UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

MAURO JOSE ARAUJO DE MELO

MODELO DE AUTENTICAÇÃO PARA SISTEMAS DE COMPUTAÇÃO EM NUVEM

São Luís

2013

UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

MAURO JOSÉ ARAÚJO DE MELO

MODELO DE AUTENTICAÇÃO PARA SISTEMAS DE COMPUTAÇÃO EM NUVEM

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão. Como um dos requisitos para obtenção do título de mestre.

Orientador: Zair Abdelouahab

São Luís

2013

Melo, Mauro José Araújo de
Modelo de autenticação para sistemas de computação em nuvem / Mauro José de Araújo de Melo – São Luís, 2013.
101 f.

Dissertação (Mestrado) – Universidade Federal do Maranhão – UFMA. Centro de Ciências Exatas e Tecnologia. Departamento de Engenharia Elétrica. Programa de Pós-graduação em Engenharia de Eletricidade, 2013.
"Orientador Prof. Dr. Zair Abdelouahab".

1. Computação em nuvem 2. Gerenciamento de identidade 3. Autenticação I. Título.

CDD: 004.738 5

MODELO DE AUTENTICAÇÃO PARA SISTEMAS DE COMPUTAÇÃO EM NUVEM

Mauro Jose Araujo de Melo

Dissertação aprovada em 29 de novembro de 2013.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Profa. Karla Donato Fook, Dra.
(Membro da Banca Examinadora)

Prof. Sofiane Labidi, Dr.
(Membro da Banca Examinadora)

“As verdadeiras conquistas, as únicas de que nunca nos arrependemos, são aquelas que fazemos contra a ignorância”.

Napoleão Bonaparte

AGRADECIMENTOS

Primeiro a Deus que me concedeu a graça da vida além da realização dos sonhos.

Ao meu Porto Seguro que é minha família, base de tudo que sou hoje agradeço a meus pais Torquato (in memoriam) e Naíde que não cessaram de insistir na minha educação. Meus irmãos que sei que torcem também pelo meu sucesso Raimundo Francisco, Nídia Socorro, e Torquato Filho.

A minha esposa, companheira e cumplice de todas as horas Francineide por passar dois anos sendo “Pãe” (Pai e mãe) dos nossos filhos e muitas vezes se pondo em meu lugar como filho e cuidando da minha mãe.

Aos meus filhos Mauro Filho e Mayara Caroline pela paciência, amor, carinho e força com que me “empurravam” para concluir sempre que quis fraquejar no meio do caminho. Pelos momentos que eram especiais para ambos e que eu não pude estar presente.

À Coordenação de Pós-Graduação de Engenharia Elétrica e a Universidade Federal do Maranhão – UFMA por haver me concedido a oportunidade de cursar o Mestrado neste Programa e aos seus professores que muito nos auxiliam e orientam.

Ao meu Orientador Prof. Dr. Zair Abdelouahab, pela grande ajuda, paciência, compreensão e dedicação dispensada à realização desse trabalho. O apoio e instrução que serviram a me mostrar um “norte” para que pudesse resolver os problemas que me deixaram “perdido” sem saber que rumo seguir.

Aos colegas de laboratório: Neto, Vladimir, Liana, Dhileane (Dhully), Luís, Renato, Gleison, Claudio, Mário, Leonardo, Carol, Wendell e Jonathan e pelos momentos agradáveis no LABSAC e também os que não foram.

Aos amigos e irmãos aqui “encontrados” Josenilson, sua esposa Karla e “Lairzinha” que me apoiaram, escutaram e levantavam meu astral em muitos momentos ruins. Ao Eduardo e sua esposa Olivia por sempre me “convidar” para passeios e saídas espairecedoras. A tia Delza que me acolheu como ela mesma diz “meu filho piauiense” que muitas vezes me convidou para sua casa para almoços e festas em sua casa.

Ao Fernando Sérvulo, companheiro de aluguel, que muito contribuiu descontraidamente para que a vida no apartamento fosse suportável.

Meus agradecimentos a CAPES pelo apoio financeiro.

A todos que, direta ou indiretamente, contribuíram para a elaboração desta dissertação.

RESUMO

O mundo atualmente encontra-se migrando a utilização dos sistemas tradicionais para um modelo computacional que oferta seus recursos de maneira bastante dinâmica e remotamente, que é conhecido como computação em nuvem. Em compensação surgem desafios relacionados à questão de segurança, o que torna notório que necessitam inovações em recursos. Por ser um ambiente onde há o compartilhamento dos recursos computacionais por diversas organizações, faz-se necessário delimitar o acesso a dados bem como aos serviços, de maneira que mantenha a privacidade de identificação dos usuários do ambiente. O primeiro mecanismo de segurança em qualquer sistema computacional é o gerenciamento de identidade, através da autenticação de acesso. Um usuário que possui suas informações de identificação cadastradas em um sistema terá o acesso garantido em sistema, caso sua autenticação seja comprovada. O objetivo deste trabalho é propor um modelo onde a autenticação de determinada identidade possa ser deslocado de uma nuvem a outra para permitir que um usuário cadastrado em um domínio possa acessar a outro domínio. Esta autenticação é realizada através de *login* único que é enviado através de script criptografado.

Palavras chave: Computação em nuvem, Gerenciamento de identidade, Autenticação.

ABSTRACT

The world today is moving the use of traditional systems to a computer model to offer its resources very dynamic and remotely, which is known as cloud computing. In compensation challenges arise related to security issue, which makes it clear that innovations require resources. Being an environment where there is sharing of computing resources by several organizations, it is necessary to define access to data and services, so that you keep the privacy of user identification of the environment. The first security mechanism on any computer system is the identity management through authentication. A user who has registered their identifiable information in a system will have guaranteed access to the system, if your authentication is proven. The objective of this work is to propose a model where the authentication given identity can be moved from one cloud to another to allow a registered user to access a domain to another domain. This authentication is performed through single sign that is sent encrypted via script.

Keywords: Cloud Computing, Identity Management, Authentication.

LISTA DE ABREVIATURAS

AC – Autoridade Certificadora
Amazon EC2 – Amazon Elastic Compute Cloud
AMQP – Advanced Message Queue Protocol
API – Application Programming Interface
ARP – Address Resolution Protocol
AS – Authentication Services
CP – Credential Provider
CTS – Credential Translation Services
EC2 – Elastic Compute Cloud
EUCALYPTUS – Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems
FISMA – Federal Information System Management Act
GPL – *General Public License*
HaaS – Hardware as a Service
IaaS - Infrastructure as a Service
IdaaS – Identity as a Service
IdP – Identity Provider
IEC – International Electrotechnical Commission
IETF – Internet Engineering Task Force
IMS – Identity Management System
ISO – International Organization for Standardization
ITU – International Telecommunication Union
KDC – Key Distribution Centre
LVM – Logical Volume Manager
MIT – Massachusetts Institute of technology
NIST – National Institute of Standards and Technology
OASIS – Organization for the Advancement of Structured Information Standards
OSI – Open System Interconnection
PaaS – Platform as a Service
PAP – Policy Access Point
PC – Personal Compute
PDP – Policy Decision Point
PEP – Policy Enforcement Point
PIP – Policy Information Points
PKI - Public K*ey* Infrastructure
QoS – Quality of Service
RFC – Request for Comments
RST – Request Security Token
RSTR – Request Security Token Response
S3 – Simple Storage Service
SaaS – Software as a Service
SAML – Security Assertion Markup Language
SLA – *Service-Level Agreement*
SOAP – Simple Object Access Protocol
SP – Service Provider
SSH – Secure Shell
SSHPASS – Secure Shell Password
SSL – Secure Socket Layer

SSO – Single Sign-On
STS – Security Token Services
STS/IdP – Security Token Services/Identity Provider
TGS – Ticket Granting Services
TGT – Ticket Granting Ticket
UML – Unified Modeling Language
URL – Uniform Resource Locator
VLAN – Virtual Local Area Network
VM – Virtual Machine
W3C – Word Wide Web Consortium
WSS4J – Web Service Security for Java
XACML – eXtensible Access Control Markup Language
XKMS – *XML Key Management Specification*
XML – eXtensible Markup Language
XRI – *eXtensible* Resource Identifier

LISTA DE FIGURAS

Sumário

1. INTRODUÇÃO

O mundo passou a apresentar um novo cenário, principalmente no que envolve a tecnologia. O homem quer seja por comodidade ou simplesmente pela busca de melhorias na sua vida, passou a inserir estas inovações no seu cotidiano. Como por exemplo: TV digital, telefonia móvel, recebimentos de salários em contas bancarias, onde antes estes eram efetuados em lojas por intermédio dos contras-cheques, pagamentos de contas que somente podiam ser realizadas em comércios para estes fins podem ser feitas digitalmente, etc.

Uma das tecnologias que se encontra entre as tendências emergentes mais importantes dos últimos anos, tanto na vida de determinadas corporações quanto no setor acadêmico, é denominada de Computação nas Nuvens (Taurion, 2012).

A computação em nuvem esta diretamente relacionada com a entrega de recursos computacionais compartilhados e estejam relacionados ao armazenamento, processamento ou mesmo de softwares para usuários através da Internet (Santos, Westphall, 2011).

Um usuário de posse de um hardware e conexão de rede pode acessar as muitas facilidades que este tipo de computação pode oferecer. Porém, até que ponto o usuário encontra-se realmente com seus dados “seguros”? E as empresas?

O fato de determinados usuários e corporações alocarem seus dados e informações particulares inseridos na nuvem, torna necessário o aumento da segurança das nuvens computacionais.

Um usuário deve autenticar-se para entrar na nuvem, e encontrar-se autorizado para acessar a todos os arquivos que lhe pertencem e permanecem implantados nesta nuvem.

Para que o acesso seja feito em uma nuvem computacional se faz necessário que o usuário possua prerrogativas que lhe assegurem o acesso dos dados particulares ou empresariais, sem que estes sejam expostos a terceiros. Portanto é imprescindível aliar as tecnologias com a segurança, de maneira que haja suporte a autenticação e autorização de acesso não somente a uma nuvem, mas que possam também ser utilizadas entre nuvens distintas.

1.1. Motivação

Quando empresas fazem migração para a nuvem, necessitam abrir mão de algum nível de controle. Neste caso precisam ter plena confiança nos sistemas e nos provedores que realizam a autenticação com a identificação. O gerenciamento de identidades, serviços de autenticação de terceiros e a identidade federada são considerados elementos fundamentais para a segurança da nuvem (Dokras et. al., 2009).

O desafio do gerenciamento de identidades federadas (*Federated Identity Management*) é oferecer uma infraestrutura que permita tanto aos usuários uma experiência *Single Sign-On* (SSO), quanto aos administradores um mecanismo para autenticação e controle de acesso aos recursos federados entre parceiros (Feliciano et. al., 2011).

O trabalho voltou-se a realização de um controle de acesso para o ambiente de computação nas nuvens, buscando minimizar a problemas inerentes do ambiente.

Em cada serviço computacional seja ele qual for, ou convencional, ou distribuído ou na nuvem é necessário que seja realizado o controle de acesso para a utilização das aplicações, caso o usuário dos serviços possuam autenticação e autorização para o mesmo.

No controle de acesso de muitos sistemas, os usuários tem que administrar as diversas formas de autenticação para cada sistema em que possui cadastro para acesso. Para minimizar o acúmulo de diversas formas de *loggin,* o usuário busca utilizar aos padrões já existentes e que são amplamente utilizados para manutenção do controle de acesso em uma nuvem. Onde esta é parte integrante de um circulo de confiança em que as informações podem ser compartilhadas.

1.2. Objetivos

1.2.1 Objetivo geral

Na segurança do ambiente computacional, conta-se com o objetivo geral de propor um modelo que realize o gerenciamento de identidades em uma nuvem computacional, uma vez que dentro dos padrões de segurança da computação em nuvens precisa aumentar proteção de dados que estão nelas inclusos.

1.2.2 Objetivos específicos

A partir da perspectiva idealizada deste projeto foi necessário direcioná-lo em busca de alcançar aos objetivos específicos, que são eles:

° Realização de pesquisa sobre os principais problemas relacionados ao controle de acesso, mais especificamente ao gerenciamento de identidades;

° Analisar técnicas utilizadas para controle de acesso de autenticação em um ambiente de computação nas nuvens;

° Montar e configurar ambientes de computação nas nuvens para realização de testes.

1.3. Estruturação da Dissertação

A dissertação encontra-se organizada da seguinte maneira:

No capítulo 2 inicia-se falando sobre Segurança da informação. No capítulo 3 é feita uma analise sobre gerenciamento de identidades. O capítulo 4 realiza-se um levantamento referencial sobre Computação nas nuvens, conceitos, utilização, modelos, serviços, problemas da segurança. No capítulo 5 mostra-se a abordagem proposta de um ambiente onde a autenticação e centralizada e outra distribuída. No capitulo 6 ocorre a realização da validação e implementação para que a meta fosse atingida e no capítulo 7 descreve-se a conclusão e sugestões de trabalhos futuros.

2. SEGURANÇA DA INFORMAÇÃO

A contemporaneidade do mundo volta sua atenção à “informação”, pois esta se torna importante tanto no mundo dos negócios quanto na vida pessoal, sendo bastante atraente para realização de novos empreendimentos.

Uma pessoa ao ser detentora de informações que podem ser consideradas importantes a empresas e / ou pessoas, leva consigo uma enorme gama de oportunidades, tornando o cenário dos negócios cada vez mais dinâmico, gerando uma competitividade comercial, o que requer que todas as informações obtenham proteção especializada.

Por outro lado, quando não se tem uma informação ou até mesmo quando se tem informação de má qualidade pode gerar uma ameaça que pode ocasionar falências às empresas.

Para tanto se torna preciso realizar a proteção de informações, o que se chama de segurança da informação.

A segurança da informação é a garantia de que as informações fiquem sempre protegidas contra o acesso por pessoas não autorizadas, independente dos formatos que estiverem (papel, mídia eletrônicas, telefone e até mesmo conversas pessoais) e que estejam disponíveis sempre que necessárias e que sejam confiáveis, (Izquierdo, 2007).

2.1. Principais Conceitos de Segurança da Informação

Para que se possa fornecer efetivamente o que vem a ser segurança da informação, é preciso compreender a alguns conceitos, como **dados, informação e conhecimento.**

Segundo (Dantas, 2011), os **dados** são especificados como a classe mais baixa da informação. Já a **informação,** é quando os dados tendem a passar por algum tipo de processamento para que possam ser acessados e compreendidos facilmente. Tendo sob sua posse a informação é gerado o **conhecimento**, onde este necessita que a pessoa possua experiência além de entender o contexto. Todos os elementos aqui mencionados são constituintes de um sistema.

Para que se obtenha a segurança da informação é necessário que se entenda o que vem a ser um sistema de informações. Segundo, (Stair, 2006) um sistema de informação é um sistema do tipo especializado, onde um há integração de um conjunto de elementos ou componentes inter-relacionados que são capazes de coletar, manipular e difundir os dados e informações, além de oferecer um mecanismo voltado para que o objetivo seja atendido, podendo ser novamente realimentado.

Um sistema de informação pode ser mais bem visualizado através da figura 1, de acordo com a visão dos seus componentes.

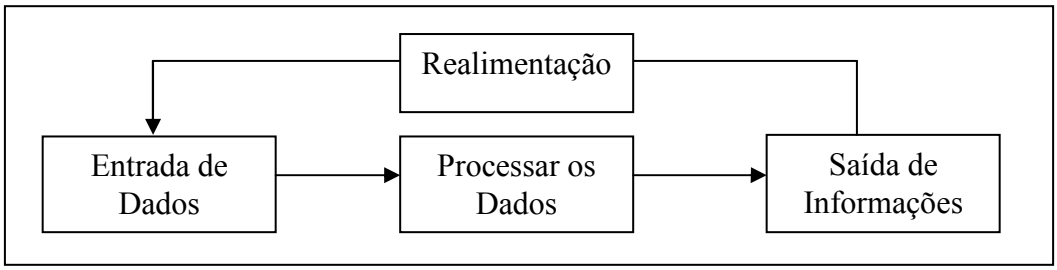

Figura 1 - Componentes de um sistema de informação.
Fonte: Autor. Adaptado de Stair, 2006.

Quando se insere os dados em um sistema qualquer e, necessita usufruir as informações, procura-se interpor que a as informações sejam acessadas de maneira segura apenas por quem é realmente seu proprietário.

Anteriormente a inserção de computadores como ferramentas no auxilio da administração de informações, tudo era feito de maneira física, com diversos papeis e ate mesmo documentos que podiam ser considerados confidenciais, sendo arquivados nas empresas. Pessoas alheias à organização e interessada nas informações infiltravam-se de maneira que se pudesse furtá-la e obter algum proveito sobre a informação obtida, vendendo-a para concorrentes.

Com o ingresso do computador, passou-se a utilizar ferramentas que fossem necessárias para proteção das informações armazenadas no computador. Principalmente quando o sistema pode ser acessado por uma rede de dados ou pela internet. Dado um conjunto de ferramentas arquitetadas para proteger dados e impedir hackers é o que se denomina como segurança de computador (Stallings, 2008).

2.2. Violações da Segurança de informação

Existem na área tecnológica alguns exemplos de violações de segurança, são elas:

a- A transmissão de um arquivo entre dois usuários, contendo informações confidenciais que devem ser protegidas contra divulgação. Sem que uma terceira pessoa não autorizada acesse ao arquivo.

b- Em uma determinada rede, o gerente transmite uma mensagem a um computador que se encontra sobre seu gerenciamento. Esta mensagem serve para que, o computador realize uma atualização nos arquivos de autorização de inclusão de identidades de novos usuários. Um usuário 'X' pode fazer a interceptação desta mensagem, e alterar o conteúdo, para posteriormente encaminhar a mensagem para o computador, fazendo as alterações conforme solicitados.

c- O usuário, ao invés de interceptar uma mensagem, cria sua própria mensagem e as repassa ao computador com se as mesmas fossem oriundas do gerente da rede. O computador aceita e faz as devidas alterações.

d- Uma organização demite um determinado funcionário. O responsável pelo setor pessoal envia ao servidor que o acesso do ex-funcionário no computador deve ser considerado inválido. O ex-funcionário pode fazer a interceptação da mensagem e adia o tempo necessário para que o seu acesso possa ser realizado pelo tempo que lhe aprouver. Esta ação pode passar despercebida pela organização.

A lista de possíveis violações de segurança não cessa e há um crescimento deste rol de problemas relacionados à segurança.

Segundo um relatório emitido pela *Internet Architecture Board* (Conselho de arquitetura de Internet) onde este estabelecia o consenso geral de que a Internet necessita de mais e uma melhor segurança, além de identificar as principais áreas que necessitam mecanismos de segurança (Braden et. al, 1994).

Os muitos ataques ocorridos na internet tornaram-se sofisticados e necessitou aumentar a habilidade e principalmente os conhecimentos, para poderem evitar que sejam causados enormes danos.

Existem alguns conceitos inerentes à área de segurança que foram padronizados pelo Setor de Padronização de Telecomunicação (*Telecommunication* Standardization *Sector*, ITU-T) da União Internacional de Telecomunicação (ITU) que é uma agencia patrocinada pelas Nações Unidas que desenvolve algumas recomendações voltadas a interconexão de sistemas abertos (*Open Systems Interconnection* - OSI).

Pode-se formar como uma definição da segurança da informação através de um conjunto de notas que são compostas de politicas de controle e de segurança, onde sua principal funcionalidade é de proteger a dados e informações inerentes de usuários e organizações para controlar o conteúdo de maneira que não seja exposto e/ou alterado por pessoa que não possua autorização.

2.3. Princípios da Segurança da Informação

A organização Internacional para Padronização (*International Organization for Standardization* - ISO*)* em conjunto com a Comissão Eletrotécnica Internacional *(International Electrotechnical Commission* - IEC), públicaram a ISO/IEC 27002:2005 intitulado *A tecnologia da informação - Técnicas de segurança - Código de prática para a gestão de segurança da informação,* onde este estabelece sobre a Segurança da informação, como: “Preservadora da confidencialidade, da integridade além da disponibilidade da informação”.

Contando ainda com outras propriedades, tais como autenticidade, responsabilidade, não repudio e confiabilidade, para garantir a segurança da informação, onde estes princípios serão elucidados a seguir:

A **autenticidade** de uma informação orienta que todos os usuários devem poder identificar a real origem da informação, onde um intruso não pode inserir uma informação irreal e modifica sua verdadeira fonte de origem.

