and that the message arrived intact. A digital certificate contains the digital signature of the certificate-issuing authority so that anyone can verify that the certificate is real.

For digital signatures there is a key pair: a secret digital signature key, which is used for signing data items, and a public digital signature verification key, which is used by others to check if a signature was made with the related signature key. If a digitally signed document is tampered with in any way, its digital signature will not verify.

A digital signature works by creating a message digest which ranges from between a 128-bit and a 256-bit number which is generated by running the entire message through a hash algorithm. This generated number is then encrypted with the sender's private key and added to the end of the message. This hash code is a uniquely-identifying digital fingerprint of the check. If even a single bit of the check changes, the hash code will dramatically change. The next step in creating a digital signature is to sign the hash code with your private key. The use of Digital Signature insures by means of verification and validation that the user is whom he/she claims to be. This is done by combine the users credential to the digital certificate and in turn this method uses one point of authentication. Digital certificates ensure data integrity providing security to user that the message or transaction has not been accidentally or maliciously altered.

A digital signature is evidence that a given private key acted on a document. Also, a digital signature proves that the document has not been changed since the signature was made.

Digital signatures is a method that insures by means of verification and validation that the user is whom he/she claims to be. Digital certificates provide confidentiality and ensure that messages can only be read by authorized intended recipients. Also digital certificates also verify date and time so that senders or recipients can not dispute if the message was actually sent or received.

The components that a digital signature comprise of:

The public key: This is the part that any one can get a copy of and is part of the verification system.
User's name and email address: This is necessary for contact information purposes and to enable the viewer to identify the details.

Expiration date of the public key: This part of the signature is used to set a shelf life and to ensure that in the event of prolonged abuse of a signature eventually the signature is reset.
Name of the company: This section identifies the company that the signature belongs too.
Serial number of the Digital ID: This part is a unique number that is bundled to Digital signature of the CA (certification Authority): This is a signature that is issued by the authority that issues the certificates.

Digital signatures deliver assurances in the role that the other party will keep their part of the bargain. Knowing the identity of the other party is one way of gaining that assurance. Keys used for digital signatures are very long series of bits, which can be represented as long series of alphanumeric characters. This makes digital signatures unfeasible for an individual to remember. To stop fraud certificate authorities have been introduced. Certificate authorities can sign user's public key, ensuring that no one else uses the information or impersonated his key. If a user is uncertain of the digital signature it is possible to verify the digital signature with the certificate authority.

2.13.1 Digital Certificates

A digitally signed combination of a signature verification key with other data that states a property of the related secret key is known as a certificate. Generally, a digital certificate is a collection of data that has been signed with a digital signature key. which is asserting the validity of the contents of the certificate and, where there is enclosed signature verification key, more specifically that they assert the stated property of the secret signing key associated with this verification key.

2.13.2 The Parties to a Digital Certificate

In principle there are three different interests associated with a digital certificate:

1. The Requesting Party is the party who needs the certificate and will offer it for use by others, they generally provide some or all of the information it contains.
2. The Issuing Party is the party that digitally signs the certificate after creating the information in the certificate or checking its correctness.
3. The Verifying Party is the party that validates the signature on the certificate and then relies on its contents for some purpose.

2.13.3Types of Digital Certificate

I. Identity Certificates

An Identity Certificate is one that contains a signature verification key combined with sufficient information to identify the keyholder.

II. Accreditation Certificates

This is a certificate that identifies the keyholder as a member of a specified group or organization without necessarily identifying them. For example, such a certificate could indicate that the keyholder is a medical doctor or a lawyer. In many circumstances, a particular signature is needed to authorize a transaction. Accreditation certificates can also be viewed as authorization or permission certificates.

III. Authorization and Permission Certificates

In these forms of certificate, the certificate signing authority delegates some form of authority to the key being signed. For example, a Bank will issue an authorization certificate to its customers saying the key in this certificate can be used to authorize the withdrawal of money from account number. In general, the owner of any resource that involves electronic access can use an authorization certificate to control access to it.

Although authorization certificates might contain identity information this will not generally be necessary. For example, in the use of digital certificates to implement 'electronic cash' individual transactions need to be anonymous. They have to provide authorization certificates without identity information. Certificates should be designed carefully for specific purposes and the information they contain should be only that required for this and no more.

2.14 XML Digital Signatures

XML Signatures is another type of digital signatures. XML Digital Signatures provide integrity, message authentication defining whether a message was altered during transmission, and signer authentication prove that a message was originated from. Data objects are digested, the resulting value is placed in an element and that element is then digested and cryptographically signed. XML Signature also called XMLDsig, XML-DSig, XML-Sig defines an XML syntax for digital signatures and is defined in the W3C recommendation XML Signature Syntax and Processing for generating and validating digital signatures for authenticating XML documents. It has much in common with PKCS but is more extensible and geared towards signing XML documents.

It is used by various Web technologies such as SOAP, SAML, and others. XML signatures can be used to sign data resources of any type, typically XML documents. There are three types of XML Digital Signatures the Enveloped, Enveloping and Detached.

Enveloped: the XML signature will itself be embedded within the signed document. An enveloped signature is a signature that is contained inside the data that it is signing.

Enveloping: the signed data is actually embedded within the XML signature element. An enveloping signature is a signature over data that is inside the signature element.

Detached: the signed entities and the XML signature are separate. A detached signature is over data that is external to the signature element. That is:

In the operation of the XML Digital Signature the sender generates a small unique a hash or digest of the document. Every minor change will cause the hash value to change, by comparing the hash that was received with the hash calculated from the received document, the recipient can verify whether the document was altered. The hash of the document signed or encrypted with the sender's private key acts as a digital signature for the document. The recipient will be able to verify or decrypt the signature by taking a hash of the message and verifying it with the signature that accompanied the message and the sender's public key.

The digital signature helps to ensure that the signature is authentic. When the receiver verifies the message with the sender's public key, the receiver knows that the sender signed it. Also the signature cannot be forged, this means that only the sender knows his or her private key. The signature is not reusable. The signature is a function of the document and cannot be transferred to any other document. The signed document is intact, if there is any alteration to the document, the signature verification will fail at the receiver's end because the hash value will be recomputed and will differ from the original hash value. The signature cannot be repudiated, so the sender cannot deny previous committed actions, and the receiver does not need the sender's help to verify the sender's signature.

XML documents can hold information with different security level. Some of the information can be read by anyone and other information should only be read by few authorized persons. With the use of XML encryption, selected information in

the document can be encrypted separately. Because XML's syntax is structured using markup, one or more elements and sub elements can be handled differently. This is very useful for large and growing documents holding lots of information for multiple parties. Since these documents can be handled by many people with different authorities it is important to secure elements in the document. Then different people can have access to the document, but only read the data they have assigned access to. The accessibility to read the encrypted information depends on the known secret key. In the proposed approach that follows in this section we use the XML Digital Signatures in order to protect users sensitive data offer data integrity, authenticity and high level of unlinkability and security in online transactions between users and distributed CSPs.