Quando se pode assegurar que uma informação é confiável por ser procedente e idônea, além de garantir que uma mensagem seja verdadeira, denomina-se de **confiabilidade.**

A **confidencialidade** garante que toda e qualquer informação seja protegida e que esta esteja acessível somente aos usuários que estejam devidamente autorizados a acessá-las.

Para que as atividades desempenhadas estejam de acordo com as normas e padrões elaborados por organizações responsáveis, deve-se ter uma **conformidade** a ser seguida.

Ao se ter uma informação que esteja armazenada em qualquer sistema computacional, essas informações devem estar acessíveis durante todo o tempo que estiver armazenada e somente a quem tiver autorização para acessá-la, sendo assim caracterizado como **disponibilidade.**

Tem-se um princípio que busca saber a veracidade de uma informação, ou seja, a constatação de que a mesma não foi modificada indevidamente, a este principio chama-se **integridade.**

Caso seja necessário se obter uma **investigação** esta terá garantia de ser realizada por pessoas competentes, que seguirão a procedimentos previamente definidos e não prejudicará aos direitos e garantias individuais.

Com o intuito de garantir que todas as ações obedecerão às legislações que são voltadas para segurança da informação, deve-se observar a **legalidade**.

Ainda há um principio da segurança que assegura que um usuário não pode negar-se ter sido o veiculo gerador de uma determinada informação, chama-se **não repúdio.**

Os dados e informações pessoais inseridos devem ser preservados, de maneira que não podem ser vasculhados por outras pessoas, recebe o nome de **privacidade.**

A **responsabilidade** está diretamente ligada a todos que produzem, manuseiam, transportam e descartam informação, com a finalidade de que estas sejam garantidas pelas partes envolvidas no sistema.

Tem-se para a Segurança da Informação alguns princípios complementares que auxiliam nos sistemas partindo do principio da auditoria, da identificação e sigilo.

Quando se relaciona a capacidade existente em um sistema a fim de que o mesmo possa definir o comportamento e as ações de um determinado usuário além de poder identifica-lo, a isto se chama **auditoria**.

O **sigilo** que também pode ser visto como privacidade principalmente quando esta relacionada à confidencialidade e garantia da privacidade de um usuário no sistema.

Para que o usuário possa acessar ao sistema é necessário ter sua **identificação,** onde esta é o meio pelo qual o usuário apresenta a sua identidade sendo esta necessária para estabelecer a autenticação e autorização.

Existem ainda princípios que são de extrema importância para a segurança de diversos sistemas, são a autenticação e autorização para o controle de acesso.

2.3.1. Autenticação

A **autenticação** nada mais é, do que o ato de um determinado usuário confirmar a sua identidade a fim de poder acessar ao sistema, de acordo com seu nível de utilização do sistema. Existem dois grupos de métodos de autenticação existentes: Autenticação de Mensagem e autenticação de usuário.

2.3.1.1. Autenticação de Mensagem.

Responsável por garantir que a mensagem emitida é procedente por quem a emitiu. Assim sendo, há também a verificação do conteúdo da mensagem, para saber se o mesmo foi alterado e caso tenha sido, se foi de modo acidental ou proposital.

Para (Menezes,1997) as técnicas que são mais utilizadas são: MAC (*Message Authentication Code)* e Assinatura Digital.

- A autenticação MAC é um tipo de função unidirecional que recebe a mensagem e uma chave secreta utilizada por aplicações que necessitam manter a integridade dos dados, sem que haja necessidade de manter a privacidade, somente sendo possível saber a saída com a posse da chave.

- A assinatura digital é um meio de autenticação que simula uma assinatura real. A mesma somente pode ser gerada por quem criou a mensagem, mas todos podem identificar quem a emitiu.

2.3.1.2. Autenticação de Usuário.

Serve para assegurar que o usuário é quem ele afirma ser. A confirmação da identidade é realizada no momento em que o usuário se comunica com a parte verificadora do sistema.

Segundo (Menezes,1997) o usuário deve apresentar a algo que o identifique, e que são classificados em três categorias:

1- **Algo que é próprio do individuo.** Todo usuário possui uma característica fisiológica individual, que também é conhecida como, característica biométrica, que pode ser impressão, voz, retina, e outros.
2- **Algo que o individuo possui.** O usuário necessita ter em mãos algum dispositivo que o possa identificar, como um gerador de senhas, uma chave eletrônica ou um *smartcard*.
3- **Algo conhecido.** O usuário deve apresentar alguma informação que somente ele saiba e que seja considerada secreta, como uma senha ou uma chave privada. Esta pode ser demonstrada em três categorias que são dispostas de acordo com o nível de segurança:
   - **a-** É o tipo de autenticação que é mais difundido, ocorre através da utilização de uma senha ou palavra-chave que é associada a cada usuário é conhecida como **autenticação débil.**
   - **b-** Tem-se como uma **autenticação forte** quando a entidade solicitante da autenticação prova sua identidade perante a entidade responsável pela verificação, o solicitante deve demonstrar em frente a entidade verificadora que conhece o segredo sem que seja necessário revelá-lo. É uma espécie de desafio variável que precisa ser respondido.
   - **c- Autenticação de conhecimento nulo**. O zero-knowledge, são responsáveis por permitirem que uma entidade a ser verificada, demonstre o conhecimento de um segredo sem que seja necessário revelar nenhuma informação considerada útil à entidade verificadora. Como exemplos destes se têm os sistemas de prova interativa, onde

precisam que sejam respondidas corretamente algumas perguntas referentes ao segredo.

2.3.2. Autorização

A autorização relaciona-se à concessão de permissões a um determinado individuo que deseja obter acesso um determinado recurso. Esta se encontra ligada diretamente com a autenticação. O usuário, após realizar a validação da sua identidade, para que acesse aos recursos, tem que saber quais as restrições se encontram dirigidas a estes, como por exemplo, o que se pode fazer com esta autorização, o que esta tentando fazer, etc.

O controle de acesso é considerado como mecanismos que tornam limitados o acesso aos recursos, e que a identidade do usuário é o principal meio de utiliza-lo. De acordo com os Pedidos de Comentários (*Request For Comments* - RFC-3127, 2001) o Controle de Acesso baseia-se na autenticação, autorização e auditoria.

Existem diversos mecanismos que são utilizados de maneira que possam realizar as restrições de acesso aos recursos que segundo (Echavarria, 2007) pode ser utilizada somente uma, ou uma combinação das duas seguintes:

- Controle de Acesso Discricionário (***Discretionary Acccess Control – DAC)*** é um mecanismo onde a decisão de controle de acesso fica a cargo do proprietário do recurso, de modo que este decidirá quem pode realizar determinadas ações sobre os recursos. Desta forma um usuário ao ser possuidor de uma permissão a determinado recurso, poderá disponibilizá-lo a outro usuário indiretamente.
- Controle de Acesso Obrigatório (***Mandatory Access Control – MAC)*** este se baseia nas regras que uma autoridade central estabelece. É quem restringe o acesso aos objetos baseando na sensibilidade da informação contida pelo sistema e a autorização dos usuários que precisam ter acesso a estas informações.

(Echavarria, 2007) fala dos mais comuns modelos de controle de acesso, que são:

- **Controle de Acesso baseado em Identidades**, as permissões de acesso estão relacionadas ao nome do usuário, ou melhor, ao

identificador do usuário. Para tanto, existem as Listas de Controle de Acesso que contém os identificadores dos usuários bem como, os seus direitos de acesso a determinado recurso.

- **Controle de Acesso baseado em Regras**. Responsável por restringir o acesso aos recursos sabendo qual a função ou papel do usuário dentro da organização. A cada papel é atribuída uma permissão para acessar a recursos.
- **Controle de Acesso baseado em Atributos.** Onde os privilégios de controle levam em conta o conjunto de atributos do usuário bem como qual politica os determina. É uma junção dos dois primeiros apresentados de maneira que se podem combinar os atributos de sujeitos, de recursos e ambiente.

2.4. Problemas da Segurança da Informação

Mesmo com os princípios básicos e auxiliares da Segurança da informação, surgem alguns problemas que a acometem, e que necessitam serem minimizados. Há inconsistências para definições das mesmas, mas apesar de serem visualizadas como sinonímias apresentam uma diferença significante, que são estas as ameaças, os ataques e as vulnerabilidades.

Em um sistema computacional uma **ameaça** é definida como qualquer evento possível que pode ocasionar a um efeito indesejado nos recursos associados ao sistema. Uma ameaça pode ser vista como algo ruim e que poderia chegar a acontecer.

Já o sistema, sofrendo uma intrusão por intermédio de uma vulnerabilidade, podendo ocasionar a uma ameaça, a isto se denomina **ataque.**

Quando algo maligno pode acontecer ao sistema, isto pode ocorrer por intermédio da **vulnerabilidade.** Para esclarecer melhor, uma vulnerabilidade é uma propriedade do sistema que torna possível que uma potencial ameaça possa ocorrer.

A definição de ameaça que (Campos, 2006) apresenta pode ser definida como as condições que originam imprevistos que comprometem as informações e os diversos ativos existentes procurando a existências das vulnerabilidades para gerar

uma não confidencialidade, indisponibilidade ou ainda comprometer a integridade destes.

As ameaças que acometem aos sistemas podem ser provocadas intencionalmente e podem ser dividas da seguinte maneira:

Caso a ameaça seja decorrente de fenômenos naturais, como enchentes, terremotos, tempestades elétricas, maremotos e aquecimentos, a ameaça é considerada **natural**.

Existe ainda a maneira que a ameaça pode ser ocasionada de maneira voluntaria, ou seja, proposital que pode ser ocasionada pelo homem, considerados hackers, invasores, espiões, ladrões, etc. são as ameaças **voluntarias**.

As ameaças **involuntárias** são causadas pelo desconhecimento, e podem ser acarretadas por erros, faltas de energia, acidentes, etc.

Considera-se identificáveis três tipos básicos de ameaças que buscam medidas para suavizar aos problemas ocasionados, são estas:

° **Vazamento de informações** nada mais é, do que espalhar as diversas informações particulares (pessoais e organizacionais) que são acessadas por pessoas que não possuem autorização para acessa-las.

° **Violação de integridade** é quando uma determinada informação passa a ser modificada de maneira não autorizada, tendo seu conteúdo alterado.

° **Negação de serviço** esta é responsável por impedir que usuários autorizados façam usufruto de qualquer recurso em virtude de apresentar-se bloqueada de maneira maliciosa, não deixando utilizar aos serviços.

Rotineiramente, todos os sistemas encontram-se expostos a diversos tipos de ameaças, necessitando, portanto de ofertar uma segurança as informações.

Caso um sistema seja invadido ou, haja uma parada repentina de determinada empresa poderá ter suas informações comprometidas podendo ocasionar a prejuízos, obtendo informações particulares de seus funcionários bem como da própria organização.

2.5. Conclusão

Neste capitulo vimos que a Segurança da Informação é uma área em constante crescimento e de muita preocupação, uma vez que as pessoas bem como as

organizações tendem a manter dados e, por conseguinte informações peculiares e importantes ao seu proprietário.

O conhecimento de considerações importantes sobre a segurança da informação, onda a partir deste ponto se pode entender o que vem a ser um dado, após visualização do mesmo, constata-se a informação que foi processada e em seguida o que se deve fazer com a informação existente.

Constatou-se que um sistema para manter a segurança das informações, tem que apresentar uma proteção bem assistida e planejada, para evitar que haja violações de segurança, para que não “vaze” informações, nem tampouco sejam transgredidas.

A segurança da informação deve seguir aos princípios básicos e auxiliares que norteiam a segurança da informação, assegurando aos usuários do sistema: autenticidade, confiabilidade, confidencialidade, disponibilidade, integridade, não repúdio, privacidade e outras.

Finalmente, os sistemas de informação podem sofrer problemas que necessitam serem diferenciados entre as ameaças, vulnerabilidades e os ataques. Não se pode planejar que haja segurança das informações, sem que se saiba lidar com as eventualidades que podem ser geradas.

3. GERENCIAMENTO DE IDENTIDADE

As muitas pessoas podem compartilhar o mesmo nome e sobrenome, então outros atributos devem ser adicionados para montar a identificação de uma pessoa, não podendo existir mais de uma pessoa com a mesma identificação.

Há um meio que é usufruído de maneira que possa fornecer informações sobre quem esta o utilizando para um determinado sistema, a este se dá o nome de **identidade.**

Ao sermos requisitados de nossa identificação, esta será utilizada para demonstrar que realmente somos quem afirmamos, esta identificação é denominada de identidade **física**. Ao adentrar-se no mundo digital se faz necessários termos cadastrados uma identificação que faça associação de determinado *nickname* (alcunha) *ou username* a um determinado usuário, a este se chama **identidade digital.**

A utilização da identidade digital esta tomando proporções imensas em virtude do crescimento do mundo digital, para que se possa realizar um determinado negócio digitalmente.

Estando na vida real se faz necessário que apresentemos uma identificação particular. Quando se refere a “vida digital”, nos referimos ao acesso efetuado por intermédio da internet, sendo necessário apresentar a identidade digital.

Segundo (Hovav, 2009) a identidade é considerada digital quando surgem propriedades que são únicas e digitais, e/ou ainda pode se estabelecer fatos que relacione uma identidade a um individuo, ou seja, certificação de que determinado usuário é realmente quem ele garante ser.

O (Ferreira, 2012) tem dicionarizado a palavra identidade, como sendo os caracteres próprios e exclusivos de uma pessoa: nome, idade, estado, profissão, sexo, etc. Esta definição serve para demonstrar amplamente que cada ser humano possui algo que lhe é peculiar e que se chama de atributos.

3.1- Conceitos Básicos

Primordialmente é preciso conhecer a alguns termos que são inerentes ao gerenciamento de identidades.

a- **Sistema** é um conjunto onde há integração entre o software e o hardware, além de armazenar informações sobre entidades, credenciais, identidades e processamento de todas as operações implantadas na vida do sistema. Como exemplo, tem-se uma organização que oferta serviços e armazena informações sobre os usuários.

b- Pode-se considerar diversas exemplificações como **entidade**, no caso pode ser uma pessoa, uma organização, uma rede (por intermédio do serviço ou aparelho). Quando se pensa em entidades associa-as imediatamente as existentes no mundo real. Tem-se as entidades contidas em um ambiente e são capazes de "sentir e agir" como **agentes**, onde estas não são consideradas como entidades isoladas, pois podem interagir com outras entidades.

c- Um sistema possui um código único ligado a uma identidade, onde se denomina com **identificador**. Um sistema pode utilizar exclusivamente um identificador, embora possa ser reaproveitado em diversos sistemas. Um identificador normalmente faz referencia a uma identidade.

d- Para se provar em um sistema uma determinada identidade se utiliza o que se chama como **credencial**. Existem diversos tipos de credenciais, que para serem utilizadas é necessário serem feitas com segurança, realizando a associação entre uma entidade e determinada identidade. Em alguns sistemas podem solicitar as credenciais sendo emitidas como uma espécie de passaporte por terceiros.

e- Uma aplicação *web* que possui o serviço pelo qual o usuário necessita acessar chama-se **Provedor de Serviço - PS** (S*ervice Provider - SP*). É o responsável por realizar a delegação da autenticação à terceiros, que no caso chama-se o provedor de identidade.

f- O responsável pela emissão da identidade de um usuário é conhecido como **Provedor de Identidade - PId** *(Identity Provider- IdP)*. Este por sua vez após realizar a autenticação do usuário, o cede uma credencial que é reconhecida como válida pelo Provedor de Serviço.

A ampliação do uso da internet gera diversos problemas, onde se cita aqui o problema relacionado ao Gerenciamento de identidade onde surgem cada vez mais aplicações como as redes sociais (*facebook, twitter, myspace, etc.*), postagem de vídeos (*youtube, uolvideos, etc.*), comércio eletrônico.

3.2. Sistemas de Gerenciamento de Identidades

Para viabilizar o gerenciamento de identidades aprimoraram-se os Sistemas de Gerenciamento de Identidades – SGI (IMS – *Identity Management System*), que envolvem uma estrutura dentro de uma organização ou várias, que arraiga a confiança entre si e que possuem o intuito de gerenciar identidades.

O SGI realiza a tarefa voltada ao tratamento da confidencialidade e as assinaturas digitais para autenticação. Este é responsável por associar e controlar o acesso aos atributos de uma identidade, pesquisando as informações pessoais que se encontram armazenadas em um determinado banco de dados.

Possui ainda, incluso no sistema as aplicações voltadas ao gerenciamento de identidades que buscam principalmente a realização de firmar a privacidade e permitir aos acessos.

Um SGI atua em vários ramos de transações realizadas eletronicamente, como o *E-health, E-government* e o que é considerado maior, o *E-commerce*.

O *E-health* volta-se a preservação dos dados de pacientes dentro da área da saúde, onde assegura a privacidade da identidade do paciente, bem como aos laudos inerentes a si.

No *E-government* o SGI serve para auxiliar ao governo a executar os procedimentos de negócios por intermédio do meio eletrônico, como declarações de impostos, investigações e outros.

Já o *E-commerce* tem segmentos próprios, que se dividem em *E-banking, E-auction e E-shopping*. O *E-banking* volta-se ao gerenciamento dos endereços onde deve ocorrer a realização as transações financeiras e eletrônicas. *E-auction* (leilão eletrônico) abrange um sistema voltado para a realização de um leilão que ocorra eletronicamente, onde busca preservar a privacidade e o anonimato tanto do cliente quanto do vendedor. Já o *E-shopping* volta-se a venda e compra de produtos

diretamente entre a pessoa que vende o produto quanto quem compra este, realizada digitalmente.

Como todo sistema necessita de alguns requisitos que são importantes para o seu bom funcionamento, o SGI também os busca, são eles: funcionalidade, usabilidade, segurança, privacidade, interoperabilidade e outros.

Existe uma lista de requisitos que um sistema de gerenciamento de identidades tem que conter para que possa atender e assegurar que os usuários possam fazer uso destes sem que afete a segurança das informações que lhe são peculiares. (Damiani, 2003) lista alguns requisitos do SGI:

- Anonimato.

Quando se utiliza um SGI, este deve assegurar ao usuário o direito de permanecer anonimamente, não relacionando a identidade digital com a identidade física, de maneira que possa descobrir seus dados pessoais ou ate mesmo outras identidades digitais. Para garantir o anonimato é que se permite a utilização de alcunhas.

- Gestão de confiabilidade.

Com a existência de provedores de identidade e provedores de serviço, estes devem apresentar uma relação de confiança entre os mesmos uma vez que os domínios são diferentes e que as identidades emitidas em um domínio devam ser aceitas pela outra.

- Interoperabilidade.

As identidades dos usuários devem possuir um formato comum e que possa ser compreendida e legitimada apesar dos diferentes e múltiplos domínios.

- Privacidade.

Quando se trata de privacidade procura-se estabelecer que uma aplicação de gerenciamento de identidade estivesse localizada na máquina do usuário, onde este tem o total controle dos seus dados considerado pessoais e de maneira que garanta os seus direitos digitais, caso não esteja na maquina do usuário e sim em um provedor, este deve seguir a legislação, e fornecer os dados somente caso haja autorização do usuário.

- Mecanismo para revogação de identidades.

Um sistema precisará de um mecanismo onde poderá fornecer um meio para que os próprios usuários possam gerenciar as informações contidas em suas identidades como ab-rogá-la.

3.3. Modelos de Gerenciamento de Identidades

O gerenciamento de identidades apresenta diferentes modelos, de acordo com (Wangham, 2010). Este assegura que os modelos são dispostos de acordo com a sua arquitetura, apresentando-se, portanto como: tradicional, centralizado, centrado no usuário e federado.

- **Modelo Tradicional**

Também conhecido como **Silo** ou **Modelo Isolado**, onde o Provedor de Serviço funciona como Provedor de identidade. Neste modelo o usuário do serviço é o principal responsável por criar e administrar sua identidade. Os usuários apresentam grandes dificuldades quanto à utilização deste modelo, pelo fato dos mesmos terem que gerir diversas contas bem como diversas senhas, muitas vezes não obedecendo às regras de segurança. A figura 2 mostra o modelo em questão.

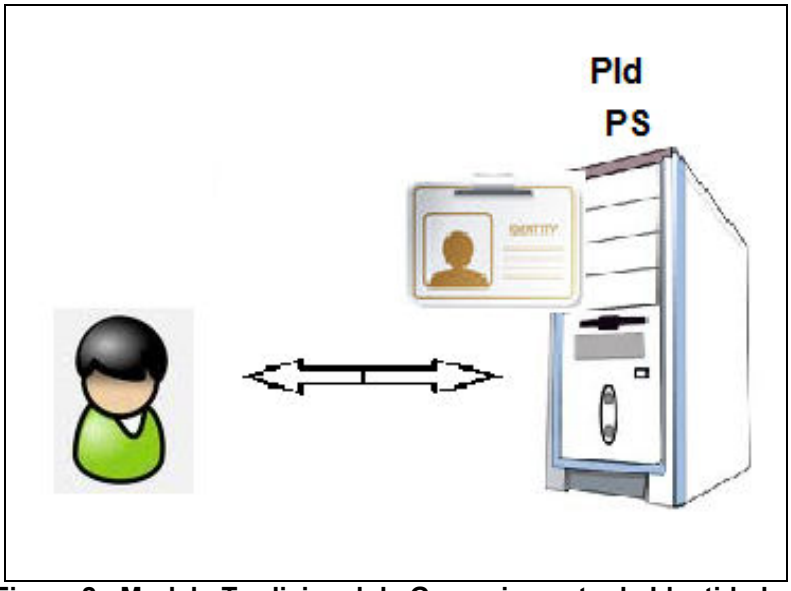

Figura 2 - Modelo Tradicional de Gerenciamento de Identidades
Fonte: Autor. Adaptado de Wanghan, 2010.

- **Modelo Centralizado.**

Como o modelo tradicional não se apresentava muito flexível, surgiu como proposta de solução o modelo centralizado, onde se passa a centralizar tudo em um único Provedor de Identidade, onde todos os serviços o consultam para a realização da autenticação dos usuários. Não é necessário possuir diversas identidades como no modelo tradicional, neste modelo é utilizado o conceito de autenticação única conhecido por *Single Sign-On* (SSO) que lhe dá direito a usufruir de todos os serviços com apenas uma autenticação. Caso o Provedor de identidade seja acometido por alguma falha, todos os sistemas estarão comprometidos. A figura 3 ilustra o modelo Centralizado.

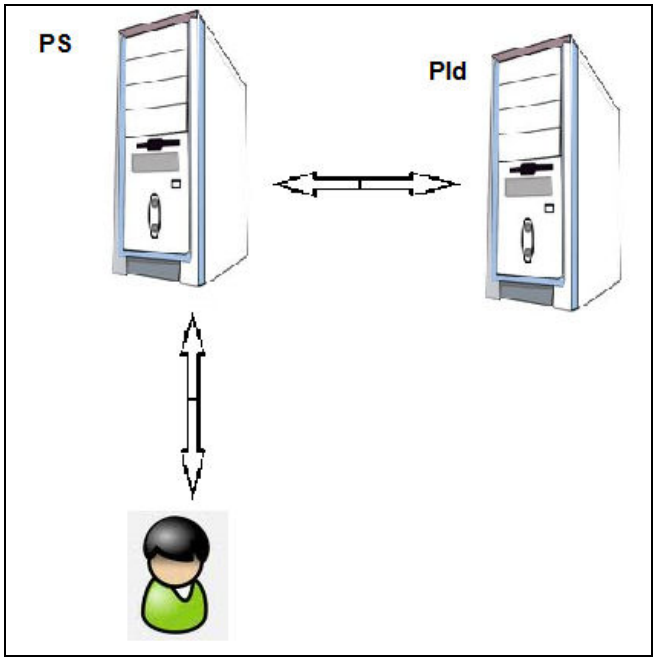

Figura 3 - Modelo Centralizado de Gerenciamento de Identidade
Fonte: Autor. Adaptado de Wangham, 2010.

- **Modelo Centrado no usuário**

Este modelo baseia-se na ideia em que o usuário pode ter o controle total sobre as informações de sua identidade digital. Este modelo solicita que o usuário tenha posse de um dispositivo como celular ou um cartão inteligente (*smartcard)* para realizar a autenticação do usuário e deste ponto repassar as informações devidas aos Provedores de Serviços. A figura 4 mostra este modelo.

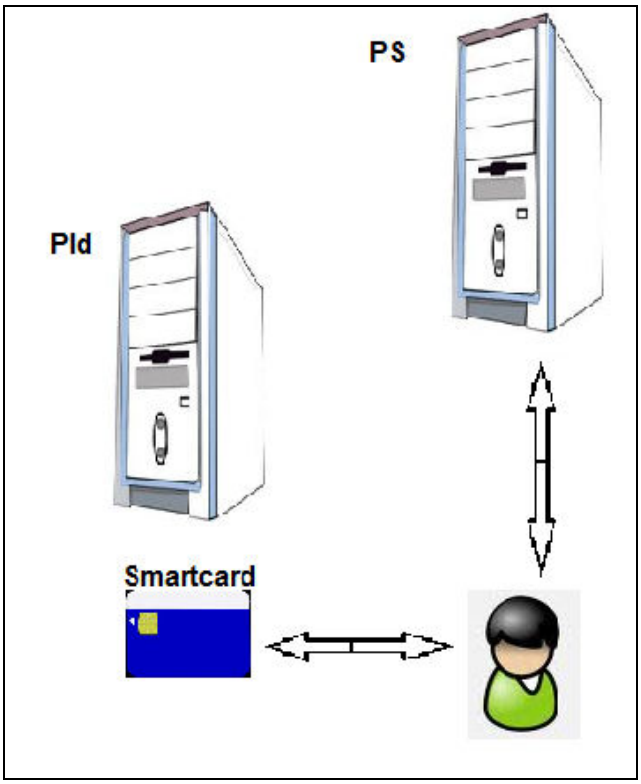

Figura 4- Modelo Centrado no Usuário
Fonte: Autor. Adaptado de Wanghan, 2010.

- **Modelo Federado**

Neste modelo a autenticação do usuário ocorre de maneira descentralizada através de múltiplos provedores de identidades, que se encontram distribuídos em diferentes domínios administrativos. Um domínio administrativo é composto de diversos provedores de serviços enquanto há somente um único provedor de identidade.

Em um determinado domínio um provedor de identidade que possui uma identidade de usuário cadastrado, pode partilhar com outros domínios, desde que tenha uma relação de confiança com outro(s) domínio(s), não sendo necessário ter uma identidade em cada domínio. A figura 5 mostra a relação.

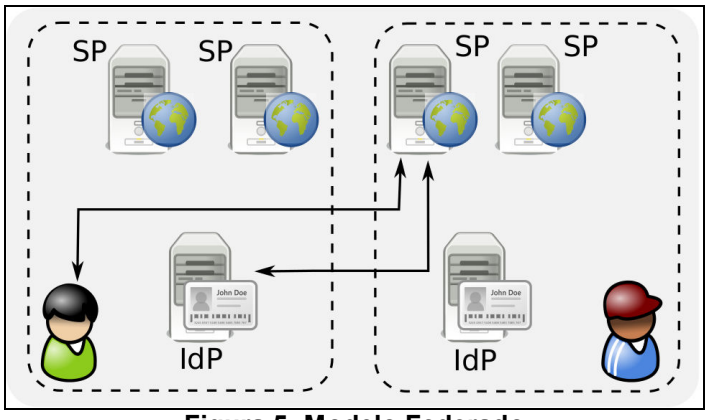

Figura 5- Modelo Federado
Fonte: Retirado de Wanghan, 2010.

3.4. Protocolos de Gerenciamento de Identidades

Para que exista uma federação de gerenciamento de identidades que sejam extremamente compatíveis se faz necessário existir uma normatização na padronização e principalmente dos protocolos. Assim surgiram diversas instituições que são responsáveis por estas normatizações, são conhecidas como consórcios e tem-se como as mais conhecidas: *World Wide Web Consortium (W3C)*, *Internet Engineering Task Force (IETF) e Organization for the Advancement of Structured Information Standards (OASIS).*

Todos estes grupos ao se reunirem realizaram uma convenção, de modo que se padronizassem duas ou mais partes comunicantes de maneira que possa ser realizada uma comunicação entre ambas. Nascendo assim alguns protocolos que são importantes nas soluções de gerenciamento de identidades, principalmente as federadas, como mostradas a seguir.

3.4.1. OpenID

Um protocolo que é capaz de eliminar a necessidade de um usuário possuir múltiplos nomes nos mais diversificados sites, o que para o usuário simplifica muito sua experiência, assim é o OpenID.

OpenID (OpenID, 2013) é uma tecnologia que é totalmente gratuita e não proprietária, onde há permissão de que se possa usar uma conta já existente para ingressar em vários sites, sem que haja a necessidade da criação de novas senhas.

Pode-se fazer a escolha da realização de associação das informações por intermédio do OpenID, podendo partilhá-la com nome ou simplesmente o endereço do correio eletrônico.

Desta maneira, ao utilizar o OpenID, você pode controlar quais informações podem ser partilhadas nos sites visitados. Ao utilizar uma senha, esta é fornecida ao Provedor de Identidade, que efetua a confirmação da identidade para os sites visitados, somente o provedor e nenhum site verá a senha, garantindo a segurança da sua identidade.

Existem muitos sites que aceitam OpenID para efetuação do login, onde se cita o Google, Facebook, MySpace e muitos outros.

O OpenId suporta o Localizador Padrão de Recursos que é mais comumente conhecido como URL - *Uniform Resource Locator* que pode ser conhecido como endereço de web, bem como XRI (*Extensible Resource Identifier* que é um identificador de recurso extensível, que serve para fornecer um formato para identificadores abstratos) como identificadores de usuários, onde podem ser públicos ou privados. A figura 6 mostra o funcionamento do OpenID.

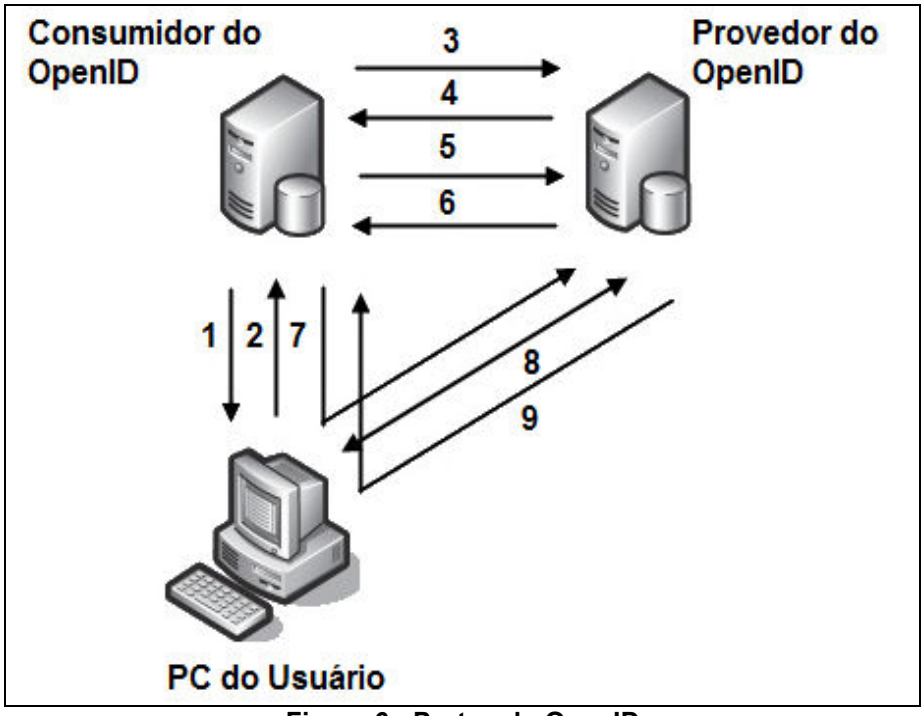

Figura 6 - Protocolo OpenID
Fonte: O autor. Adaptado de Leancode, 2007.

A figura acima apresentada demonstra o fluxo do protocolo OpenId, explanado a seguir:

1- O usuário acessa um site que possui habilitação OpenID, o site envia um formulário ao usuário onde solicita do usuário sua identidade OpenID.
2- O usuário digita a sua identidade, como por exemplo: *mauro.myopenid.com.* Logo após o formulário é enviado ao site.
3- O servidor web acessa http://mauro.myopenid.com.
4- Recupera a localização do provedor do OpenID através da *tag* link:

```
<link rel="openid.server" href="http://www.myopenid.com/server" />
```

5- Um “associado” realiza um pedido por intermédio de mensagens de websites para o provedor. Onde as duas máquinas estabelecem um relação de segurança, fazendo um segredo usando uma chave de troca Diffie-HellMan[1].

```
openid.mode                    Associate
openid.assoc_type              HMAC-SHA1
openid.session_type            DH-SHA1
openid.dh_consumer_public
openid.dh_modulus
openid.dh_gen                  Ag==
```

6- O provedor fornece o site com uma resposta “assoc_handle”, além de validade para ser utilizado em futuras solicitações. A partir deste ponto os dois servidores estabeleceram um compartilhamento de segredos, sem que corra o perigo ter sua segurança quebrada ao ser transmitida.

```
assoc_handle     {HMAC-SHA1}{47b0ec92}{5hMN8A==}
assoc_type       HMAC-SHA1
dh_server_public
enc_mac_key
expires_in       1209600
session_type     DH-SHA1
```

7- A resposta do Consumidor a partir do passo dois contém um direcionamento para o fornecedor, que contém certo numero de parâmetros em uma *string* de consulta. A *String* de consulta contém o estabelecido “assoc_handle” além de um “nonce” ser colocada no “Return_to”.

[1] **Diffie-HellMan** Método de criptografia específico para troca de chaves, onde admite que duas partes que são conhecedoras entre si, que compartilhem a uma chave secreta sob um canal de comunicação considerado inseguro.

openid.mode	checkid_setup
openid.identity	http://mauro.myopenid.com/
openid.return_to	http://openidconsumer.test/cp/login.aspx? **& nonce = vovudmLa**
openid.trust_root	http://openidconsumer.test/cp
openid.assoc_handle	{HMAC-SHA1 47b0ec92} {} {} 5hMN8A ==
openid.sreg.required	sexo, código postal, fuso horário
openid.sreg.optional	e-mail, país
openid.sreg.policy_url	

8- Agora que o usuário se encontra no site do fornecedor, executa o passo para autenticação, que pode ser, por exemplo, digitar sua senha de acesso.

9- O provedor realiza o redirecionamento do usuário para o site do consumidor, em conjunto com os parâmetros na *string* de consulta. Permitindo o usuário a acessar.

nonce	vovudmLa
openid.assoc_handle	{HMAC-SHA1 47b0ec92} {} {} 5hMN8A ==
openid.identity	http://mauro.myopenid.com/
openid.mode	id_res
openid.op_endpoint	http://www.myopenid.com/server
openid.response_nonce	2012-09-12T14:54:53 ZyUUam3
openid.return_to	http://openidconsumer.test/cp/login.aspx? nonce=vovudmLa
openid.sig	EpvWdJtxacv2WtCaZLbud85M84k =
openid.signed	assoc_handle, identity, mode, op_endpoint, response_nonce, return_to, signed, sreg.country, sreg.email
openid.sreg.country	BR
openid.sreg.email	testuser@webmail.com

Esta string de consulta contém uma gama de segurança para evitar que seja atacado. Pode-se ver no “assoc_handle”, onde o consumidor utiliza para saber o segredo que foi estabelecido. Há na assinatura digital os valores dos parâmetros que são listados na “*openid.signed*” que usufrui o segredo estabelecido. Caso um invasor realize a alteração do “*openid.identity*”, para que possa acessar se passando por outra pessoa, a assinatura não apresenta correspondência (o atacante por não possuir o segredo, não pode recriar a assinatura para utilizá-la). No “openid.return” ele apresenta uma única utilização. Caso um invasor queira fazer o reenvio a *string* de consulta mostra através do *nonce* que já fora utilizado.

Somente após a versão 2.0 que dá suporte a formação de federação de identidades.

3.4.2. Security Assertion Markup Language - SAML

Seguindo a padrões baseados em *eXtensible Markup Language* – XML (Linguagem de Remarcação Extensível), que fora desenvolvido pela *OASIS International Consortium* onde consente que um provedor de identidade possa declarar as afirmações sobre as identidades dos usuários dando-lhes seus direitos e definindo seus atributos (Seabra, 2009).

O SAML é utilizado em produtos de cunho comercial, bem como em produtos de cunho acadêmico, onde faz a padronização de *tokens* de segurança e realiza as definições das mensagens que foram usadas para realização de solicitações, emissões além de troca de dados de autenticação e autorização.

A possibilidade de fazer asserções da SAML (OASIS, 2005) torna possível a propagar declarações de segurança capazes de solucionar problemas de comunicação em diversos domínios que fazem o SSO. A figura 7 ilustra um exemplo de asserção SAML.

```
1   <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
2         Version="2.0" IssueInstant="2012-09-12T14:45:00Z">
3           <saml:Issuer Format="urn:oasis:names:SAML:2.0:nameid-format:entity">
4                   http://labsac.ccet.ufma.br
5           </saml:Issuer>
6              <saml:Subject>
7                 <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:namid-format:emailAddress">
8                           labsac@ccet.ufma.br
9                 </saml:NameID>
10             </saml:Subject>
11                <saml:Conditions NotBefore="2012-09-12T14:45:00Z"
12                 NotOnOrAfter="2012-09-12T14:55:00Z" />
13           <saml:AuthnStatement AuthnIsntante="2012-09-12T14:45:00Z"
14                 SessionIndex="75675698923">
15             <saml:AuthnContext>
16                     <saml:AuthnContextClassRef>
17                         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
18                     </saml:AuthncontextClassRef>
19              </saml:Authncontext>
20            </saml:AuthnStatement>
21  </saml:Assertion>
```

Figura 7 - Asserção SAML
Fonte: Autor

No exemplo dado, aparecem as principais informações de uma SAML, que são:

- **Radical**. É a declaração dos nomes da SAML (*saml:Assertion)*, a versão utilizada (*Version*) e o momento da emissão pela autoridade (*IssueInstant*), podendo ser visualizado nas linhas 1 e 2.

- **Emissor** ou ***Issuer***. É onde apresenta as informações inerentes acerca da entidade solicitante, encontrado na linha 3 a 5, neste caso *http://labsac.ccet.ufma.br.*
- **Assunto** ou ***Subject***. Representa um conjunto de condições que devem ser levadas em conta ao avaliar a validade de uma afirmação SAML, onde este é opcional, da linha 6 a 10, no caso o e-mail é de posse *labsac@ccet.ufma.br*.
- **Particularidade** ou ***Conditions***. Representa um conjunto de condições que devem ser levadas em conta ao avaliar a validade de uma afirmação SAML, o prazo da validade se encontra nas linhas 11 e 12, onde consta entre as 14h45min e 14h55min, datado de 12-09-2012.
- **Afirmação de autenticação** ou **Authentication statement**. É criada pela entidade que efetivou a autenticação do usuário, no caso da linha 13 a 20, é o elemento saml:AuthnStatement. Neste trecho é indicada se o usuário conseguiu efetuar sua autenticação além de citar o mecanismo que foi utilizado para realizar o transporte está protegido por senha, das linhas 15 a 19 pelo elemento AuthnContext.

3.4.3. eXtensible Access Control Markup Language - XACML

É um padrão criado pela *OASIS*, que se baseia no XML, onde serve para assegurar a interoperabilidade no processo da autorização que é uma propriedade básica da segurança (OASIS, 2005).

Este serve para descrever uma estrutura aonde define as politicas de controle de acesso, bem como, definir um formato para as mensagens de pedido e de resposta. A Política de controle de acesso deve ser bem definida com o intuito de definir sobre quem possui direitos da acessar, e a o que este pode acessar. Já as mensagens são descritas padronizando a maneira que deverão ser realizadas o pedido e como deverão ser as respostas.

As mensagens que tem formato de pedido e resposta são definidas pela Políticas de Pontos de Decisão (*Policy Decision Point* - PDP), local que efetua o processamento da política e ainda a Política de Ponto de aplicação (*Policy*

Enforcement Point – PEP) que é o responsável por concretizar as decisões de politicas (Seabra,2009).