Chapter 3

Proposed System

3.1 A privacy-echancing security model

The model that is introduced is a new infomediary-based, privacy enhancing security model, which is capable of providing unlinkability and security, to users in their online transactions over the cloud. Using this model, users can access services without revealing their real identity or preferences to cloud service providers, and cloud service providers can offer services without violating the privacy of users.

Using cloud services, users can easily access their personal information and make it available to various services across the Internet, for these reasons and to protect users' personal data confidentiality, this protocol uses an Infomediary (I/M) mechanism that can help authenticate users and services based on credentials and characteristics. A key issue concerning I/M in clouds is interoperability drawbacks that could result from using different identity tokens and identity negotiation protocols. Existing password-based authentication has an inherited limitation and poses significant risks.

In the proposed protocol the user can authenticate himself with only one Single Sign On, providing his Username and Password. Then he can have access to the cloud and ask for any service from a service directory. The services usually are located in distributed cloud service providers. Assume that *CSPn* is a set of cloud service providers/ servers (CSPs) in a multi-server environment, or else in a cloud

environment where *CSPn : {CSP1, CSP2. . . , CSPn }.* Initially users U authenticate themselves to the system providing username and password, then U requests services from the infomediary entity I/M. The I/M receives the requests and finds the most suitable service which is available from a list of cloud service providers (CSPs) and sends it back to the user. The users communicate with different CSPs only through the I/M they do not have direct contact with CSPs and they do not authenticate themselves to the CSPs nor provide any other information about their identity.

Individuals should be given the choice for unlinkable communication over the Internet. Infomediaries (I/M) are business entities supporting the development of anonymous business models. We strongly believe that they can be used to increase security and unlinkability in online transactions. Their basic role is to accumulate information about web users, and deal services on behalf of them. On the one hand, I/M can protect privacy by hindering in third parties from collecting users personal data, while they offer distributed services over the internet. An I/M system should be able to protect private and sensitive information related to users and processes. While users interact with I/M, in order to get access to a service, this service might need to ensure that their identity is protected from other services with which it interacts. In multi-tenant cloud environments, providers must not know customer identity and authentication information. For example a user called Alice asks a distributed service. The request contains sensitive data such as credit card numbers, email and other personal information, so it is important to keep it secret. In the receiving operation, different cloud service providers (CSP) handle different parts of the request. A malicious user can intercept these orders and may try to read them, to get the confidential information. In order to avoid such attacks we introduce the use of an Intermediary I/M, this third party receives the users sensitive information and requests and creates a secure communication channel between the User and CSP, however user and CSP do not communicate directly with each other but only through I/M.

This chapter presents such a security protocol, capable of supporting security and unlinkability in cloud environment, such as in electronic transactions. The use of I/M enable users to use multiple services from different and distributed Cloud Service Providers without revealing personal data and, without violating privacy.

The suggested I/M model includes three distinct entities: the user, the Cloud Service Providers (CSPs), and the Infomediary (I/M), in this model we assume the I/M can be trusted and each legitimate user can get service granted from the cloud service providers through the I/M, and it considers the additional functionalities.

- **Users**: send their requests of service to the I/M.
- **Cloud Service Providers**: send their description of services to the I/M.
- **I/M**: is a trusted third party and takes the requests, matches them with the Cloud Service Providers in the local database and develops a secure unlinkable and trusted connection between them. Also it manages user personal information (credentials).

The assumptions include:

- Users should trust I/M as they are agents of their personal information. I/M is considered to be a trusted part of the system.
- I/M may ask user for personal information/ credentials in order to authenticate the user and provide him with services, through a secure communication channel.
- The user's authentication in I/M is materialized with the use of certain certificates/ techniques, such as Xml Digital Signature.
- The I/M anonymizies the personal data by substituting the person's identity with a randomly generated number (Random User ID).
- During authentication the I/M gives user a random identity. I/M provides unlinkability between users and cloud service providers.
- After authentication user has access to a list of services through I/M.
- Cloud Service Providers authenticate themselves to the I/M with the use of XML Digital Signature, thus they consider I/M legitimate party and communicate through a secure channel. And also send a description of their service in I/M, and their Server ID.

Infomediary (I/M) is a protocol entity whose main role is to collect user's information, and secure users personal data from different service providers. I/M basic operation is based on taking users requests and finding the most suitable available service for them over the internet. In order to do so, users send their request for a service to I/M and the latter develop a secure communication channel between the users requested service and the offering services from CSPs. On the other hand, CSPs are available to I/M, without establishing a direct communication channel with users. The matching between user's requests for services and different cloud service providers is compiled by I/M. Taking for granted that I/M is operating in a trusted environment, security and anonymity of users' personal data are ensured. I/M is a third party designed to manage users identities and prevent the linkage of their identities.

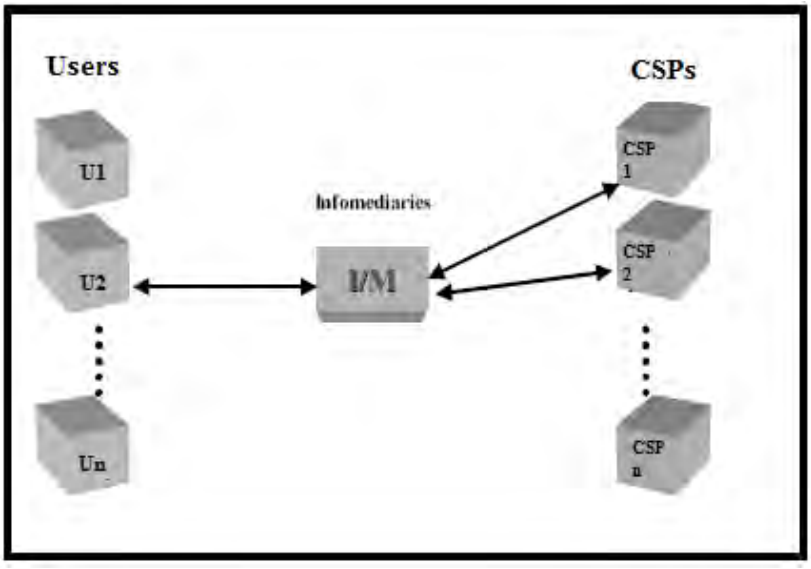


Fig. 3.1: Operation of infomediary

The user authentication is materialized with XML Digital Signature for a number of reasons, such as:

- Messages may be captured while they are in transit, so there is the need to prevent unauthorized users from reading them by hiding the information of the message using encryption, also the need to express encrypted elements in a

standardized XML format to allow encrypted data to be nested within an XML message. Otherwise, different applications cannot interoperate. Different parts of a message may be intended for different recipients, and not all the information contained within a message should be available to all the recipients. Thus, recipients should be able to read only those parts of the message that are intended for them.