Desenvolvido com a finalidade de garantir que houvesse interoperabilidade em diversas aplicações, a figura 8 mostra um exemplo do XACML.

```
<soapenv:Envelope
   xmlns:soapenv ="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <xacml-samlp:XACMLAuthzDecisionQuery
         xmlns:xacml-samlp="urn:oasis:xacml:2.0:saml:protocol:schema:os"
         ID="_e064bd912f83c1544fea110307000acf"
         IssueInstant="2007-05-21T22:00:36Z"
         Version="2.0">
         <xacml-context:Request
            xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os">
               <!-- See [XACML-Request-01] for sample content of this element -->
         </xacml-context:Request>
      </xacml-samlp:XACMLAuthzDecisionQuery>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 8 – XACML
Fonte: John, 2008.

Um pedido também conhecido como *request* tem em sua composição:

- Atributos associados ao usuário de onde a requisição esta sendo gerada;
- Identificações de recursos desejáveis;
- Ações que serão executadas pelo recurso;
- Atributos do ambiente.

Em uma resposta, também conhecida como *response*, é composta por:

- *Allow* ou *permit* que garante o acesso, ou permite o acesso.
- *Deny* que nega o acesso.
- *Not applicable* no caso há falta de políticas ou regras associadas ao recurso.
- *Indeterminate* caso durante o processamento apresente a erros.

A figura 9 demonstrará o fluxo de interações do XACML.

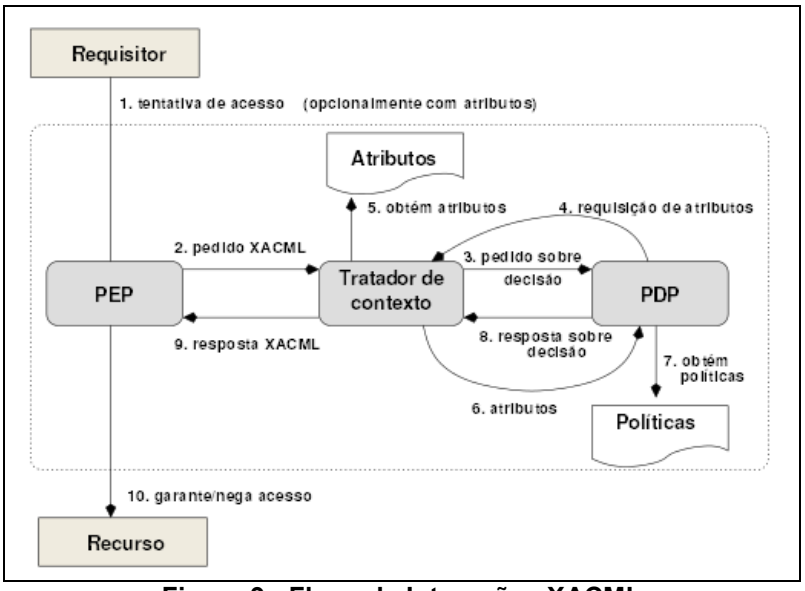

Figura 9 - Fluxo de Interações XACML
Fonte: Mello, 2006

A figura explana o fluxo de dados onde um cliente tenta acessar a um recurso em que utiliza o XACML.

1- O usuário tenta acessar ao recurso e envia ao PEP um pedido no formato XACML.
2- O PEP por sua vez o encaminha ao tratador de contexto.
3- O tratador de contexto encaminha a solicitação do pedido para o PDP, onde o mesmo tome uma decisão sobre a tentativa de acesso.
4- Logo em seguida o PDP pode solicitar ao tratador de contexto informações dos atributos dos sujeitos e dos recursos.
5- O tratador de contexto obtém os atributos.
6- O PDP recebe do tratador de contexto os atributos.
7- Assim que os PDP encontram-se de posse dos atributos, faz a solicitação das políticas que estão associadas com as entidades abrangidas.
8- De posse dos atributos gera uma resposta acerca da decisão a ser tomada.
9- Há uma geração de resposta em XACML pelo tratador de contexto e a envia ao PEP.
10- Finalmente, o PEP de posse das informações assegura o acesso ao recurso solicitado ou o nega.

3.4.4. X.509

X.509 é um padrão que foi públicado inicialmente pela União Internacional de Telecomunicações (*ITU)* e que passou a seguir a uma padronização oriunda da Organização Internacional para Padronização (ISO), onde o referido faz utilização de chaves assimétricas, ou seja, possui um par de chaves (públicas) que servem para criptografar um pacote de informações.

Para a realização de uma autenticação do certificado X.509 existe uma autoridade certificadora que os emite. Na sua composição existem duas chaves: sendo uma chave pública e a outra privada.

A base deste modelo de autenticação é a confiança entre o sistema que garante a autenticidade do usuário com a autoridade emissora do certificado.

O certificado X. 509 tem sua estrutura alguns campos que se diferenciam entre si de acordo com as versões, apresentando especificidades relativas a cada versão.

- **Versão:** é responsável por indicar a versão do certificado.
- **Número Serial do Certificado:** Número único atribuído pela entidade que realiza a emissão do certificado.
- **Identificador de Algoritmos de Assinatura:** Contém o identificador do algoritmo criptográfico responsável por gerar uma função hash[2], que é utilizada pela Autoridade Certificadora (AC) para assinar o certificado.
- **Nome do Emissor.** Identifica a entidade que assinou e emitiu o certificado.
- **Validade.** É considerado o intervalo de tempo garantido pela AC, onde mantem as informações sobre o estado do certificado (data e horário do inicio e do final).
- **Nome do sujeito:** É a identificação da entidade que se encontra associada a uma chave pública que esta localizada no campo chave pública do certificado.

[2] **Função *hash***. É qualquer algoritmo que realiza um resumo de dados, sendo capaz de transformar uma grande quantidade de dados em uma pequena quantidade de informações e que tem como sua principal aplicação de realizar a comparação de dados grandes ou secretos.

- **Informações sobre a chave pública do sujeito.** Informações sobre a chave pública além do algoritmo do proprietário.
- **Identificador Único do Emissor:** É o campo utilizado para identificar o nome do emissor, onde pode haver necessidade de usufruir novamente o nome.
- **Identificador Único do Sujeito:** Identifica o nome do sujeito, caso também seja necessário reutilizar os nomes.
- **Extensões:** Inseridos nos certificados, podem incluir opções de acrescentar informações adicionais, de maneira que possam, por exemplo, realizar a identificação mais segura do sujeito do certificado, sobe a validade e principalmente acerca da autorização.

A figura 10, apresenta a estrutura de um certificado X.509.

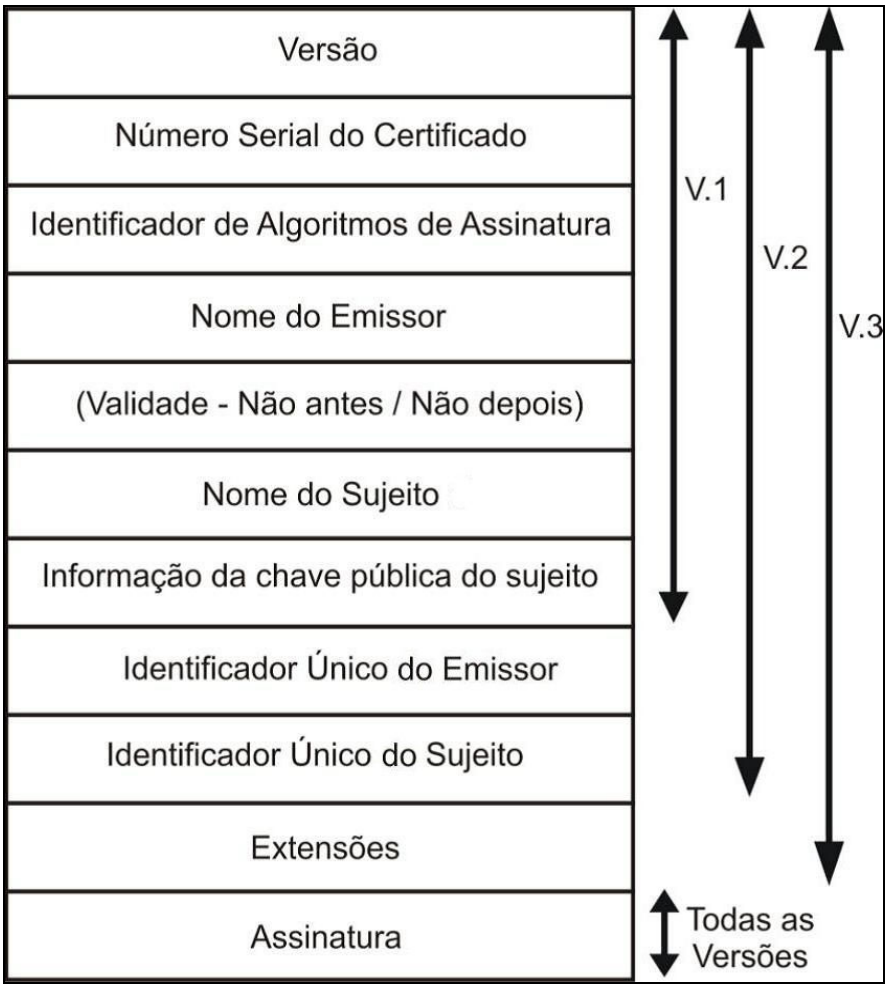

Figura 10 - Certificado X.509
Fonte: Burnett, 2002

3.4.5. WS-Trust

Como uma aplicação voltada para segurança em mensagens do tipo *WS-Security*, o *WS-Trust* nasceu para complementa-lo de maneira a definir como duas entidades confiáveis, criado pelo OASIS.

A *WS-Trust* define um modelo confiável onde caso um usuário não possua as devidas credenciais que são solicitadas por um provedor de serviço, pode realizar a solicitação de credenciais para uma autoridade emissora de certificado chamada de Serviço de segurança de *token* (*Security Token Services-STS)* que pode emitir, trocar e validar credenciais.

No caso de necessitar efetuar a comunicação do WS-*Trust* com STS são definidos um protocolo de solicitação e resposta, onde apresentam as solicitações de *token* de segurança (*Request Security Token-RST)* e solicitação de reposta de *token* de segurança (*Request Security Token Response- RSTR)*.

O WS-*Trust* apresenta um mecanismo que realiza desafio entre usuários e os STS, a fim de atender aos pedidos e respostas. As interações realizadas entre os clientes e o STS, podem ser vista como exemplificadas na figura 11.

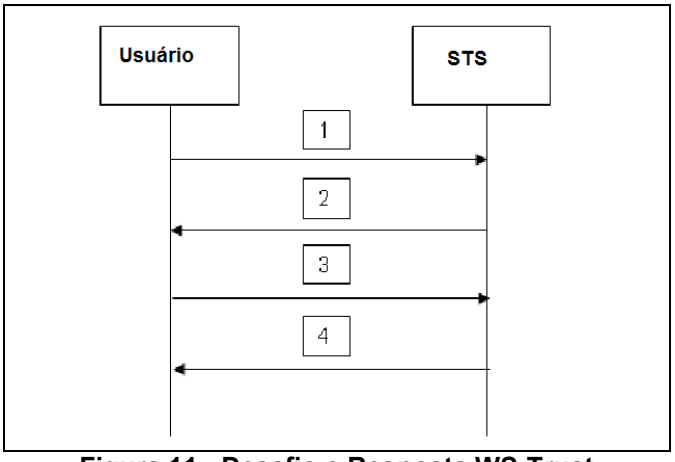

Figura 11 - Desafio e Resposta WS-Trust
Fonte: o Autor

A relação de interações será explicada sequêncialmente de acordo com o seu funcionamento.

1- O cliente solicita uma credencial através da solicitação de *token* de segurança.
2- É enviado um desafio ao usuário através da RSTR.
3- O usuário emite uma resposta do desafio ao STS, com RSTR.
4- Após confirmação do desafio a credencial solicitada é enviada ao usuário.

3.4.6. *Kerberos*

É um protocolo de autenticação que foi desenvolvido pelo Instituto de Tecnologia de Massachussetes (*Massachusetts Institute of Technology – MIT*) que utiliza criptografia simétrica entre o cliente e o servidor, de acordo com o (MIT, 2012). O *Kerberos* segue um padrão da *IETF*, onde é padronizado pela RFC – 4120.

Segundo (Camargo, 2006) o *Kerberos* possui uma autenticação realizada centralizada enquanto a autorização é descentralizada. Tem como seus componentes o Serviço de Autenticação (*Authentication Service – AS*) e o Ticket de Concessão de Serviços (*Ticket Granting Services – TGS)*, onde estes se encontram inseridos no Centro de Distribuição de Chaves (*Key Distribution Centre – KDC).*

Uma vez realizada a autenticação, estes recebem um *Ticket Granting Ticket – TGT* que é visto como uma credencial onde apresenta uma validade e realiza a solicitação ao TGS de uma credencial para realizarem a comunicação com os serviços ao qual se deseja utilizar. Assim sendo o serviço fica responsável por verificar se o usuário pode ou não usufruir ao serviço.

A figura 12 serve para mostrar como é o processo de autenticação realizado pelo *Kerberos*.

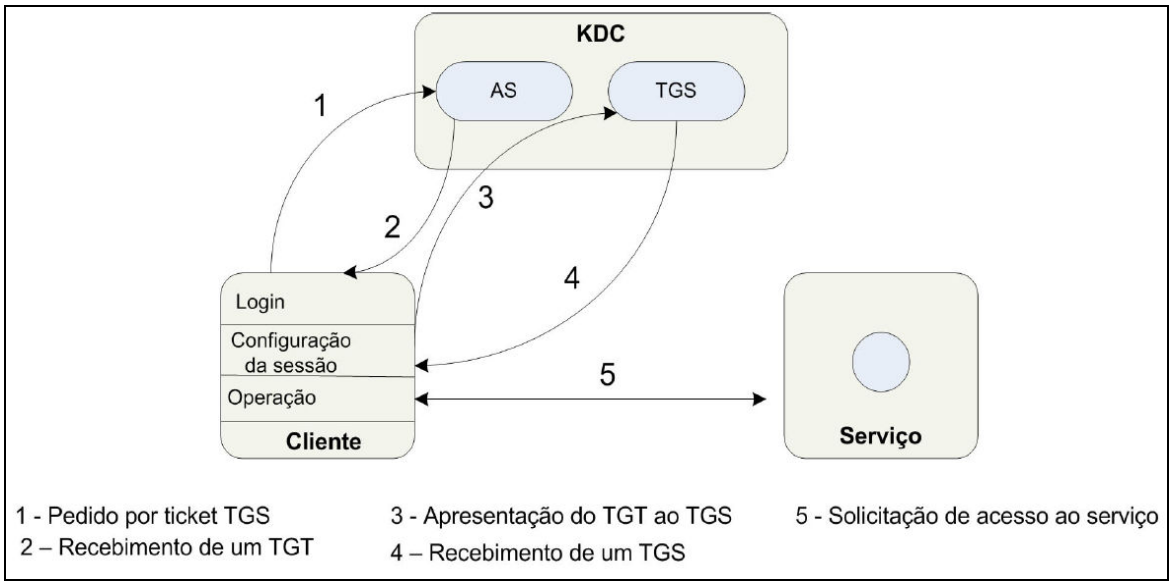

Figura 12 - Processo de Autenticação Kerberos
Fonte: Camargo, 2006.

Observando a figura 12, podemos sequênciar da seguinte maneira o processo de autenticação realizado pelo *Kerberos:*

1- O usuário emite uma solicitação para acessar o serviço ao AS e prepara uma chave criptografada que utiliza o *hash* da senha do usuário.

2- O usuário recebe o TGT para que possa realizar a comunicação com o TGS. O AS realiza a criptografia da chave de sessão com o *hash* da senha do usuário.

3- Neste momento o usuário requisita novamente ao TGS criptografado com a chave de sessão que fora definida, no passo dois, pelo AS.

4- O TGS agora retorna ao cliente um novo TGT possuidor da chave de sessão criptografada com uma chave privada do serviço que deseja ser usufruído.

5- Após ter recebido o TGT, o usuário inicia a comunicação com o serviço. Logo depois que o serviço recebe a identificação do usuário, é que toma a decisão de liberar o acesso ou não, caso haja permissão para o mesmo.

3.5. Estado da Arte

Nesta seção serão vistos os trabalhos pesquisados como soluções de gerenciamento de identidades que serviram para consolidação de um modelo de gerenciamento de identidade, foram analisadas por artigos, dissertações e teses que foram formuladas dentro de instituições de ensino superior e são voltadas para o meio acadêmico. Fora das instituições acadêmicas dentro do mercado há diferenças nas diversas soluções que sempre primam por algo que as torne diferenciada e que possa sobrepor sobre as demais.

3.5.1. Transposição de autenticação de identidades Federadas em Serviços *Web*

(Camargo, 2006) sugeriu como proposta um modelo arquitetural onde a utilização de identidades dava suporte a qualquer das tecnologias de segurança usufruídas para os membros da federação. Para validar o modelo sugerido (Camargo,2006) confeccionou um protótipo de acordo com a arquitetura por ele proposta.

O trabalho foi desenvolvido através de Linguagem de programação *Java*, ainda apresentando padrões abertos como XML-Security e ainda o *Serviço da Web Segurança para Java (Web Service Security for Java – WSS4J).* A figura 13 serve para ilustrar ao trabalho referenciado.

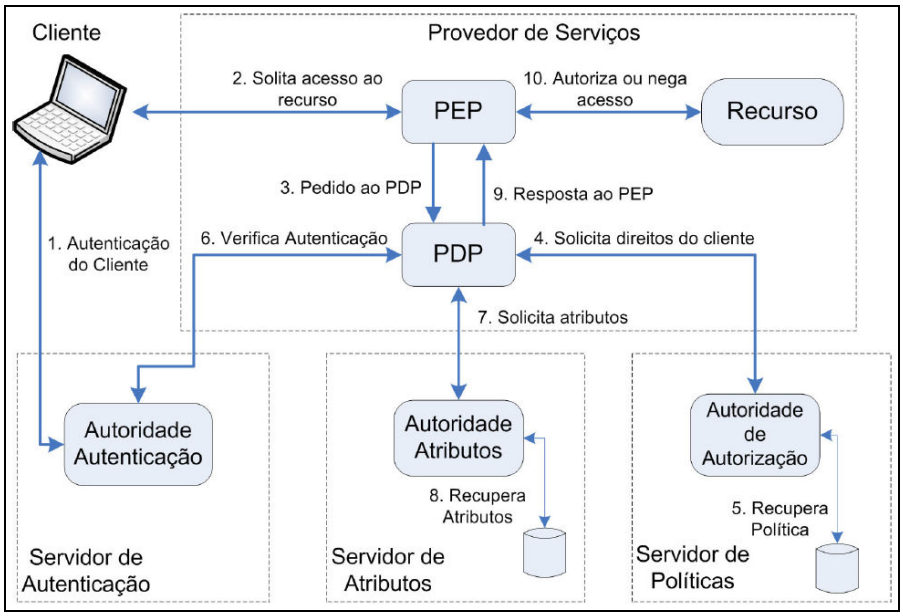

Figura 13 - Processo de Autorização
Fonte: Camargo, 2006

A figura embora complexa segue obedecendo a sequência de passos conforme relacionados a seguir:

1- Ao realizar a autenticação o cliente recebe um *token* configurado no tipo *SAML* que pode ser convertido para qualquer outro tipo de *token* que seja utilizado pelos provedores de serviços.

2- No passo seguinte o cliente pode acessar o provedor de serviços e tem o serviço desejado. Dentro do serviço a requisição é encaminhada ao PEP onde tem a responsabilidade de aplicar as politicas.

3- Porém antes que sejam aplicadas as politicas, o PEP procura o PDP para obter a um parecer sobre as politicas que podem ser aplicadas.

4- As politicas de acesso são sondadas pela Política Ponto de Acesso (*Policy Access Point* - PAP), após a sondagem o PDP faz a tradução do *token* SAML enviado pelo cliente para o utilizado pelo Provedor de serviços.

5- O Serviço de Autenticação que se encontra conectado diretamente ao provedor de serviços é responsável por realizar a tradução, além de validar o *token* que foi enviado pelo PDP. Após obter a validação do *token* é preciso que seja acompanhado de informação dos usuários, que pode ser obtido consultando-se a Política de Ponto de Informação (Policy Information Points - PIP).

6- O PIP pode realizar uma averiguação no Serviço de token de Segurança de um provedor de identidade (S*ecurity Token Service / IdP)* onde o Cliente realiza sua autenticação.
7- Após conseguir obter todas as informações inerentes à utilização do sistema o PDP estabelece uma resposta para o PEP.
8- E finalmente o PEP toma a decisão sobre a autorização ou negação do acesso ao Provedor de serviços.

O trabalho em questão apresentou um serviço voltado principalmente para tradução das credenciais através de um serviço conhecido como (*Credential Translation Service - CTS*), onde inclui os componentes PAP e PIP que são oriundos do XACML.

(Camargo, 2006) exibiu um modelo que permitiu realizar autenticação no domínio de origem e acessar a recursos que se encontram espalhados em domínios diferentes mas, que se encontram dentro das relações de confiança. Assim, procurou basear-se nos Serviços de diferentes tecnologias onde solicitasse que os serviços de segurança permitissem que fosse realizada a transposição de autenticação bem como dos atributos.

5.1.2 - Serviço de Gerenciamento de Identidades para computação em Nuvem

Outro trabalho que evidencia gerenciamento de identidades foi o escrito por (Anselmo Junior, 2011), onde busca propor um gerenciamento de identidades federadas como um serviço, assim criou uma Identidade como um Serviço (Identity as a Service – IdaaS).

Quando se trata de observar ao modelo de gerenciamento de identidades realizado tradicionalmente e o proposto para a computação nas nuvens constata-se que não apresenta diferença nenhuma do modelo proposto.

Seguindo aos padrões de gerenciamento de identidades ocorrido em uma federação de nuvens computacionais, (Anselmo Junior, 2011) definiu requisitos específicos para representação do cenário proposto, além de trabalhar com SSO que é um padrão para trabalhar em federação.

O trabalho procurou sobrepor requisitos para que a identidade seja gerenciada dentro de organizações que possuem uma relação de confiança federativa, são eles:

a- O IdaaS deve ser ofertado pela nuvem. O motivo no qual este deve ser ofertado pelo provedor da nuvem, se dá em decorrência de que o provedor da nuvem já possui os acordos de confiabilidades e contratos de toda estrutura disponível.
b- Outro requisito esta relacionado aos mecanismos de provisionamento de identidades que se relacionam ao ciclo de vida da Identidade como serviço.
c- Deve-se ter ainda uma classificação das informações relacionadas aos níveis de acesso que os usuários tenham para acessar aos provedores.

Para elaboração deste projeto foi utilizado uma solução de controle de acesso conhecido como OpenAM externamente para que o processo de autenticação da nuvem fosse substituído pelo mesmo. O OpenAM baseia-se no OpenSSO e apresenta um suporte a SAML bem como ao *WS-Federation*.

O trabalho apresentou uma funcionalidade bem simplória e que se encontra voltado apenas para o meio acadêmico, não estando, segundo (Anselmo Junior, 2011), preparado para utilização no mundo real.

5.1.3 – Gerenciamento de Identidades com Privacidade do Usuário em Ambiente Web

O foco deste trabalho está no gerenciamento de identidades que é centrado no usuário, onde teve com o objetivo de assegurar que o usuário tenha seus atributos privados sem que se possam ter comprometidos e possam ser usufruídos os serviços. Desta forma o usuário se manteria no anonimato quando qualquer serviço fizesse uma solicitação de um atributo específico (Sakuragui, 2012).

A tese em questão foi voltada de maneira simplificada onde utiliza o *OpenID.* OpenID é um protocolo de gerenciamento de identidades federadas, e é amplamente adotado na internet e usa o SSO para depois modificá-lo para o *NibbleID* que realiza o endereçamento dos objetivos da privacidade.

(Sakuragui, 2012) apresentou em seu trabalho a arquitetura simplificada do OpenID, para que depois fossem modificadas para o *NibbleID* onde garantissem

alcançar os objetivos. Seu objetivo é permitir que um usuário pudesse realizar a comprovação dos atributos pertencentes a sua identidade para um serviço de maneira que preservasse a sua privacidade.

Os sistemas de gerenciamento de identidades apresentam em sua arquitetura tipicamente três elementos: o cliente, o serviço e o provedor de identidades. A arquitetura do *NibbleID* realizou uma cisão nas funções para o provimento de identidades, onde terá o provimento de pseudônimo e provimento de credenciais. Onde se definiu dois elementos para seu utilização, a primeira é do *IdP* que foi utilizado para o provimento de pseudônimos, a segunda é o Provedor de Credenciais (*Credential Provider – CP)* que será o provedor de credenciais.

A arquitetura foi confeccionada de maneira que as duas entidades estejam alocadas em organizações diferentes, e que o usuário tem como optar pelo provedor de Identidades existente na internet, independente das entidades que se encontram capacitadas a certificarem seus atributos.

Segundo (Sakuragui, 2012) o NibbleID encontra-se no meio de duas soluções de Sistema de Gerenciamento de Identidades consideradas opostas, e que servem para comprovar os atributos além de oferecer privacidade.

Quando se usa um ambiente *Web*, faz-se a utilização de diversos navegadores em quaisquer dispositivos, do que possui maior capacidade de processamento ao que possui o seu processamento limitado, sugerindo que haja mecanismos bem simples para realizar criptografia. Outro ponto de vista que foi levado em conta neste trabalho, foi o do anonimato.

3.6. Conclusão

Neste capitulo vimos que para se acessar a alguns serviços computacionais é preciso possuir uma identidade, e que há meios que se encontram voltados para realização do gerenciamento destas.

Visualizadas as principais concepções de algumas peças fundamentais no gerenciamento de identidades.

Apresentaram-se os modelos de gerenciamento de identidade, que servem como base na procura de implantar a que melhor se adeque ao seu modelo, seja ele tradicional, centralizado, centrado no usuário e federado.

O gerenciamento de identidades deve seguir a normas e padrões específicos, e utilizar protocolos que seja mais viável na sua utilização. Tendo visto os principais protocolos que são utilizados no mercado, bem como no meio acadêmico.

Foram analisadas algumas pesquisas existentes na literatura nesse contexto de gerenciamento de identidades na computação nas nuvens, em ambientes federados e em serviços web, além de contornar aos problemas de identidades.

4. COMPUTAÇÃO NAS NUVENS

No ano de 2006 o Diretor Executivo da Google, Eric Schmidt, ao discorrer sobre a descrição dos serviços inerentes da empresa, proferiu pela primeira vez o termo Computação nas Nuvens. Apesar de ter sido usufruído posteriormente pela *Amazon* ao efetuar o lançamento do seu serviço, este termo foi difundido amplamente por um artigo públicado na edição da *Magazine Wired* do mês de outubro, designado de “As fábricas de informações” (*The Information Factories*) escrito por George Gilder, segundo a (CONSEGI, 2010).

O Instituto Nacional de Padrões e Tecnologia (National Institute of Standards and Technology - NIST) desenvolveu um documento em favor das responsabilidades legais sob Lei Federal de Gestão de Segurança da Informação (Federal Information Security Management Act - FISMA) de 2002, a Lei Pública 107-347, onde se incluem definições e diretrizes sobre a Computação nas Nuvens.

A computação nas nuvens é um modelo para permitir o acesso à rede ubíqua, de acordo com a demanda, onde pode ser acessado um pool de recursos computacionais que são configuráveis de maneira compartilhada (exemplificando-se tem rede, servidores, serviços, aplicações e o armazenamento) que podem ser rapidamente provisionados e liberados com um mínimo de esforço no gerenciamento e/ou até as interações com o provedor de serviços (Mell, 2011).

Em geral, o termo computação nas nuvens é usado pelas pessoas atualmente para rotular os muitos serviços acessados por intermédio da internet. Ao usufruir ao processamento, armazenamento e a aplicações online, onde outrora, tudo ocorria nos computadores pessoais (Personal Compute - PC), pode ser vista como sendo realizada “na nuvem”.

Este modelo computacional foi desenvolvido objetivando a fornecer serviços, por intermédio da internet, sendo este acesso realizado de um modo fácil, com custos baixíssimos e que garanta a disponibilidade.

A designação de computação nas nuvens é basicamente derivada fazendo a associação do nome às figuras de rede onde usualmente aparece a demonstração da internet como sendo uma nuvem.

4.1. A Computação em Nuvem e suas características essenciais

As soluções da computação nas nuvens vêm a oferecer algumas características essenciais que são consideradas como “vantagens”. Estas características validam a definição da computação nas nuvens e a distingue de outros modelos. A seguir serão relatadas essas características (CSA, 2009), conforme a figura 13.

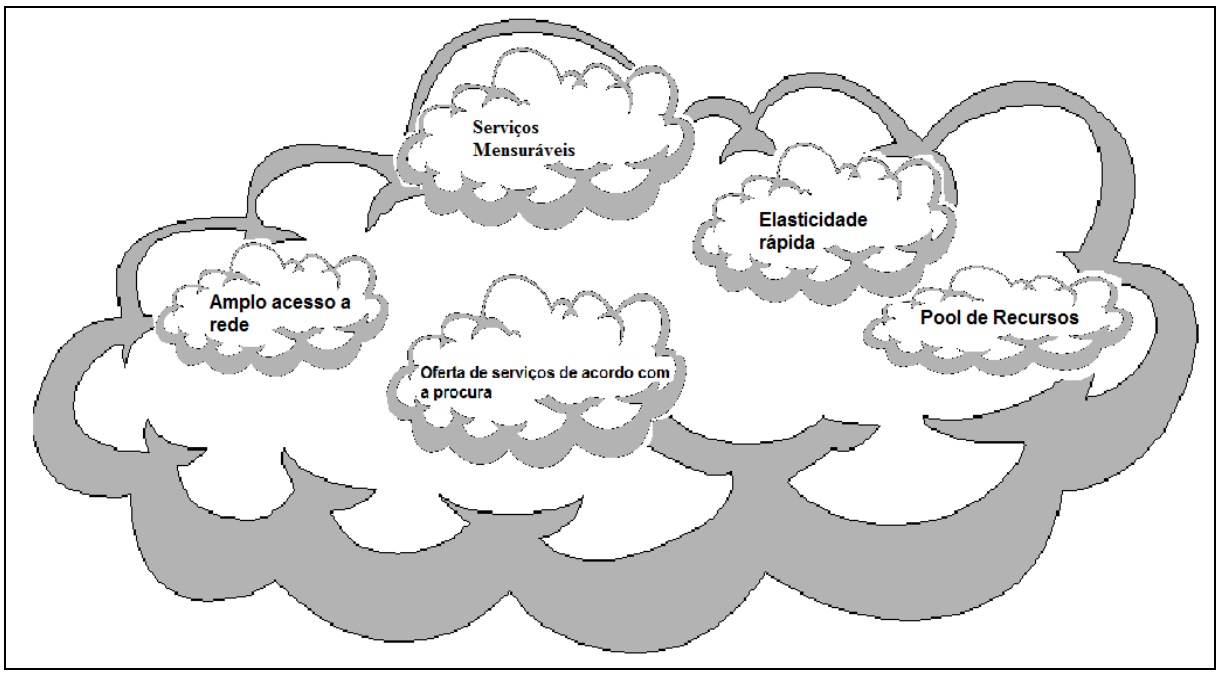

Figura 14 - Características da Nuvem
Fonte: o Autor

4.1.1. Oferta de serviços de acordo com a procura – (*On-demand Self-Service*)

A computação em nuvem oferece recursos *sob demanda,* ou seja, quando o consumidor desejar. Um usuário ao usufruir os serviços da nuvem, pode necessitar, realizar o aumento ou a diminuição das capacidades computacionais (relativas ao hardware e software) alocados, tudo isso sem ser necessário que exista uma interação do homem junto ao provedor de serviços.

4.1.2. Amplo acesso a rede (Ubiquitous Network Access)

O acesso à rede ocorre de maneira ampla, mostrando assim que os serviços da nuvem são acessíveis independentes da plataforma utilizada, sendo ou não heterogêneas.

4.1.3. Pool de Recursos (*Resource Pooling*)

Os recursos computacionais da nuvem são agrupados de maneira a atender a diversos usuários. Os recursos virtuais são dinamicamente atribuídos ou desatribuídos pelo cliente de acordo com sua demanda ou devolvidos pelos usuários de acordo com sua demanda. O usuário não sabe especificamente, onde se encontra a real localização dos recursos que estão sendo utilizados, caso necessário saberá somente o país, ou *DataCenter.* Os recursos incluem o armazenamento, processamento, memória, as máquinas virtuais, e outros.

4.1.4 – Elasticidade rápida (R*apid Elasticy*)

A capacidade que um sistema possui de alocar mais ou menos recursos no momento em que for necessário e de maneira ágil, é o que se define como elasticidade. O usuário tem a visão de que a nuvem parece ser infinita, pelo motivo de poder adquirir mais ou menos poder computacional para suas aplicações de acordo com suas necessidades. O que serve como uma grande ajuda na computação em nuvem é a virtualização, onde se podem criar várias instâncias de recursos solicitados usufruindo apenas um recurso real. A virtualização é a criação de ambientes virtuais, onde se faz a abstração de características físicas do hardware, em que este pode emular vários sistemas operacionais em uma única plataforma computacional.

4.1.5 - Serviços Mensuráveis (Measured Service)

Na nuvem todos os serviços são automaticamente controlados e monitorados, de maneira que todos os que precisam do serviço, seja para ofertar ou para consumir o serviço, visualizam-no transparentemente. Para que haja uma garantia da qualidade de serviço (*Quality of Service - QoS)* é utilizada uma abordagem que

se baseia no acordo de nível de serviço (*Services Level Agreement - SLA)* onde este fornece informações sobre os níveis de funcionalidade, disponibilidade e outros atributos além das possíveis penalidades, caso haja violações dos mesmos.

4.2 – Modelos de serviços

Na computação em nuvens a distribuição dos recursos é ofertada como serviços, desta maneira têm-se três modelos com relação aos serviços ofertados (Armbrust et al., 2009), a figura 14 mostra os principais modelos.

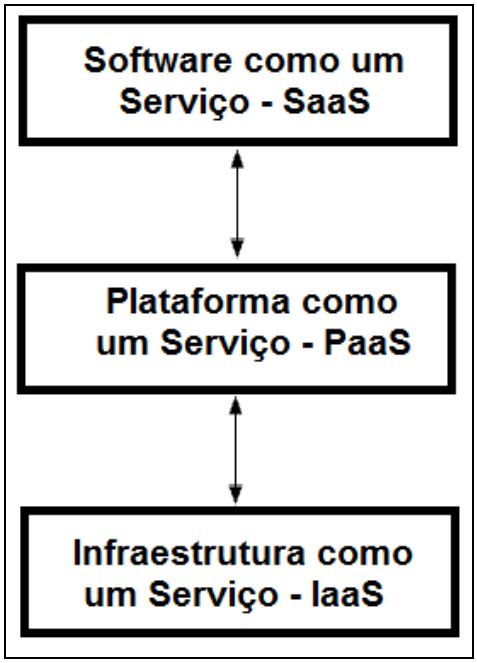

Figura 15 - Modelos de Serviços
Fonte: o Autor

Estes modelos são utilizados por públicos específicos, como por exemplo, a IaaS é utilizada geralmente pelos arquitetos de informação, no PaaS a utilização é feita pelos desenvolvedores e o usuário final usa o SaaS. Sabendo-se quem utiliza os modelos de serviço, então se procura compreender as definições a seguir:

4.2.1. Infraestrutura como serviço (*Infrastructure as a Service - IaaS).*

Infraestrutura como um serviço também pode ser chamado de *Hardware* como um serviço (*Hardware as a Service – HaaS*).

Na camada IaaS estão os serviços que ofertam os recursos computacionais que são disponibilizados pelos provedores de infraestrutura. Os recursos são muito semelhantes com uma infraestrutura física e pode ser possível controlar os softwares utilizados. Alguns recursos que se encontram disponíveis na IaaS são o armazenamento de dados, sistemas de arquivos distribuídos, distribuição de processamento e virtualização de nós computacionais.

Com o intuito de atender às regras da Computação nas nuvens é oferecida uma interface de gerenciamento das camadas superiores que torna possível a automação do processo de inicialização de nós computacionais, configurar rede de dados, definir a capacidade de armazenamento, configurar as normas de tolerância a falhas e outras.

A IaaS é usufruída pelos provedores de serviços e pelo usuários que necessitam usar os recursos computacionais.

O provedor de infraestrutura quando é acessado por usuários realiza também o papel de provedor de serviço.

Podem ser destacados como provedores de infraestrutura da camada IaaS, o *Elastic Compute Cloud (EC2)* da *Amazon* e o Simple Storage Service (S3).

4.2.2. Plataforma como serviço (Platform *as a Service - PaaS).*

Na camada plataforma como um serviço encontram-se os serviços que ofertam plataformas de programação e de execução. Nesta camada são ofertadas ferramentas e Interfaces de Programação de Aplicações (*Application Programming Interface – API*) que servem para facilitar ao desenvolvimento de aplicações além de realizar a implantação das mesmas na nuvem.

Na PaaS os desenvolvedores devem ter seu foco voltado exclusivamente às regras de negócio da aplicação, enquanto que a plataforma será a responsável pelo volume de tráfego, desempenho, escalabilidade e armazenamento de dados.

A principal vantagem da PaaS é que o desenvolvedor de acesso a infraestrutura com um mínimo custo ou até mesmo custo zero. Tendo com desvantagem as APIs que se encontram no mercado atualmente em sua grande maioria são consideradas como proprietárias, ficando muito complicado efetuar o transporte de uma aplicação de uma nuvem a outra.

4.2.3. Software como serviço (Software *as a Service - SaaS).*

As aplicações desenvolvidas especificamente para o ambiente computacional nas nuvens encontram-se na camada software como um serviço, onde são fornecidos como serviços por provedores aos usuários.

Nesta camada os serviços que são aqui utilizados podem utilizar os ambientes de programação e compilação que são fornecidos pelos provedores de PaaS ou por intermédio da IaaS, caso seja necessário utilizar diretamente uma infraestrutura.

Por ser um ambiente computacional, em que o usuário realiza o acesso remotamente às aplicações, não é necessário que haja instalado no hardware as aplicações, sendo cobrado apenas o que for utilizado.

Uma vez feita à solicitação deste serviço, o usuário não necessita preocupar-se com manutenção, operação e suporte, pois estas tarefas são de responsabilidade do provedor do serviço.

4.3 – Modelos de Implantação de Nuvem

Em um ambiente computacional nas nuvens, quando se relaciona as disponibilidades e/ou ao acesso, apresentam-se tipos de modelos de implantação que são distintos.

Uma empresa pode desejar que alguns usuários passem a ter acesso a determinados recursos, bem como, decidir quais as restrições dos mesmos ao realizarem o acesso no ambiente computacional nas nuvens, é o que se chama de restrição de acesso ou permissão.

Os modelos de implantação de serviço da computação nas nuvens independem do modelo de serviço utilizados (Software, Plataforma ou Infraestrutura como um serviço) são divididas em 04 (quatro) tipos, nuvens pública, privada, comunitária e híbrida, conforme visualizadas na figura 16.

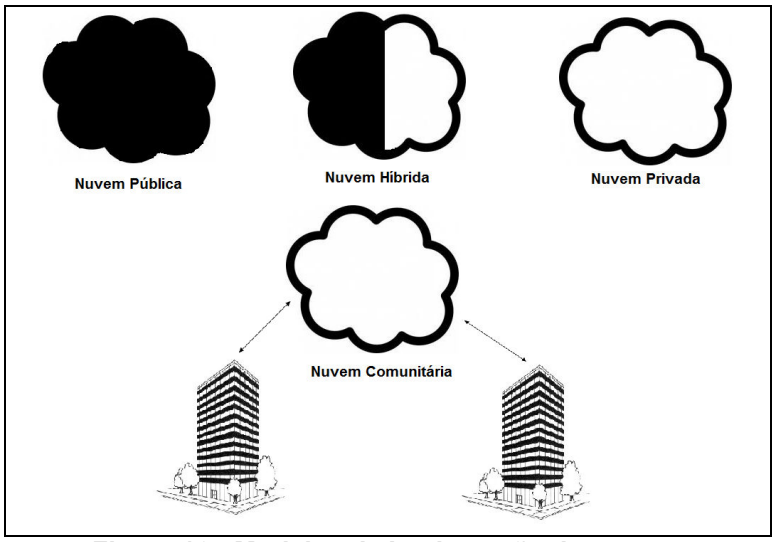

Figura 16 - Modelos de Implantação de nuvens
Fonte: o Autor

4.3.1. Nuvem Pública

Quando uma infraestrutura de uma nuvem é ofertada ao publico em geral ou para uma grande corporação, e pertence a uma organização que negocia a venda de serviços (Mell, 2011).

As nuvens consideradas como públicas são as executadas por terceiros, onde as aplicações de vários usuários encontram-se embaralhadas nos sistemas de armazenamento. Em uma nuvem pública leva-se em questão aspectos fundamentais inerentes a sua implementação, como segurança e desempenho uma vez que é composta tanto por prestadores quanto por usuários.

Neste modelo não podem ser aplicadas restrições de acesso, com relação ao gerenciamento de redes, bem como a autenticação e autorização.

A infraestrutura que é disponibilizada em uma nuvem pública é ofertada a todo público, como o nome já diz, e esta pode ser acessada por qualquer usuário desde que o mesmo conheça a localização do serviço.

4.3.2. Nuvem Privada

Quando uma infraestrutura de nuvem é usufruída exclusivamente por uma organização, chama-se de nuvem privada. Uma nuvem privada é aquela que foi construída exclusivamente para uma empresa (usuário), podendo existir no local ou ser acessada remotamente.

O usuário ao possuir toda infraestrutura da nuvem que geralmente é construída sobre um datacenter, terá o total controle sobre as aplicações implementadas para a nuvem. Neste modelo são utilizadas as políticas de acesso aos serviços, por contarem com dados e informações inerentes a empresa.

Neste modelo usufruem as tecnologias de autenticação e autorização para acessar a este modelo.

4.3.3. Nuvem Comunitária

Quando duas ou mais organizações compartilham a infraestrutura de nuvem e que partilham as mesmas preocupações (requisitos de segurança, políticas, ofício), são conhecidas como nuvem comunitária.

A nuvem comunitária é gerida por organizações ou por um terceiro, e pode ser acessada no local ou até mesmo remotamente.

4.3.4. Nuvem Híbrida

Havendo a combinação de duas ou mais infraestruturas de nuvens distintas (públicas e privadas), onde os usuários podem usufruir de uma maneira possível dentro dos padrões estabelecidos, o que o provedor de nuvem oferta com o intuito de evitar que ocorram as falhas e ter uma alta disponibilidade.

Para exemplificar, mostra-se que o usuário ao usar uma nuvem híbrida terá seus recursos de acordo com as premissas de empresa em uma nuvem privada e para fazer a tolerância contra falhas utilizar de uma infraestrutura de nuvem pública do provedor.

4.4. Federações de Nuvens Computacionais

A computação nas nuvens procura alcançar a níveis melhores de eficiência no que se refere à disponibilização dos serviços. Conforme visualizado nas seções anteriores, verificou-se que os serviços que são ofertados, a um usuário simples, a organizações empresariais consideradas grandes ou até mesmo a instituições

acadêmicas, podem variar de acordo com o desejado, seja na infraestrutura, plataformas e softwares.

Grandes organizações mantem nuvens públicas consideradas grandes, embora estejam sendo implantados atualmente milhares de nuvens menores, para organizações de todos os portes. Desta forma brotou o cenário em que relaciona nuvens computacionais de maneira a existir uma interoperabilidade entre as mais diversas instituições fazendo a intensificação no uso dos recursos.

Uma federação de nuvens computacionais, também pode ser chamada de inter-*cloud* ou *cross*-cloud (Celesti, 2010), tem a definição como sendo um conjunto de provedores de nuvens computacionais, quer sejam públicos ou privados, que se conectam por intermédio da internet.

A federação possui entre seus objetivos, de acordo com (Buyya, 2010):

- Procura eliminar a dependência de um único provedor de infraestrutura;

- Dá uma sensação de que a existência dos recursos é ilimitada e que estão disponíveis para utilização.

- Intensificar o uso dos recursos dos provedores federados.

De maneira que se consiga atingir aos objetivos, a federação admite que cada provedor de nuvem computacional possa realizar o aumento da capacidade de processamento e armazenamento, ao requisitar mais recursos às demais nuvens integrantes da federação. Os usuários têm a requisições dos recursos satisfeitas mesmo que em sua nuvem computacional esteja com o provedor fora do ar, podendo ser requisitados a outro provedor, além de solicitar a utilização de recurso que se encontram ociosos em outros provedores.

A implementação de uma federação de nuvem computacional não é realizada de modo simplório, uma vez que cada nuvem possui características específicas. Para que se possa criar uma federação de nuvens computacionais é necessário atender aos seguintes requisitos:

- Um membro de uma federação, ao ingressar em uma nuvem, usufrui mecanismos capazes de realizar a descoberta das demais nuvens que são integrantes da federação e seus recursos automaticamente.

- O sistema deve possuir alguma forma de prever as demandas dos serviços que são ofertados, de maneira que consiga fazer o escalonamento dos serviços dinamicamente entre os provedores que participam da federação.

- Os serviços ofertados pela federação devem buscar melhores níveis de eficiência, custo-benefício e utilização, ou seja, a execução deve ser realizada combinando o melhor hardware e software de maneira a garantir que o serviço seja de qualidade tenha o menor custo possível, levando-se em conta que não se possuem certeza de que o recurso encontra-se disponível.

- A federação oferece a permissão de integração de diferentes tecnologias voltadas à segurança, fazendo com que as nuvens componentes não precisem modificar as políticas de segurança ao serem inclusas na federação.

- Como podem ter uma grande quantidade de participantes, a federação deve ser capaz de trabalhar com as mais diversas requisições sendo todas mandadas ao mesmo tempo, conseguindo gerenciar sem que perca o desempenho e a escalabilidade.

4.5. Ferramentas

Em uma empresa ou instituição educacional pode ser realizada a instalação e administração de nuvens computacionais para prover os serviços e armazenamento, para tanto existem ferramentas que podem realizá-las.

4.5.1. OpenStack

O OpenStack é um software do tipo *open source* (código aberto), também conhecido como um sistema operacional da Nuvem, pois cumpre o mesmo papel de um S. O. porém em uma escala bem maior (OpenStack, 2011).

O passo inicial foi dado pelas empresas *Rackspace* que tratou de trabalhar o armazenamento (*Object Storage*) juntamente com a NASA que buscou programar o lado computacional (*Compute*).

A figura 17, mostra a arquitetura simplificada do *OpenStack*, com a sua divisão em três componentes para facilitar os principais serviços, são estes:

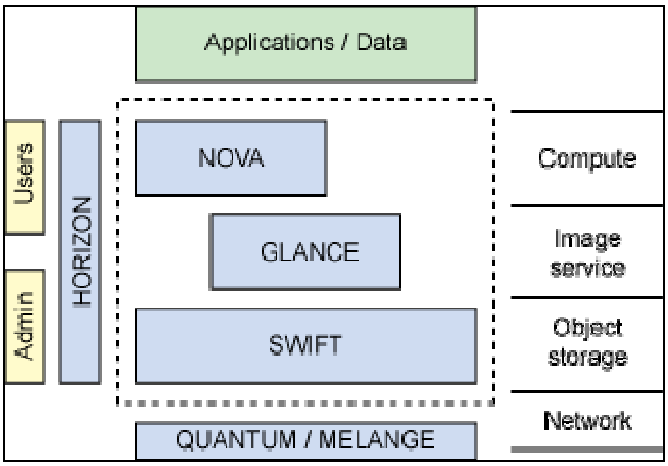

Figura 17 - Arquitetura Simplificada do OpenStack
Fonte: IBM, 2012

1- ***OpenStack Compute***, também conhecido como ***Nova***, tem sob sua responsabilidade gerenciar a infraestrutura da nuvem. Neste componente acontece o gerenciamento dos recursos computacionais da rede, da autorização e da escalabilidade da nuvem.

Foi idealizado para trabalhar utilizando a qualquer *hypervisor (*VMware, Xen, Citrix XenServer, Microsoft Hyper-V Server, VirtualBox, Workstation, Server, Player, QEMU, Microsoft: Virtual PC, Virtual Server) apesar de não apresentar capacidade de virtualização própria, tendo que utilizar APIs *libvirt* para realizar interações com os *hypervisores* suportados.

Fazem parte do *Nova* os seguintes componentes:

• API Server (nova-api)

Também é conhecido como servidor de APIs, que é responsável por fornecer uma interface que realiza a interação da infraestrutura da nuvem com o usuário, usando web services.

• Message Queue (rabbit-mq server)

Este é conhecido como “Fila de Mensagens”, onde todos os componentes do OpenStack comunicam-se através do protocolo Advanced Message Queue Protocol – AMQP (Protocolo avançado de filas de mensagens).

Usando para a requisição chamada do tipo assíncrona, que ao receber as respostas aciona o mecanismo de *call-back* (chamada de retorno).

• Compute Workers (nova-compute)

São programas de computadores executados em segundo plano, também conhecido como *daemon,* que operam o ciclo vital das instâncias das máquinas virtuais. O Message Queue transmite as requisições de gerenciamento para a execução das operações correspondentes.

• Network Controller (nova-network)

Responsável por controlar a rede, efetuando a configuração das máquinas nodos (hospedeiras). Desempenha as operações como configurar a Virtual Local Area Network – VLAN (rede local virtual), alocar endereços IP, configurar redes e programar grupos de segurança.

• Volume Worker (nova-volume)

Gerencia os volumes do tipo *Logical Volume Manager* – LVM (Gerenciador de Volume Lógico), de maneira que obtenha a criação de um volume de uma instancia, além de poder apagá-lo, conectá-lo e desconectá-lo. Este se volta ao uso das instâncias no disco principal realizando associação considerada não persistente, onde há perda nas modificações sofridas, caso ocorra à desconexão do disco ou simplesmente encerramento da instância. Embora se queira manter o conteúdo faz-se necessário que conecte o volume a uma instância. Os dados podem ser acumulados e acessados posteriormente fazendo a reconexão do volume à mesma ou a outra instância.

• Scheduler (nova-scheduler)

É um escalonador que faz o mapeamento das convocações realizadas das APIs aos componentes ideais. Executa como um *daemon* e a partir de um pool de recursos opta por um servidor seja ele compute, network ou volume de acordo com o algoritmo que foi configurado. Os fatores que influenciam o escalonador a adotar uma decisão são a memória, distância física, carga, etc.

Alguns algoritmos básicos implementados pelo Scheduler:

- Algoritmo da sorte (*chance*), que há a escolha de um *compute* dentro das zonas de disponibilidade aleatoriamente, daí a designação de sorte.

- Zona de disponibilidade (*availability zone*) funciona igual ao chance, porém a diferença se dá porque a escolha é realizada dentro de uma determinada zona de disponibilidade.

- Simples (*simple)* realiza a escolha do host que ira executar a instância, desde que tenha a carga menor, que um balanceador de carga (load balancer) emita esta informação.

2- ***OpenStack Object***, ou chamado de *Swift*, responsável pela criação e armazenamento de informações na razão de peta-bytes, é quem mantém e autoriza a acessar a qualquer arquivo ou dado.

3- OpenStack Image Service, Glance realiza a administração bibliotecária das imagens dos servidores virtuais além de catalogá-las.

4.5.2 – *Eucalyptus*

Tendo como acróstico o *Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*, Eucalyptus é um software *General Public License* – GPL (Licença Pública Geral) usado facilmente para criar e realizar a manutenção de nuvens privadas e públicas no modelo Infraestrutura como um Serviço.

Elaborado para realização de pesquisas que precisavam de uma computação de alto nível de desempenho, e baseia-se no Linux podendo ser instalada em qualquer distribuição.

A nuvem da *Eucalyptus* realiza o instanciamento de máquinas virtuais (*Virtual Machine – VM)*, armazena imagens e dados das máquinas virtuais, além de construir redes virtuais, possui interfaces administrativas e voltadas para o usuário da nuvem. Uma vez que o usuário passa a possuir permissão para iniciar, controlar, acessar e encerrar às maquinas virtuais através de uma API da *Amazon* que é a mesma do Amazon EC2 (*Amazon Elastic Compute Cloud)*. Apesar do sistema utilizar uma plataforma, seus desenvolvedores buscam aceitar a outras plataformas (Nurmi, 2009).

A figura 18 mostra a arquitetura da Nuvem, para um único cluster, apresentando a seguir a descrição de cada.

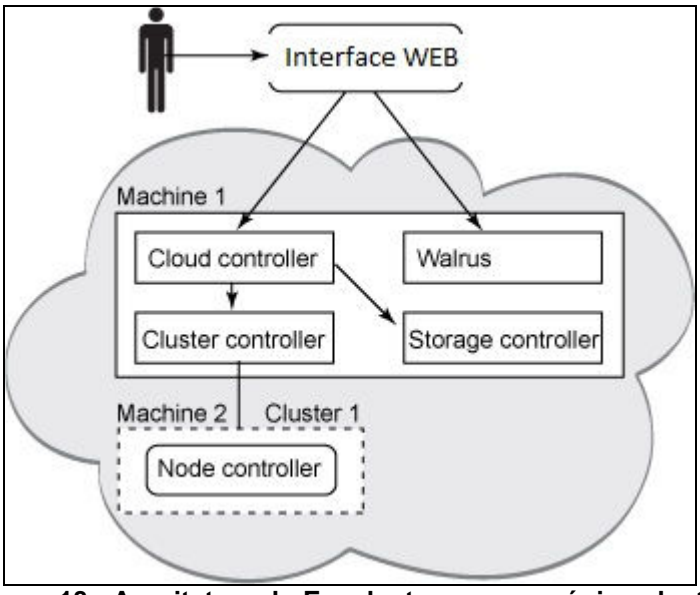

Figura 18 - Arquitetura do Eucalyptus para um único cluster
Fonte: IBM, 2010

1-Controlador de Nó (*Node Controler*)

Executado em todo nó, onde é encarregado para controlar bem como, realizar a execução das máquinas virtuais alocadas em uma máquina física. Em cima da plataforma de virtualização são realizadas comunicações entre o controlador de Nó e o *hypervisor* das máquinas físicas realizando ações (iniciar, parar e captar informações das VM). O *hypervisor* é um elemento integrante da virtualização onde realiza o gerenciamento do hardware da máquina física para compartilhar as muitas VMs.

2- Controlador de Cluster (*Cluster Controller*)

É responsável por gerenciar um ou mais controladores de nós, que pode ser executado em qualquer máquina que haja uma conexão de rede entre o Controlador de nós e o controlador de *Cluster*. É o responsável por captar informações das máquinas virtuais a fim de realizar o escalonamento das requisições entre os diversos Controladores de nós. Com o intuito de visualização, a utilização do controlador de nuvem solicita que sejam instanciadas máquinas virtuais, desta forma o controlador de *cluster* deverá verificar os recursos que estão disponíveis e decidir em qual máquina física devera instanciar as máquinas virtuais.

3- Controlador de Armazenamento *Walrus (Walrus Storage Controller)*

É responsável pelo armazenamento, e usufrui a tecnologias de Serviços voltados para internet (Web Services), apresenta uma interface para os usuários compatível com o Serviço Simples de Armazenamento da Amazon (Amazon Simple Storage Service – Amazon S3). O usuário pode depositar e retirar dados do *Walrus*, que também é responsável por armazenar as imagens das máquinas virtuais. Utiliza-se Protocolo Simples de Acesso a Objetos (*Simple Object Access Protocol – SOAP)*,por se tratar de um serviço para web.

4- Controlador de Nuvem (Cloud Controller)

Na nuvem Eucalyptus, os recursos virtualizados são expostos e gerenciados pelo controlador da nuvem, e é considerada como porta de entrada do sistema tanto para os usuários quanto para os administradores. Tem como funções realizar o monitoramento das instâncias em execução, além da disponibilidade dos recursos e decidir quais *clusters* serão usados para provisionar tais instancias.

O controlador de nuvem é formado por serviços voltados para web que estão amontoados nas seguintes funções:

a- Serviços de Recursos

Os usuários realizam solicitações para que suas requisições sejam atendidas como a manipulação das propriedades das máquinas virtuais e ainda o recebimento das informações sobre o estado das mesmas. Realiza a comunicação com os controladores de Cluster fazendo a alocação, desalocação ou disponibilizando informações dos recursos destinados às máquinas virtuais. As requisições serão atendidas dependendo das SLA firmadas além da disponibilidade dos recursos.

b- Serviços de Dados

Serve para controlar o armazenamento de dados dos usuários e de estado do sistema. Podem ser utilizados a fim de que possam obter as informações dos recursos do sistema, onde utilizam as informações dos serviços de dados para realizar a verificação dos parâmetros das requisições realizadas pelos usuários.

c- Serviço de Interfaces

Para que usuários e administradores possam obter acesso ao sistema e realizarem as alterações desejadas de suas propriedades, são ofertados interfaces, que são implementadas através de SOAP, ou ainda por uma interface web. Objetiva efetuar a tradução de comandos dados pelos usuários ao sistema.

4.6. Conclusão

Tendo sido apresentado como um novo modelo computacional, a computação nas nuvens vem ultrapassando fronteiras tanto nas instituições acadêmicas, quanto em empresas de cunho comercial. Neste novo modelo, o acesso a informações dos usuários, bem como as aplicações e serviços que o usuário deseja usufruir encontram-se armazenados em locais longínquos e que podem ser acessados através da Internet.

A computação nas nuvens não apresenta em seus conceitos novidades, o que trata como novo é realizar a união de todos esses conceitos em um sistema considerado maior.

Ao apresentar características essenciais ao seu funcionamento, devem-se levar em conta que todas são complementares umas a outras e extremamente necessárias.

Viu-se também os modelos de implantação existentes, assim como os modelos de serviços inerentes a computação nas nuvens. E ainda algumas ferramentas que são utilizadas para implantar um sistema de nuvem próprio em algum empreendimento e até no meio acadêmico.

5. ABORDAGEM PROPOSTA

Em um ambiente computacional nas nuvens, verifica-se que o usuário realiza o acesso remotamente e, que para a segurança dos recursos seja garantida, os serviços deve efetuar a identificação dos usuários de acordo com as informações fornecidas pelos mesmos para que sejam autorizados a usufruírem aos serviços desejados. Desta maneira quando o usuário for acessar a um recurso, este deverá confirmar suas informações de identidades.

Com a finalidade de aumentar a segurança na computação nas nuvens, de maneira que sejam minimizados os problemas inerentes à privacidade e evitar que sejam roubadas as identidades é que se trabalha o gerenciamento de identidades.

Muitas pessoas e empresas ainda não adotaram a computação nas nuvens por apresentarem preocupações acerca da segurança, em face de que ao depositarem no sistema os dados e informações que poderão ser utilizados posteriormente, podem sofrer violações dos seus direitos.

Neste trabalho são propostas duas soluções voltadas à autenticação em um ambiente computacional nas nuvens, são elas: uma abordagem de autenticação centralizada e uma abordagem de autenticação distribuída.

Para que os objetivos fossem atingidos, o gerenciamento de identidades pode ser realizado entre diferentes plataformas de computação nas nuvens, de maneira que possam manter suas características internas e também suas políticas.

Para modelagem do sistema utilizou-se a linguagem unificada para modelagem de sistemas (*Unified Modeling Language - UML*). Dentro da linguagem de modelagem apresentam-se diversos diagramas que servem para expressar os requisitos funcionais e até mesmo o comportamento do sistema, além de contar com a ajuda do software *StarUML* para confecção dos diagramas (StarUML, 2005).

5.1.Proposta de abordagem de autenticação centralizada

A proposta da abordagem que leva em consideração que a autenticação deve ocorrer de maneira centralizada, é mostrada na figura 19.

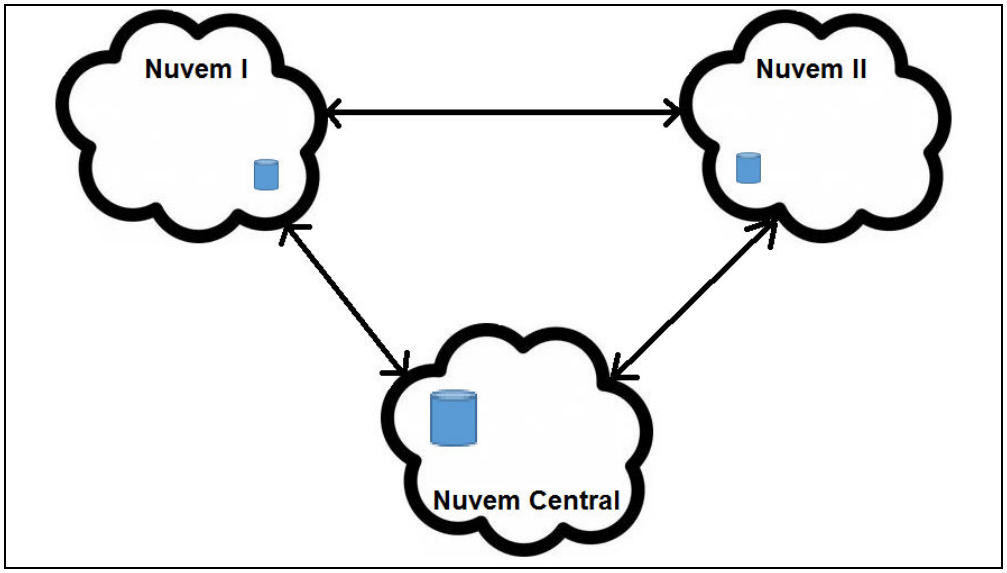

Figura 19- Proposta para Modelo de Autenticação Centralizada
Fonte: o Autor

A figura visualizada acima apresenta uma esquematização da maneira como as nuvens realizam o gerenciamento de identidades, partilhando a autenticação.

A autenticação realizada por um usuário em determinada nuvem, deve ser analisada em outra nuvem que é considerada central, em virtude desta ser possuidora das credencias dos mesmos, conforme demonstrado na figura 20, do caso de uso em questão.

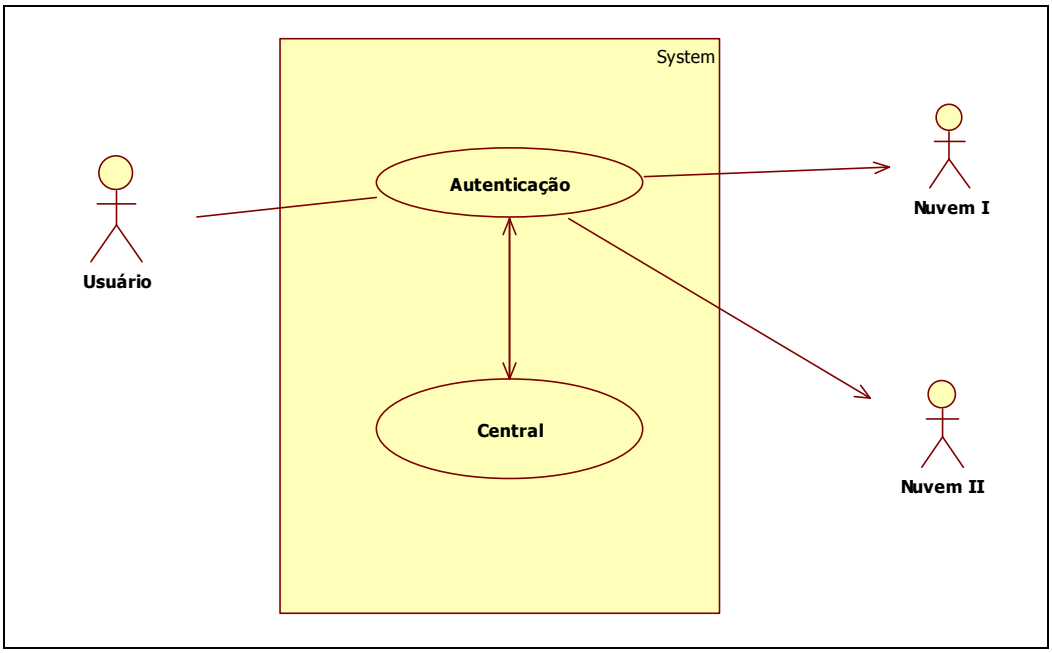

Figura 20 - Caso de uso da proposta da abordagem centralizada
Fonte: o Autor

A UML é utilizada com a finalidade de ilustrar o funcionamento dos processos dentro de um sistema computacional, dando ênfase a sequência lógica em que as mensagens são trocadas, é o que se chama de diagrama de sequência (OMG, 2012).

A proposta de autenticação centralizada é realizada conforme diagrama de sequência, apresentado na figura 21.

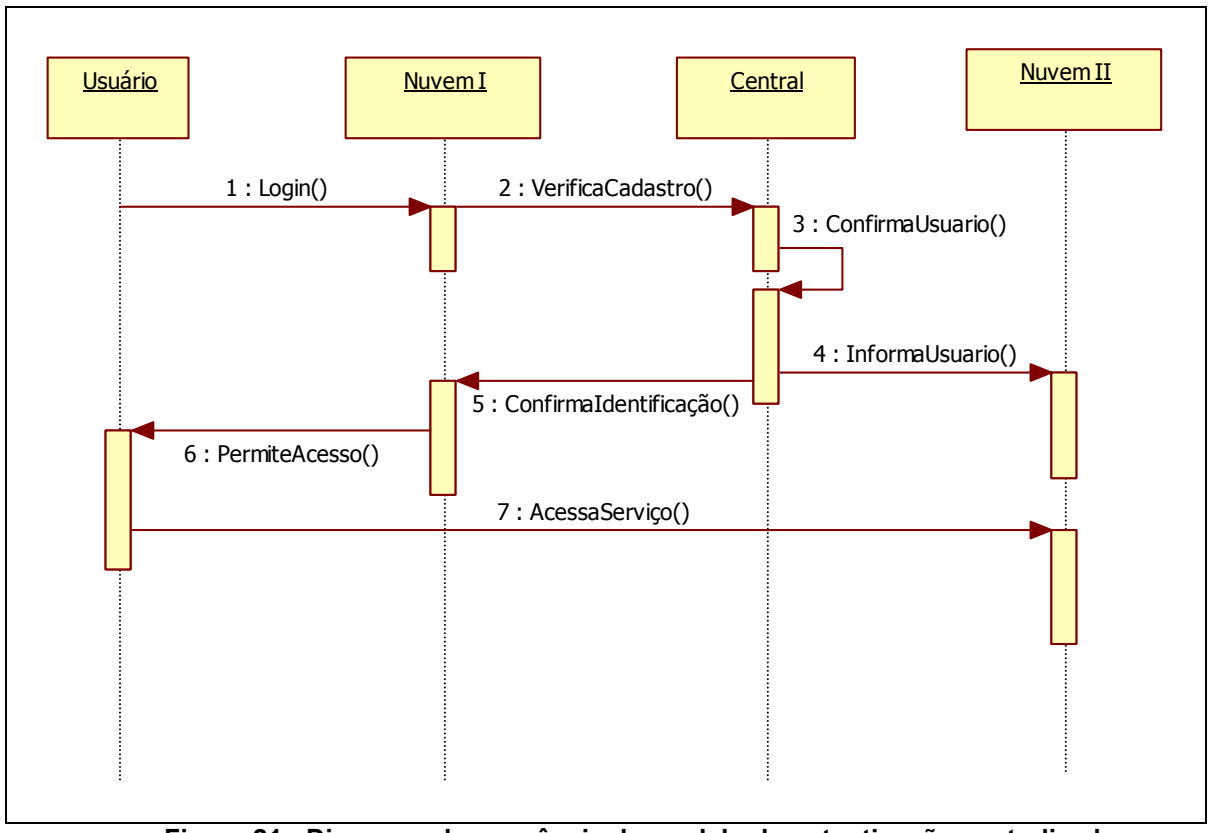

Figura 21 - Diagrama de sequência de modelo de autenticação centralizada.
Fonte: o Autor

A visualização da figura 21 demonstra como vem a funcionar a proposta da autenticação centralizada de computação em nuvem:

1- Usuário efetua *login* na Nuvem I;
2- A Nuvem I envia os dados de *login* do usuário para a nuvem central que é responsável por prover os dados de todos os usuários;
3- A Nuvem Central ao receber os dados de *login* do usuário, realiza a verificação em sua lista de cadastro de usuários e atributos;

4- A Nuvem Central informa a Nuvem II que o usuário tem acesso a alguns recursos da nuvem e que estará diretamente *logado* na Nuvem I;
5- A Nuvem Central confirma a identificação do usuário;
6- A Nuvem I permite que o usuário acesse aos serviços;
7- O usuário acessa a alguns serviços existentes na Nuvem II.

Ao ser realizada a autenticação do usuário a sua informação é repassada em forma de mensagem de *broadcast*[3], o que tornará possível que o usuário autenticado em um provedor possa utilizar a outro.

5.2. Proposta de abordagem de autenticação distribuída

Outra sugestão que é proposta como uma solução é a abordagem de autenticação realizada distributivamente.

A verificação da abordagem se dá ao se realizar a autenticação em um provedor de nuvem e este, por sua vez, envia a outros provedores a autenticação realizada pelo usuário, o exemplo pode ser mais bem compreendido ao visualizar a figura 22.

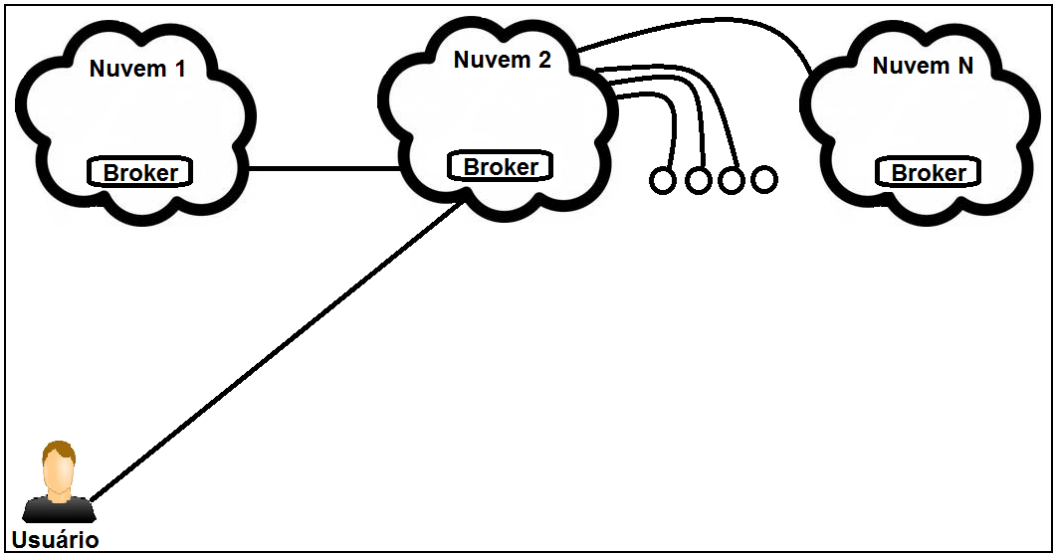

Figura 22 - Proposta de modelo de autenticação distribuída
Fonte: o Autor

[3] *Mensagem de Broadcast. São mensagens que são enviadas para cada remetente existente em uma rede.* (Networking - Define Broadcast, Unicast and Multicast - July 31, 2009 at 17:00 pm by Vidya Sagar).

A figura 22 apresenta um esquema da maneira que deve ser visualizada a proposta em que a autenticação realizada em provedor de nuvem dar-se-á de maneira distribuída, e que participam de um círculo de confiança onde as diversas “N” nuvens permitem-se compartilhar da autenticação realizada.

Após formalização do esquema, foi elaborado um diagrama de caso de uso, de maneira que fossem visualizados os principais atores, conforme demonstrado na figura 23.

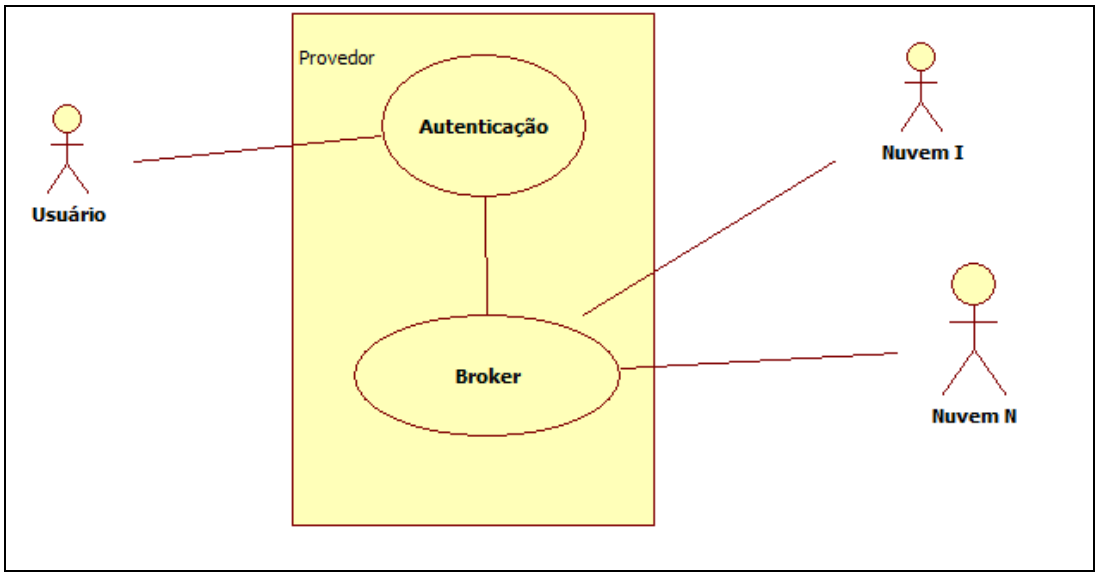

Figura 23 - Caso de uso da proposta da abordagem distribuída.
Fonte: o Autor

Tendo visualizado o caso de uso, elaborou-se o diagrama de sequência pertinente ao sistema, para que fosse auxiliado na prototipação do mesmo, conforme na figura 24.

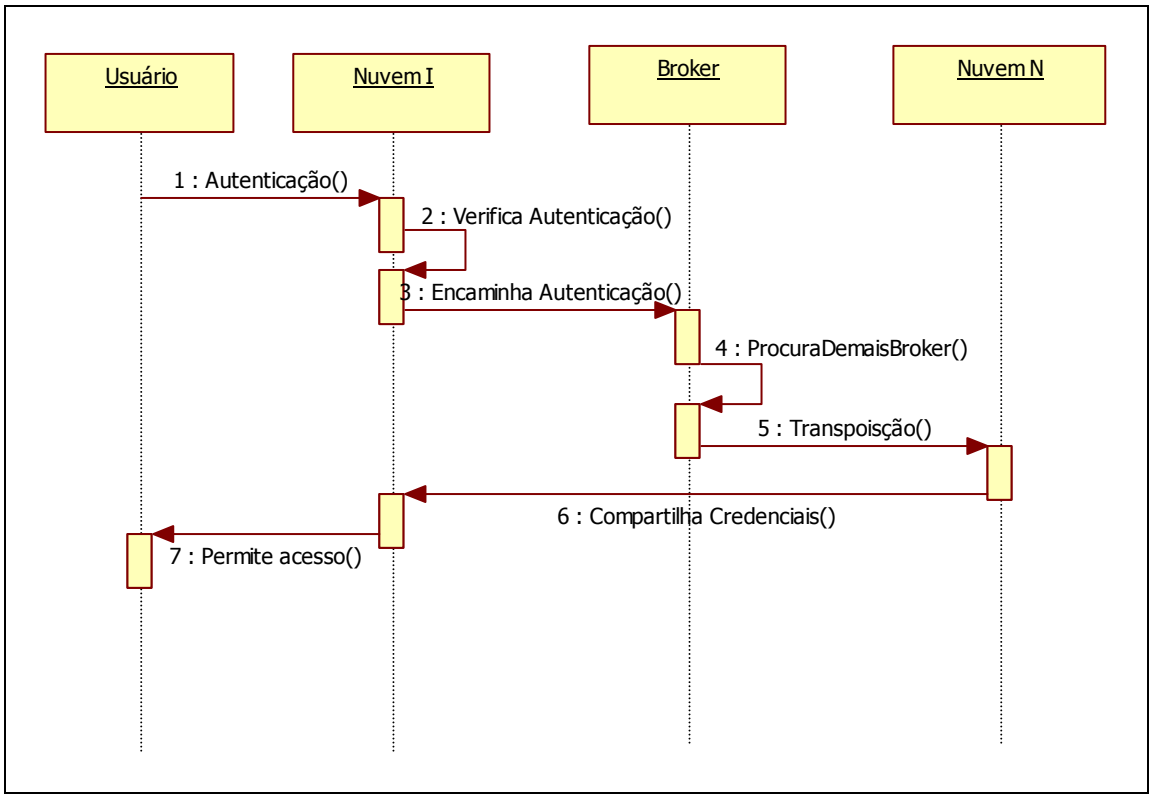

Figura 24- Diagrama de sequência da solução da abordagem distribuída.
Fonte: o Autor

O diagrama de sequência demonstra o funcionamento do sistema proposto de maneira que abaixo elucidará a troca de mensagens.

1- O usuário realiza a autenticação para acessar a nuvem I;
2- A nuvem I realiza a verificação da autenticação na nuvem I;
3- Ao ter a autenticação confirmada, esta é encaminha ao *broker* da nuvem I;
4- Posteriormente este *broker* procura os demais *brokers* das diversas Nuvens;
5- Efetua a "transposição" de credenciais para as N nuvens que existem no circulo de confiança;
6- Compartilha com as credenciais do usuário;
7- Permite o acesso do usuário nas diversas nuvens.

Ao se dispor desta proposta de autenticação foi levada em conta que podem ser disponibilizadas diversas nuvens (nuvem I, nuvem II, ..., Nuvem N) homogêneas ou não.

Para tanto é necessário que esta comunicação seja efetuada de maneira segura, onde a confiança da identidade das entidades envolvidas seja garantida. Este controle de acesso deve permitir que apenas os agentes que são devidamente autorizados possam executar operações e acessar aos recursos.

Para efetivação de uma comunicação segura, foi vislumbrada a utilização de uma **Especificação XML para Gerenciamento de Chaves** (*XML Key Management Specification* – XKMS), pois o sistema apresenta a comunicação realizada de maneira não segura, utilizando a criptografia.

No XKMS os protocolos servem para que sejam geradas pares de chaves, realizar o armazenamento, localizar e validar as informações das chaves públicas, além de realizar a validação de assinaturas (Morais, 2009).

Este padrão apresentado divide-se em duas partes: a primeira voltada para efetuar a localização das informações que se encontram associadas às chaves é o X-Kiss (*XML Key Information Service Specification*) a outra é responsável por realizar o registro das informações é o XKRSS (*XML Key Registration Service Specification*).

Assim sendo, foi necessário que todas as entidades que compõem o sistema necessitam que sejam realizados os registros de chaves públicas além de um servidor de XKMS que funciona como um repositório para as chaves.

Para que se possa aumentar a medida de segurança do sistema proposto, foi usada uma chave secreta, que é temporária e compartilhada usada com a finalidade de criptografar e descriptografar as mensagens de registro entre as entidades e o servidor de XKMS. A chave gerada é única, e cada entidade recebe uma que é diferente das demais.

Uma vez que qualquer entidade aceite os termos impostos pelo servidor XKMS, inicia-se o processo de registro para que a partir deste ponto possa ser gerada uma chave secreta para ambos através do algoritmo de *Diffie-Helman* (Ramos, 2002). Em seguida as entidades ao iniciar as atividades pode gerar separadamente seu par de chaves pública e privada. A chave pública que é gerada é criptografada utilizando a chave secreta enviando ao servidor XKMS uma mensagem cifrada. O servidor XKMS ao estar de posse da mensagem, se encarrega de utilizar a chave secreta para descriptografá-la e efetuar o armazenamento do conteúdo, que é a chave pública da entidade, na base de dados em uma Infraestrutura de Chaves Públicas (conhecido como *Public* Key *Infrastructure - PKI*).

Logo após o servidor XKMS expede uma resposta à entidade portando a sua chave pública criptografada com a chave secreta que é a mesma do início, de posse da mesma o usuário realiza o mesmo processo que o servidor XKMS efetuou para conseguir decifrá-la. Ao terminar todo este processo a chave secreta é descartada e o registro finalizado, conforme visualizado na figura 25.

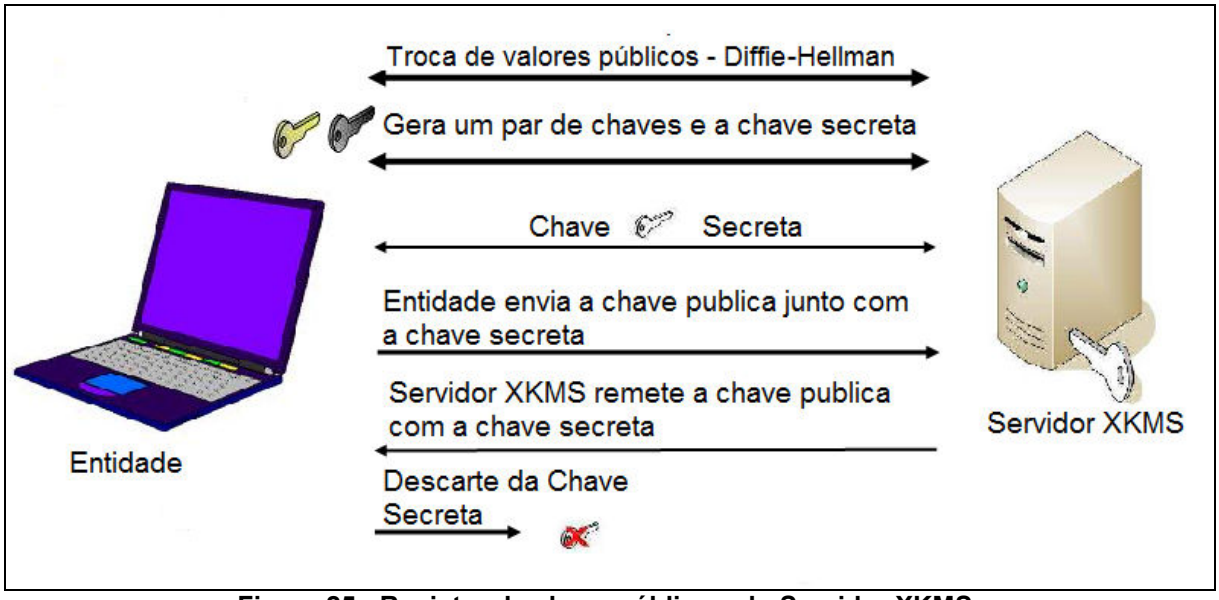

Figura 25 - Registro da chave pública pelo Servidor XKMS.
Fonte: Adaptado de Moraes, 2009.

Para que o sistema apresente uma proteção a mais no quesito segurança, propõe-se também neste trabalho, a utilização do processo de criptografia com a finalidade de proteger as informações de configuração, que apresentam dados inerentes tanto ao servidor XKMS quanto ao parâmetro que será utilizado pelo algoritmo para troca de chaves. Por tanto estas informações são armazenadas em arquivo, de maneira a evitar que usuários mal intencionados passem a interferir na referida comunicação e consigam enganar as entidades do sistema.

Os arquivos não efetuam a troca dos parâmetros que neles encontram-se contidos por intermédio da rede e as entidades usufruem o endereço MAC do servidor XKMS de maneira estática o que causa uma dificuldade para que possam sofrer possíveis ataques.

Como o arquivo de configuração é apresentado em XML o mesmo apresenta internamente dados que podem ser interceptados como endereço IP e MAC, além do algoritmo de troca de chaves.

A solução imposta leva em consideração um esquema para realização da proteção que se baseia em uma geração dinâmica das informações de configuração e que seja armazenado um arquivo criptografado. Toda vez que o servidor XKMS tiver sua execução inicializada o procedimento será realizado, da seguinte maneira:

O servidor XKMS cria um par de chaves, que contém uma chave privada e outra pública que recebem a denominação de “chaves de configuração”. Enquanto o sistema é executado, o arquivo é gerado e realizado a criptografia que usufrui a chave privada de configuração, para posteriormente obter o compartilhamento com as entidades através de um documento XML criptografado.

As entidades do sistema podem executar a leitura do arquivo, somente de posse com a chave pública de configuração que foi fornecida pelo servidor XKMS. A chave pública somente é de posse das entidades que forem autenticadas e autorizadas pelo servidor e somente as pertencentes ao sistema.

Para realização de uma comunicação segura, é imprescindível que somente as entidades que estão envolvidas no sistema tenham acesso às funções do servidor XKMS.

O servidor XKMS ao utilizar o mecanismo de autenticação, realiza a restrição do acesso aos serviços de registro, além da localização das chaves públicas somente aos que integram ao sistema. Já o mecanismo de autorização juntamente com o servidor de XKMS será o responsável por controlar aos recursos que podem ser acessados pelas entidades que estão devidamente autenticados.

O modelo de comunicação que é efetuado entre as entidades e o servidor XKMS deve ser realizado de maneira que as identidades sejam preservadas e estabelecidas de maneira confiável. Conforme mostrado na figura 26, podem ser realizados em etapas que servirão como barreiras para que o sistema não possa ser utilizado por quem não possuir a autorização devida.

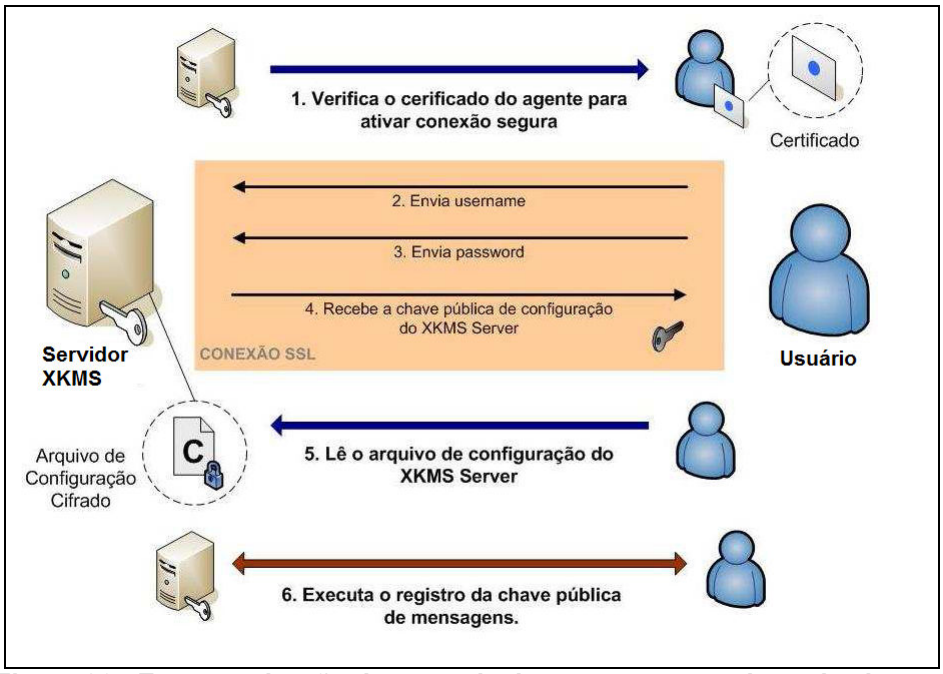

Figura 26 - Esquematização do controle de acesso com registro de chaves. Fonte. Moraes, 2009.

A primeira barreira é tratada pelo servidor de XKMS que é também responsável por estabelecer um canal de comunicação considerado seguro através da camada de proteção de socket (*Secure Socket Layer – SSL),* é o local em que percorre o trafego das informações de *login*, o envio da chave pública de acesso, sendo necessário principalmente para locais onde acontecem transações que necessitam manter os dados confiáveis (Apache, 2013).

As camadas em que ocorre a atuação do SSL é entre a camada de transporte TCP *(Transport Control Protocol – TCP)* e a camada de aplicação, esta pode ser executada nos mais diversos protocolos TelNet, File Transfer Protocol – FTP, o mais usual de todos o *HyperText Transfer Protocol* – HTTP, além de outros.

Dentro da camada SSL, o usuário informa a sua identificação através do seu *login* (*username* e *password*), ficando passível a aplicação de medidas consideradas básicas e que são voltadas a cada usuário, onde o servidor efetua as restrições aos recursos que previamente foram definidas, este é outra barreira.

O servidor XKMS faz a análise do *login* do usuário e ao confirmar a sua autenticidade, disponibiliza a chave pública para que possam ler as informações codificadas em arquivo de configuração do servidor XKMS, com a finalidade de que seja gerada a chave secreta.

Ao ter gerada a chave secreta, esta é compartilhada entre o usuário e o servidor XKMS, para posteriormente iniciar o registro de chave pública.

Para aumentar a segurança, propôs aumentar a criptografia do sistema, necessitando fazer a sua implantação de maneira que entidades alheias ao sistema tivesse acesso indevido às chaves que são utilizadas para este fim.

Após ter realizado a autenticação e criptografada a mensagem, por intermédio das chaves, um broker realiza uma procura a outros brokers existentes em cada nuvem e verifica que a nuvem é integrante do círculo de confiança, por intermédio de uma certificação digital. Caso esta nuvem tenha verificada a autenticidade do agente usuário este terá sua autenticação disseminada as outras nuvens por *broadcast* e estas por sua vez permitirão o acesso a alguns recursos destas.

6. Validação da proposta

Para solucionar o problema estudado, foram propostas duas soluções fazer uma abordagem de que envolvesse o gerenciamento de identidade, através de uma abordagem de autenticação centralizada e outra abordagem de autenticação distribuída. Para que fossem solucionadas as problemáticas abordadas foi elaborado o protótipo voltado para a solução da autenticação realizada de maneira distribuída.

6.1. Cenário utilizado

Para implementação do ambiente de teste foram montadas duas estruturas computacionais de nuvens privadas com a finalidade de que se obtivesse a base adequada para a realização da proposta.

No ambiente cada nuvem era composta com máquinas que possuíam processadores core i5, uma memória de 4 GB, HD de 500GB. Internamente continha como sistema operacional Linux Centos V.6 que é uma distribuição gratuita do Linux. Como ambiente de nuvem utilizou-se o *Eucalyptus* V.3.3.1.. A figura 27 mostra o cenário da abordagem.

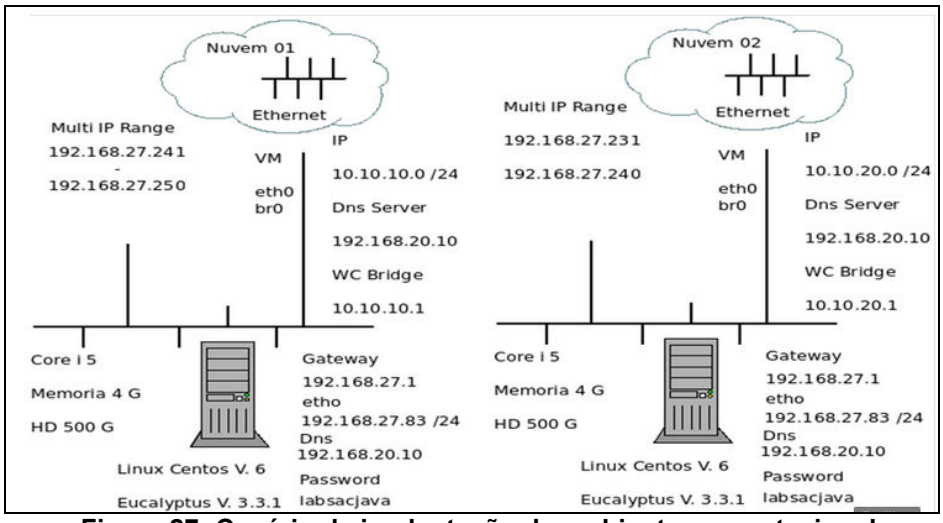

Figura 27- Cenário da implantação do ambiente computacional
Fonte: o Autor

O cenário serviu como modelagem para realização do sistema com as nuvens integrantes desta proposição. Para a implementação foi escolhida a linguagem de programação *Java*.

6.2. Implementação da abordagem

O mecanismo utilizado para realizar a proteção da informação de configuração para fornecer a segurança no acesso às informações sensíveis do servidor XKMS.

Para a prototipação adotou-se o mecanismo de segurança onde foram criados métodos para que as informações (IP, MAC e valores do *Diffie-Hellman*) deixem de ser armazenadas em um documento estático e legível para que se possa ser gerado, criptografado e armazenado de maneira dinâmica.

Para melhor demonstração foi elaborado um diagrama de sequência, mostrado na figura 28, de maneira a compreender o funcionamento da implementação realizada.

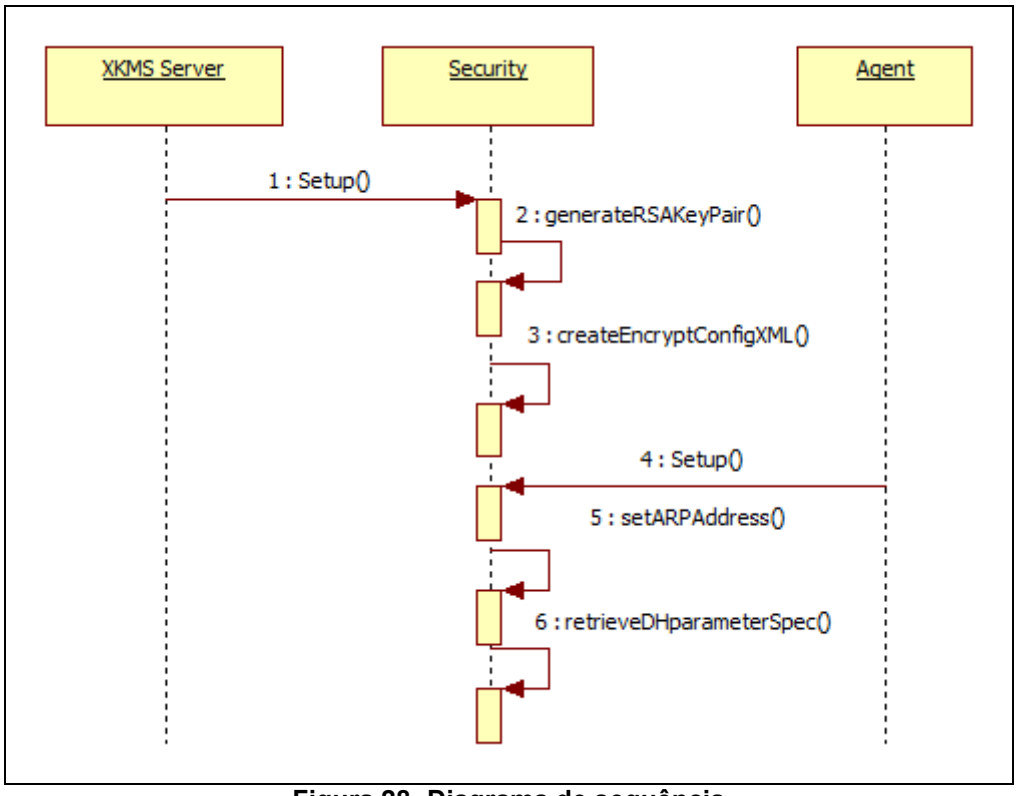

Figura 28- Diagrama de sequência
Fonte: o Autor

O diagrama de sequência demonstrado acima seguirá a seguinte ordem:

1°. O *XKMS server*, a fim de encriptografar as informações de configuração, ativa o método *setup* da classe *Security*;

2°. A classe *Security*, apresenta o método *generateRSAKeyPair*, que é responsável por gerar um par de chaves (pública e privada);

3°. Logo após o método *createEncryptConfigXML* se encarrega de criar um arquivo XML de configuração encriptografado, que contém também os valores correspondentes às informações que serão compartilhadas, depois o documento é submetido a cifragem através do par de chaves geradas.

4°. A classe *Agent* busca obter o acesso com a classe Security, com a finalidade de que somente as entidades integrantes ao sistema possam realizar acesso em segurança, iniciando o método *setup*;

5°. Os agentes são responsáveis por realizar a leitura das informações de configuração, onde o método *setARPAddress* decodifica as informações de endereços do servidor (IP e MAC) e as disponibiliza para que seja executado o comando para execução do Protocolo de Resolução de Endereços *(Address Resolution Protocol - ARP)* para que se possa dispor da entrada estática no Sistema Operacional através do endereço MAC do servidor XKMS.

6°. E tem-se também o método *retrieveDHparameterSpec* que decodifica e dispõe de volta os parâmetros do algoritmo que é responsável por gerar as chaves secretas de acordo com o *Diffie-Hellman*.

A seguir será mostrado um fragmento do código da classe Security, utilizado para a prototipagem.