- For flexibility reasons, both symmetric and asymmetric encryption algorithms should be supported. If a secret key is embedded in the message, it should be protected. Otherwise, an attacker could read some messages.

3.2 Protocol description - Protocol process of communication

Entity	**Requirements (expectations)**	**Functionality**
User (U_x)	Requests services from cloud service providers. Retain privacy and anonymity	
Cloud Service Provider (CSP_x)	Distributed servers which offer services to users, don't know users identity or their preferences.	Delivery of services to I/M
Infomediary (I/M)	Act on users' benefit, own large databases. Trusted by cloud service providers and users.	Collection of offering services, gathering of users requests for services. Supervision of model procedures. Protection of anonymity, privacy, and authenticity,

		Collection of personal information, Collection of user preferences.

Table 3.1: Model's entities, requirements, and functionalities

3.2.1 Protocol operation

Fig. 3.1 presents the basic operations of a single I/M. Additional functionalities should be described, should such a network of interconnected users and services, operating in an untrusted environment. These functionalities include:

1. A secure acquaintance mechanism, an I/M directory service between different users and cloud service providers. Thus, all members are capable of introducing themselves to the I/M.

2. Users should trust I/M as they are agents of their personal information. Thus, a proper security infrastructure (i.e. XML Digital Signature) should be provided to implement this trust.

3. Integration of the basic I/M operations (i.e. user sing in, services collection, and matching) may violate users anonymity in case of collusion attack. The suggested I/M model takes into account the additional functionalities. It includes three distinct entities: the User, the Cloud Service Provider, and the I/M (Table 1). For each entity there is a number of requirements and functionalities. The requirements refer to each entity expectations. The functionalities refer to the implemented functions, as well as to the roles played by each entity. Initially, U_X, CSP_X have to register to I/M. A typical registration procedure may be followed using the web.

The protocol used for the implementation of the suggested model is the following:

1. The user sends a message with the service request and his personal information (PI), and the message (M) which are encrypted with his private key to the I/M. Then it creates a digital signature with the secret key of the U, the message of the PI and the message.

$$E\_k(M) \quad \textbf{(1)}$$

$$S\_k(M) \quad \textbf{(2)}$$

$$D\_k\text{-priv\_U}(\{\text{PI,M}\}) \quad \textbf{(3)}$$

2. I/M receives encrypted message containing the service request *SR* and then decrypts and verifies the XML Digital Signature using his key. Where the timestamp *T* denotes the current date and time when this M message was signed and is sent to the I/M.

Service Request: SR

$$M = \{SR, T\}$$

$$I/M : \{PI, M\} XMLDigSig \quad \textbf{(4)}$$

3. I/M anonymizes the information and substitutes user's personal information with a Random User Id, in order to hide his identity. This Random User Id is constituted by numbers and it is 10 bits long. Every time a user requests a new service this Random User ID (UID) changes. So there would be no identity disclosure and no trace- back of the history of downloaded item which can be gathered at a service provider. And a third party may not be able to identify the user from the information that is given to the service provider.

The method that is used is Secure Random. When a new identifier is needed, I/M generates a random id number using SecureRandom. The random id number is encoded in byte and checks if the result is already being used, if it is not already taken, it is suitable as a unique identifier.

$$I/M \rightarrow U_X : \{UID, M, T\} XMLDigSig \quad \textbf{(5)}$$

4. Then the I/M sends a request to the CSP and asks a description of the offered services, using XML Digital Signature.
Service Description: SD
Timestamp: T

$$I/M \rightarrow CSP : \{SD, M, T\} XMLDigSig \quad \textbf{(6)}$$

5. CSP decrypts and verifies the encrypted message and sends to the I/M his Server ID with the description of the offering service.
Cloud Service Provider Identity: CSPID

$$CSP \rightarrow I/M : \{CSPID, SD\} XMLDigSig \quad \textbf{(7)}$$

6. The I/M decrypts the received message using secret key and adds in the local database (the framework uses hash tables) the requested service and the User ID and sends to the user the description of the service with the essential Ids in order to hide the identity of the involved parts in the transaction.
When matching is performed, then U_X sends the following messages:

6.1To I/M: It sends SD and UID encrypted with Xml Digital Signature (XMLDigSig).

$$I/M \rightarrow U_X : \{UID, SD\} XMLDigSig \quad \textbf{(8)}$$

7. Finally I/M sends from the list of the available services the CSPID of the service that matches to the user request encrypted with his secure key.

$$I / M \rightarrow U_X : \mathrm{E}\_k(SR, CSPID) \quad \textbf{(9)}$$

8. U_X decrypts the received message using the private key and accepts services.

Using XML Digital Signatures and timestamps the authenticity and integrity of messages is enhanced. When the receiving M is sent , the I/M, the User and CSPs check the validity of messages, so there is T1 compared to T2, $T2 - T1 \leq \Delta T$ where T2 denotes the date and time when the entity received the request M, and ΔT is a predefined time interval to balance the reasonable transmission delay and potential replay attack.

The major element, which has been introduced by the suggested model, is the use of an I/M operating in a trusted environment, as a means to protect user privacy. The basic operation of this model is the notion of an I/M supervising the communication between the user and distributed Cloud Service Provider. This element is introduced to increase privacy, anonymity, unlinkability and security of users.

In order to avoid collusion attacks, and direct communication between Users and Cloud Service Providers personal information, there are different tables for users, and available services. Only I/M manages users personal information, and substitutes them with a random identity number. This way no personal information is being known to the Cloud Service Providers. The intermediation of I/M during delivery of services makes impossible the direct communication between Users and Cloud Service Providers.

3.3 The use of XML Digital Signatures in the proposed protocol

The proposed protocol uses the technique of XML Digital Signature because is one of the basic technologies for sharing data and information over internet and other networks. XML data is stored in plain text. This means that XML is software and hardware independent and therefore easy for all applications to handle and transfer. XML digital signatures prevents the repudiation, it is a significant technique in improving the security of web services and particular in recognizing the identities of users. The digital signature is one of the most important cryptographic primitives. It provides data integrity, message authentication and non-repudiation, which are required attributes in security critical services, such as electronic commerce, voting or health care.