```
1    public class Security {
2       [...]
3       public void setup() {
4          [...]
5          confRsaKeyPair = generateRSAKeyPair();
6          [...]
7          createEncryptConfigXML();
8           [...]
9          }
10      private void createEncryptConfigXML() {
11          [...]
12          conf = doc.createElement("conf");
13          xkms = doc.createElement("xkms_server");
14          ip = doc.createElement("ip_address");
15          mac = doc.createElement("mac_address");
16          dh = doc.createElement("diffie-hellman");
17          [...]
18          doc.appendChild(conf);
19          conf.appendChild(xkms);
20          xkms.appendChild(ip);
21          ip.appendChild(doc.createTextNode(InetAddress.getLocalHost()
                        .getHostAddress()));
22          xkms.appendChild(mac);
23          mac.appendChild(doc.createTextNode(NetworkInfo.getMacAddress()));
24          conf.appendChild(dh);
25          dh.appendChild(p);
26          p.appendChild(doc.createTextNode("17914[...]"));
27          dh.appendChild(g);
28          g.appendChild(doc.createTextNode("10080[...]"));
29          dh.appendChild(l);
30          l.appendChild(doc.createTextNode("1023"));
```

```
31          [...]
32          Encryptor e = new Encryptor(doc, generateDESKey(),
                        AlgorithmType.TRIPLEDES, confRsaKeyPair.getPrivate(),
                        AlgorithmType.RSA1_5);
33          Document encrypt_doc = e.encrypt();
34          DOMSource source = new DOMSource(encrypt_doc);
35          StreamResult result = new StreamResult("\\\\"
                    + InetAddress.getLocalHost().getHostAddress() +"\\mtp\\conf.xml");
36          transformer.transform(source, result);
37        }
38  [...]
```

Como pode ser visualizado no trecho do código acima, nas linhas 5 e 7 ocorre a geração das chaves de configuração, bem como é evocado ao método que é responsável por gerar o arquivo de configuração cifrado, *createEncryptConfigXML*.

Da linha 12 a 16 ocorrerá a criação do documento XML, que conterá as *tags* possuidoras da identificação das informações que deverão ser armazenadas. Imediatamente das linhas 18 a 30 tem-se que cada informação é vinculada às *tags* correspondentes, bem como na linha 32 e 33 o documento recebe a criptografia devida utilizando a chave de configuração privada. Por fim, o documento que foi codificado é salvo e armazenado em um arquivo chamado de "*conf.xml*", como visualizado nas linhas de 34 a 36.

Conta-se com o arquivo que é responsável por realizar a recuperação das informações que se encontram inseridas no arquivo de configuração, conforme demonstrado a seguir, onde há a invocação os métodos *setARPAddress* na linha 3, bem como o *retrieveDHparameterSpec* na linha 17.