The purpose of XML encryption is to obtain confidentiality and privacy in XML documents. Its goal is to add security functionality to existing XML documents by encrypting the important data it is holding. The encryption and decryption of XML is based on existing encryption algorithms. This makes the use of XML encryption easy for businesses and organizations that want to protect their confidential information in transfer (e.g. between client and server). Organizations and businesses handle confidential information. For example medical records are data, holding sensitive information about persons/patients. Each hospital and many other health care environments are using records to store this information about patients. These journals can for example contain information to identify that person (name, address). It can hold financial information and it will mostly hold some medical information about that person (prognosis for chronic conditions). Name and address are usually considered as information which is not so sensitive, although it may not be necessarily for a doctor to know where the person lives. Financial information is also not important for a doctor to know, except the data about the patients insurance and social security number. The medical information for the patient is essential information that only certain users should know.

To keep different information in files protected for different parties in a hospital, XML encryption can be used to restrict the access to this information. The doctor can get access to the medical information they need to know to do their job. The financial responsible in the hospital (or other health care environment) can get access to

and take care of the economics. General information, like name etc, can be available for the hospitals information.

Users are worrying about how their personal data will be used and how this data can be protected against unauthorized access. A defence against online privacy infringement consists of security models, protocols and approaches that do not reveal the identity of the communicating parties. Such approaches are called anonymous. Internet operation may be based on anonymity.

3.4 How Common Threats Are Dealt with Resistance to common attacks

The basic encryption mechanism this protocol follows is the use of XML Digital Signatures. Such technology is deployed to defend against most common threats.

- *Unauthorized access*: it is avoided through the use of secure Random User Id number, with that technique users are protected because when the user authenticates himself to the I/M, the I/M sends back a Random User Id number which will be users Id for only one transaction. Every time the user requests a service from the I/M, it always give user a different Id number. So attackers cannot find out the real identity of users, the random Id always changes. As it was mentioned the use of I/M as an intermediary between the users and CSPs and with the use of the above techniques offers a secure communication channel among them, without having actually direct contact. The different cloud service providers do not request users Ids because the offered services are known only to the I/M and the I/M is the only responsible entity to match the user requests with offering services from different distributed services. Users random Id and the

- *Data integrity and Authenticity:* Data integrity in XML Digital Signatures is maintained by checking the digital signatures of signed parts, when data is digitally signed, the signature can be verified to check the data integrity and authenticity, if a user changes parts of the XML document then the data has been modified and the digital signature makes the content invalid. That means the data didn't come from the entity that claims to have created and signed it. The data object which is signed originally should be accessible to verify the signature.

- *Collusion*. There is no way of direct communication between CSPs and users. I/M is the intermediatory, also is a trusted party between these two entities. User's description requests and personal information are encrypted with XML Digital Signatures.

- *Resistance to Modification Attack:* In the i^{th} request for a service session, when the user sends D_k-priv_U({PI, M}) to the infomediary, assuming that the attacker intercepts the transmitted message and tries to modify the message. In the proposed scheme, the attacker is unable to modify the message due to the use of XML digital signature, even if *A* attacker or an eavesdropper modifies the message the I/M cannot validate the message because the hashed message is different from the original one.

- *Resistance to Replay Attack and Impersonation Attack:* To resist the replay attack, our scheme uses the concept of timestamp. When the intruder replays the previously intercepted messages and wants to masquerade as a legal user. The intruder must change timestamp T into a new time $T*$ such that $(T - T*) \leq T$, where T is the timestamp given at the time when the I/M receives the illegal message. Timestamps fulfill a critical need in the validation process. Once T is changed, the value of hash also changes so the message fails in the validation phase. And the I/M will reject the request from the intruder. A digital signature includes a timestamp from the local system's clock. Inclusion of such a timestamp ensures that any tampering with the timestamp value will be detected when the recipient verifies the signature. In addition, a copy of the

timestamp accompanies the digitally signed message when the message is routed to its destination.

If the hash of a message is calculated again and is informed then the initial hash A is compared with the last hash B inside the signed message to confirm they are equal, so compares the recomputed hash value with the received hash value; if the two are not identical, fails during the validation.

- *No replay or repudiation of messages*: Avoided by time-stamping all messages. The technique of electronic time-stamping allows users to get an electronic proof of the existence of a document at a specific point in time. XML Digital Signatures are time-stamped. It is a validation and integrity mechanism. The X.509 Public Key Infrastructure Time-Stamp Protocol, which is used, describes the use of a time stamp authority to establish evidence that a signature existed before a given time. Time value added by the signer representing the claimed time of signing, the value of this element is the declared signing time of a document, meaning when the signature was generated. The user that created the message has the secret key, which has signed the message and only he can change the content of the message. The ability to ensure that the original signed message arrived means that the sender cannot easily repudiate it later. If the signature contains different time-stamping, then the declared signing shows that the message has been altered. So replay, spoofing, eavesdropping and trace back attacks are avoided.

  For example consider the case where someone signed the action part of the request, but didn't include the user name part. In this case an attacker can easily take the signed request as is, and just change the user name and resubmit it. These replay attacks are much easier when you are signing a small part of the document. To prevent replay attacks, it is recommended to include user names, secure keys, and timestamps, into the signature.

- *Resistance to Eavesdropping Attack:* We assume that an attacker A eavesdrops on U's message in the i^{th} authentication session and request for a service sends the message to I/M for the $(i+1)^{th}$ authentication session. Obviously, I/M rejects the message, because U's has digital signed it and the timestamp of the message is different and Id number is always updated in every new request for a service, I/M checks the valid time interval and the Id number. Therefore, our model protects U and CSP from replay attack.

3.5 Test Results

The test and the evaluation of protocol process took place in NetBeans IDE 4.9 Profiler by Oracle and ran on Windows OS. The simulation results were measured according the memory consumption, the model needs to use through the running time of the application, in order to detect memory leaks, also the surviving generations and garbage collection and the time they consume during the process and the live threads and the number of loaded classes. The test/ simulation time was set to be in 20 ms and there are four simulation case studies. In the first there are 1000 users that request simultaneously many services. In the second there are 2000 users, in the third 5000 users and in the fourth 7.000 users. The authentication of every user is programmed to occur every 1 second. These tests are materialized because there is the need to discuss different parameters in cloud environment and if the proposed model is suitable and scalable to be used in cloud environment.

3.5.1 Performance parameters

As it was mentioned there is an impact of parameters/characteristics, which influence the performance of the framework and there is the need to analyze them in detail.

- *Heap size:* indicates the allocated size of the JVM heap which takes care of the new objects being created.

- *Used heap size:* indicates the amount of heap space actually is in use.

- *Threads:* are the active threads in the JVM.

- *Surviving generations (SG):* is the number of different ages of all the Java objects on the JVM's heap, age is defined as the number of garbage collections that an object has survived.

- *The Avg.Age:* value is calculated using the Live Objects. The age of each object is the number of garbage collections that it has survived.

3.5.2 CPU application performance

The CPU obtains detailed data on application performance, including the time it took to execute methods and the number of times the method is invoked. Table 3.2 represents and compares the different number of users and the memory that is actually used, the garbage collection according time, the alive objects.

We measure the amount of additional memory that is required for facilitating the proposed approach. This measure refers to the amount of memory that must be dedicated to the performance of framework during the execution of the protocol. This obtains detailed data on object allocation and garbage collection. The following table presents the basic performance parameters. As the number of users increase the same happens to the heap space which is used, also the surviving generations and garbage collection rate increases, and the threads and loaded classes. In other words as the number of users increases, such the amount of memory the framework needs to perform is getting higher too.

Users	Heap size (MB)	Used max heap size	SG	Threads (max)	Loaded classes (max)	Time (ms)
1000	**153**	**137**	**28**	**1.005**	**1.916**	**20**
2000	**260**	**232**	**26**	**1978**	**2.714**	**20**
5000	**260**	**252**	**49**	**3.823**	**1.741**	**20**
7.000	**260**	**260**	**61**	**4.726**	**3.916**	**20**

Table 3.2: CPU measurements

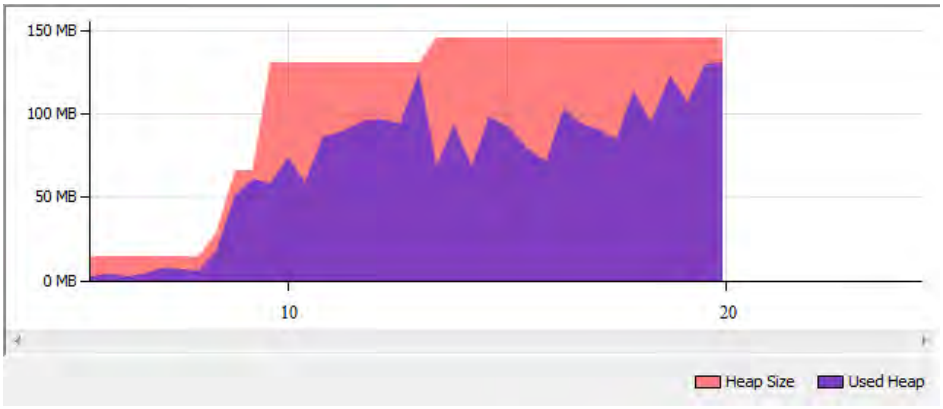


Fig. 3.2: CPU heap size 1000 users

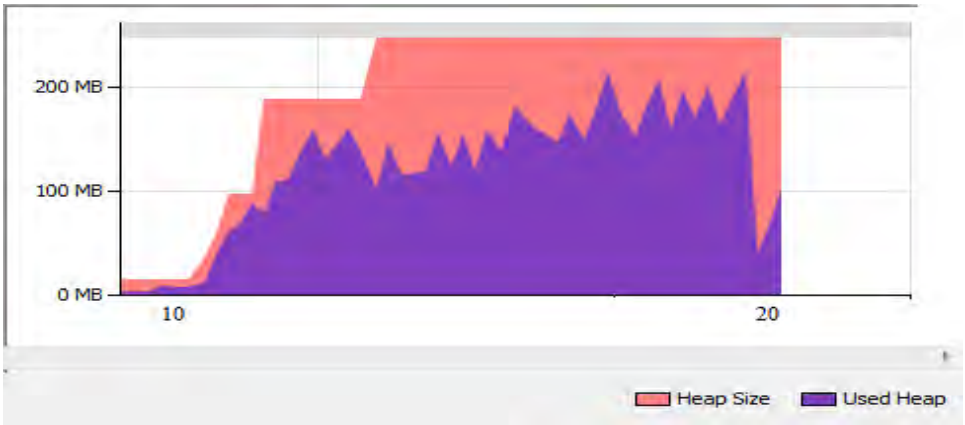


Fig. 3.3 : CPU heap size 2000 users

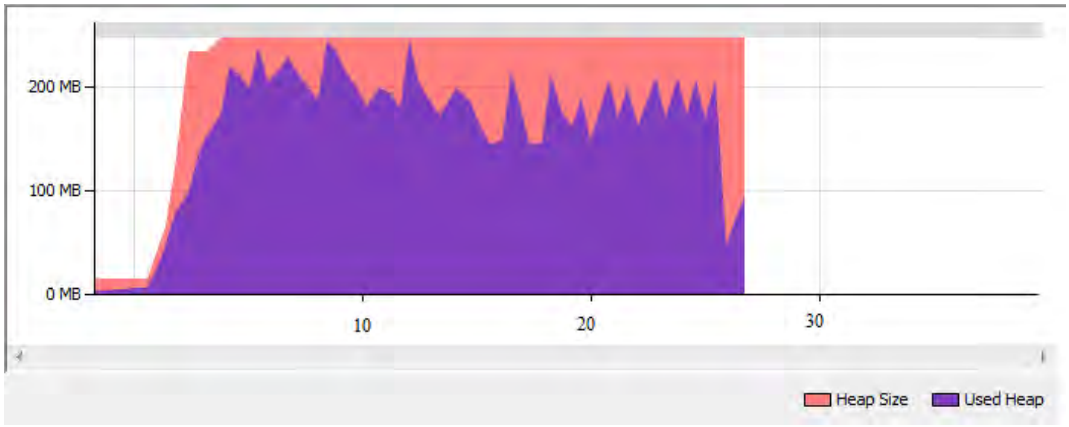
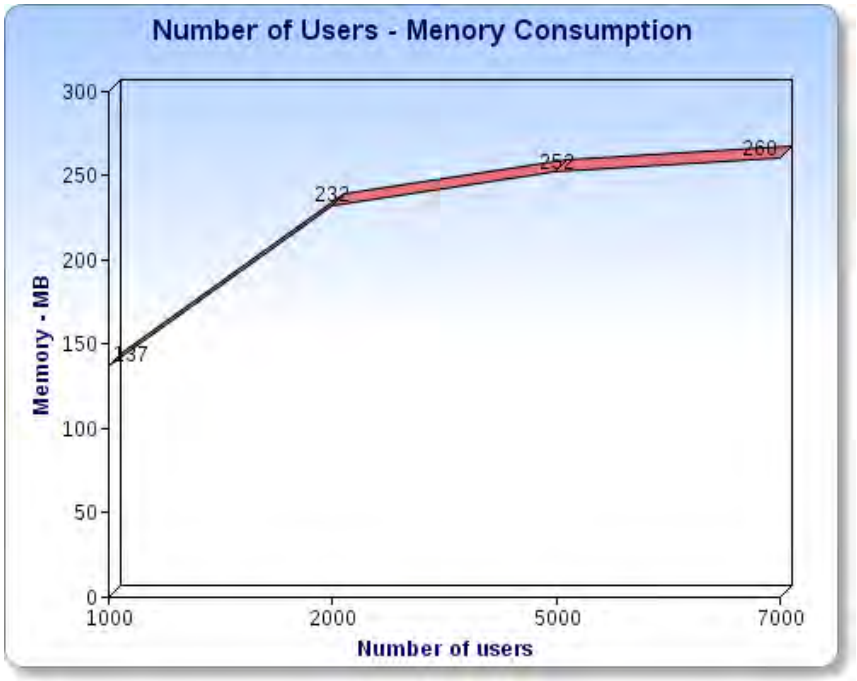