```
1    public class Security {
2       [...]
3    private void setARPAddress() {
4       [...]
5       publicKeyXkms = locateConfigKey();
6       DOMParser parser = new DOMParser();
7       parser.parse("\\\\"+ InetAddress.getLocalHost().getHostAddress()
                    +"\\mtp\\conf.xml");
8       Document doc = parser.getDocument();
9       [...]
10      Decryptor dec = new Decryptor(doc, publicKeyXkms, xpath);
11      Document decrypt_doc = dec.decrypt();
12      ip = decrypt_doc.getElementsByTagName("ip_address").item(0)
              .getFirstChild().getNodeValue();
14      mac = decrypt_doc.getElementsByTagName("mac_address").item(0)
              .getFirstChild().getNodeValue();
15      [...]
16     }
17    private DHParameterSpec retrieveDHparameterSpec() {
18      PublicKey publicKeyXkms;
19      if (agent.getAID().equals(xkmsServer))
20      publicKeyXkms = confRsaKeyPair.getPublic();
21      else
22      publicKeyXkms = locateConfigKey();
23      DOMParser parser = new DOMParser();
24      parser.parse("\\\\"+ InetAddress.getLocalHost().getHostAddress()
                    +"\\mtp\\conf.xml");
25      Document doc = parser.getDocument();
```