Fig. 3.4 : CPU heap size 5000 users

Fig. 3.5 : CPU heap size 7000 users



Graph 3.1: Comparing number of users and memory consumption

The graphs in figures 3.3, 3.4, 3.5, 3.6 show heap size, the red shading indicates the allocated size of the JVM heap which takes care of the new objects being created. The purple overlay indicates the amount of heap space actually in use. The graphs above examines the allocated heap size at the last update was between 140 and 260 megabytes. If the objects being created exceed the max heap size of the max heap size in any case study, then it will throw an error saying memory Out of Bound. The performance analysis in the first and second case studies show that at the runtime of the application, when the program actually executes the heap size that is needed and is used is lower than the existing amount of heap space. But in the third case study with 5.000 users the used heap size is 252 MB and the available heap size is 260 MB, also in the fourth case study with 7.000 users the used heap size is the same with the available heap size, it consumes 260 MB there is risk of memory error because it is all used so in case studies with users more than 7.000 there is high memory consumption and may cause memory errors.

The graph 3.1 represents a comparison between the number of users and the memory they need to perform in order to authenticate themselves and request services from the system. As it is shown as the number of users increases the same happens to memory consumption.

The difference between 1000, 2000, 5000 and 7000 users is obvious. The 1000 users need 123 MB, while the 7000 need 260 MB.

The following graphs show the creation of new objects and garbage collection in the application in four different case studies. The first case study (fig 3.6) represents the authentication and the simultaneously request for service of 1000 users. The second one represents the same scenario with 2000 users (fig 3.7), the third with 5000 users (fig 3.8) and the last one with 7.000 users (fig 3.9). The increase of surviving generations between those four graphs has a steadily rate over relative time means that there are references to older objects as the number of users increases. So there may be an insignificant memory loss because after a few time and until the end of the application the increasing rate is stable. Only at the start up of the application there is a significant increase in surviving generations.

The following graphs show two important heap statistics:

- The purple line is the percentage of execution time spent by the JVM doing garbage collection and is graphed against the y-axis on the right edge of the graph. Time spent by the JVM doing garbage collection is time that is not available for it to run your application.

- The red line is surviving generations and is graphed against the y-axis scale on the left edge of the graph. The count of surviving generations is the number of different ages of all the Java objects on the JVM's heap, age is defined as the number of garbage collections that an object has survived. When the value for surviving generations is low it indicates that most of the objects on the heap have been around about the same amount of time. As is shown in the figures the increase in surviving generations in the case studies, indicates that as the number of users increases the same occurs to the surviving generations, however the increase at the start up of the application in more obvious. This indicates that application is allocating new objects while maintaining references to a few of the older objects it already allocated. Those older objects may lead to wasting or leaking memory of the application. But in our

situation this is possible in the third and fourth case studies because the incising rate is higher in these two case studies. Also the time it took to execute is getting higher as the number of users increases. So potential limitations in the performance will be when the rate of surviving generations and garbage collection start to increase and cause memory loss.

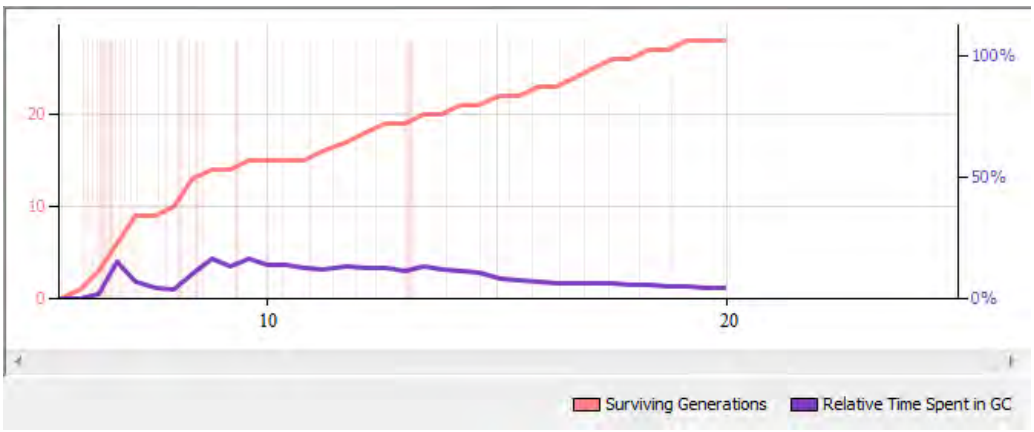


Fig. 3.6: CPU Surviving Generations according time - 1000 users

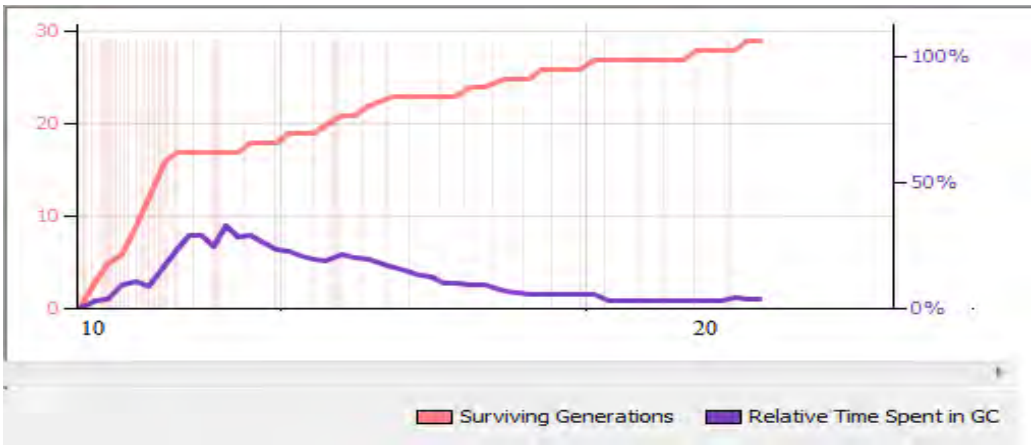


Fig. 3.7: CPU Surviving Generations according time - 2000 users

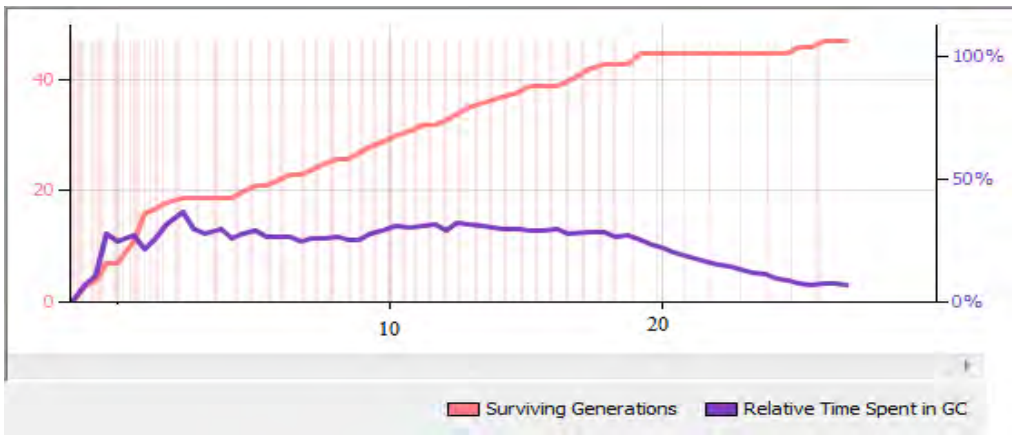


Fig. 3.8: CPU Surviving Generations according time - 5000 users

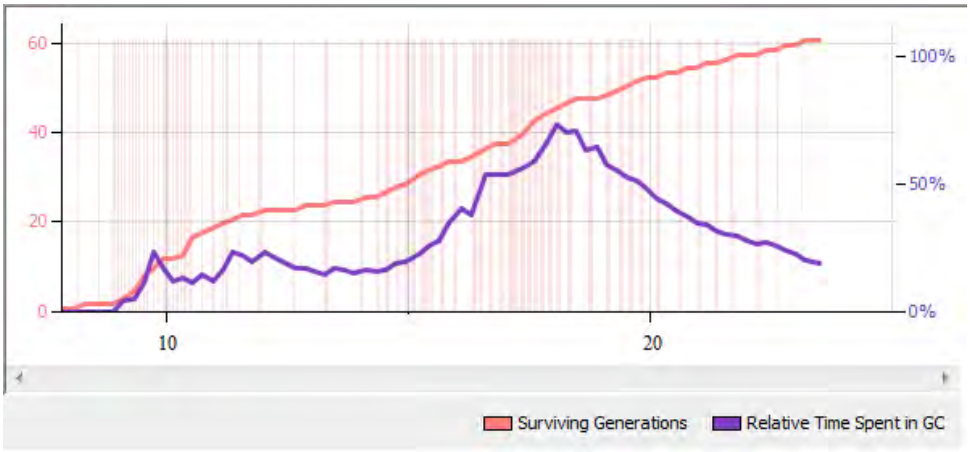


Fig. 3.9: CPU Surviving Generations according time - 7000 users

Threads are increased at a stable rate, the same happens with loaded classes. On the Threads Timeline, the NetBeans profiler displays the current and past status of all threads in an application.

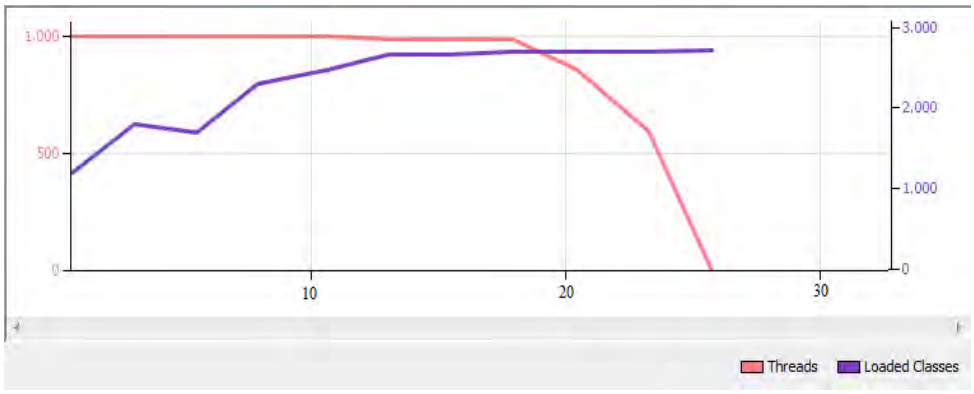


Fig. 3.10: CPU threads and loaded classes- 1000 users

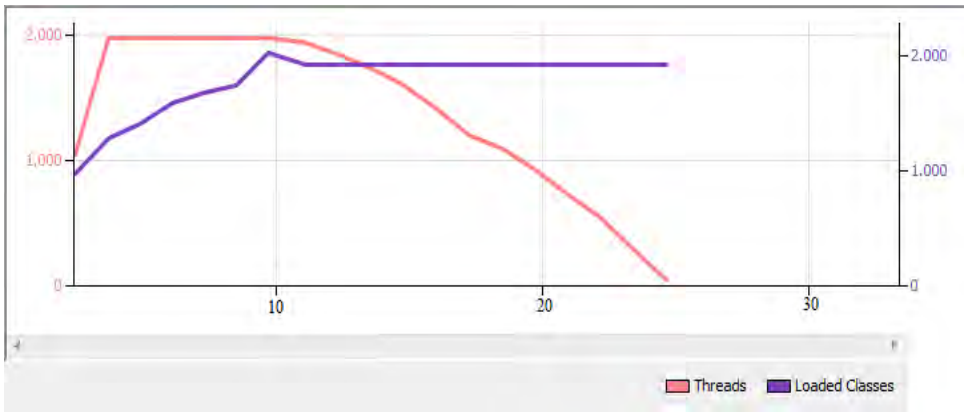


Fig. 3.11: CPU threads and loaded classes- 2000 users

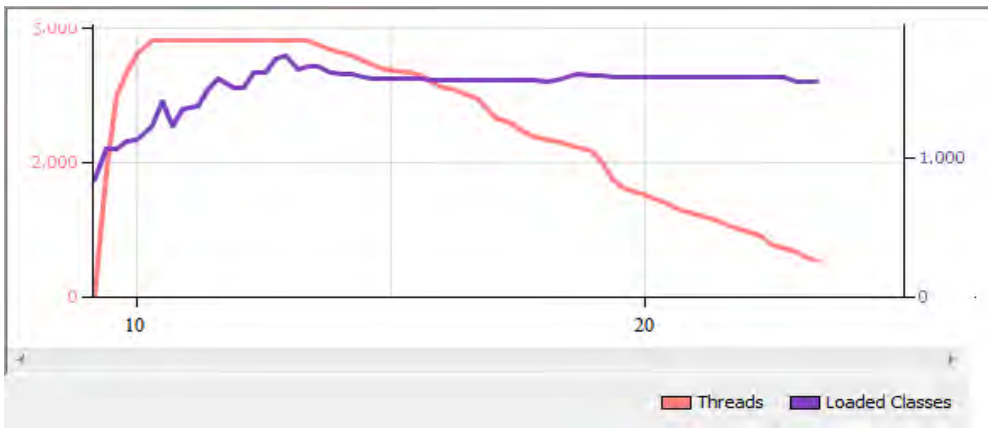


Fig. 3.12: CPU threads and loaded classes- 5000 user

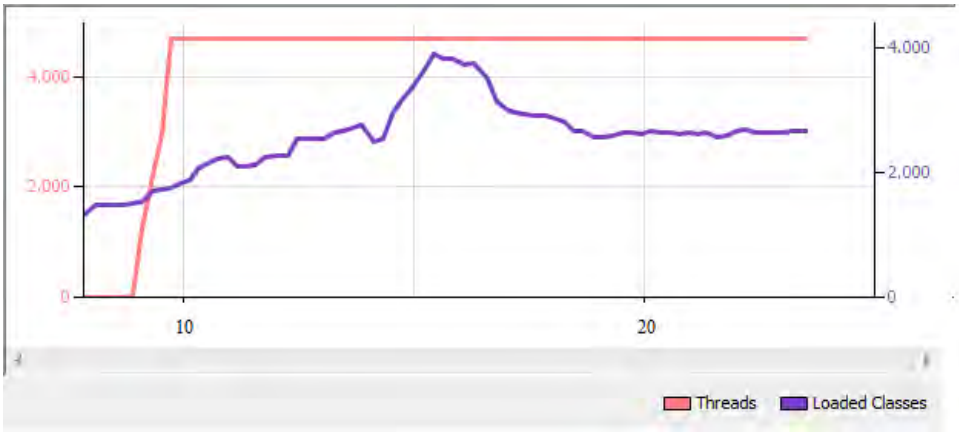


Fig. 3.13: CPU threads and loaded classes- 5000 user

3.5 Model limitations

Cloud enables users to access and use web applications that reside in vast data centres located around the world instead of on their own personal computers. These applications based in the cloud are supposed to offer users multiple on-demand service, and be also scalable.

The proposed model faces the problem of single point failure, because in the proposed system if the infomediary collapse then the users would not be able to access to services they are interested in. I/M is a main part of a system which, if it fails, will stop the entire model from working. This kind of problem is mostly undesirable in any system with the goal to offer high availability and scalability, like a software application which is designed to perform in cloud environment.

Through the tests and the evaluation of the framework, there is the observation that as the number of the users increases the same happens to the overhead and the memory consumption. These may be not scalable for large scale cloud environments. As it is shown in measurements (table 3.2 and graph 3.1) when the framework manages 7.000 users who requests services, it is not as much scalable as manages 5.000 users.

Chapter 4

Conclusion

4.1 Future Work

Cloud computing is the next step in the evolution of on-demand information technology services and products. “Cloud” computing builds on decades of research in distributed computing. It implies a service-oriented architecture, reduced information technology overhead for the end-user, great flexibility, reduced total cost of ownership, on-demand services and many other things. But one of the greatest concerns is the security, anonymity and unlinkability of users when they transact in such an environment.

The Proposed approach provides secure transactions to users and protects them from different type of attacks such as replay, spoofing, eavesdropping and traceback attacks. Also the proposed framework proves that such approach operates really well in a small scale cloud environment. However the cloud is an evolving area much research and work should be done in the field of unlinkability and privacy of the users.

Several issues have been identified and need to be addressed, new secure techniques of sharing sensitive data over the cloud, also including questions regarding the use of

other encryption techniques that can be integrated with the one presented in this thesis. Our future work also includes the further implementation from a simple framework which is developed to perform in a small scale cloud to a framework which will be able to operate in web and will offer unlinkability to users over the internet. Also protocols are needed able to address three requirements: confidentiality of business relations among the various CSPs, user privacy, and strenght of identity verification. We argue that with continued research advances in encryption methods, providing trusted and secure computing, life in the cloud can be advantageous.

In essence we believe that our proposed approach encourages the development of innovative cryptographic systems in order to provide security and privacy in cloud environments.

4.2 Conclusion

Cloud computing is still in its infancy. Cloud computing is clearly one of today's most enticing technology areas due, at least in part, to its cost-efficiency, flexibility and storage. There is a wide variety of service providers, both large and small, delivering a wide variety of cloud-based services. For example, there are applications, support services, mailfiltering services, storage services, etc. However, cloud computing offers many packages of products and services but shields security risks. It is important to understand the evolution of computing in order to protect users' safety of how they got into the cloud environment and provide them unlinkability. Looking at the evolution and the growth of the cloud computing many protocols and mechanisms have been established in order to offer protection to users. As technology migrates from the traditional on-premise model to the new cloud model, security risks evolve almost daily.

Unlinkability and anonymity on the Internet is tough to ensure. However, there is a crucial need for secure and unlinkable communication in the information society. Although existing technologies offer, more or less, technical advances to secure communication for users over the internet, they present many disadvantages. On the

other hand, infomediaries is a promising technology not only on the financial area, but also in security area over cloud environments. This thesis represents a high-level description of proposed system, it is a system that offers unlinkable transactions between users and distributed CSPs through an intermediary trusted party I/M which is designed to run on cloud environment.

The basic characteristics of the suggested model are.

- The introduction of I/M as a privacy-enhancing agent in multiple transactions over the cloud. This protocol introduces a stand-alone I/M model with entities co-operating in a secure and anonymous way, offering unlinkable and anonymous transactions.

- The I/M model, is not vulnerable to collusion attacks. The separation of protocol's entities and operations is estimated to largely contribute in making multiple transactions over the cloud.

- Suggested model's candidate technologies are well established not only in the application level but also in the international standards area. This makes model's implementation feasible and compatible with existing privacy enhancing technologies.

- The evaluation of the proposed framework proves that such approach operates really well in a small scale cloud environment. The analysis of performance according to measurements shows that the framework needs small amounts of memory when it manages less than 5000 users and consumes more memory and more overhead and has memory leaking when it manages over 7000 users.