```
26      [...]
27      Decryptor dec = new Decryptor(doc, publicKeyXkms, xpath);
28      Document decrypt_doc = dec.decrypt();
29      String parameterP = decrypt_doc.getElementsByTagName("prime_modulus_P")
                .item(0).getFirstChild().getNodeValue();
30      String parameterG = decrypt_doc.getElementsByTagName("base_generator_G")
                .item(0).getFirstChild().getNodeValue();
31      String parameterL = decrypt_doc.getElementsByTagName("bit_size_exponent_L")
                .item(0).getFirstChild().getNodeValue();
32      return new DHParameterSpec(new BigInteger(parameterP),
                new BigInteger(parameterG),
                Integer.parseInt(parameterL));
34     }
35  [...]
36 }
```

No trecho acima, considera-se que há o recebimento da chave de configuração pública. Esta é imprescindível para que haja a decodificação das informações, como demonstrado nas linhas 5, 20 e 22. De posse da chave pública, o arquivo de configuração e o conteúdo existente nele é capturado no ambiente do servidor e submetido ao processo de decifragem, como visto nas linhas 7 e 8, e nas linhas 24 e 25. Para execução do comando é necessário que se tenha o endereço IP e MAC do Servidor XKMS, portanto nas linhas 12 e 14, há a visualização destas que serão usadas para que seja executado o comando ARP, para a parte estática do documento. Para que a chave secreta seja gerada, é preciso que haja definição do parâmetro utilizado pelo algoritmo do *Diffie-Hellman* que serve para a decodificação, como visto entre as linhas de 29 a 31.

A prototipação segue agora para o mecanismo de autenticação e autorização para acessar aos recursos do XKMS Server. Para que o XKMS Server possa adimplir a operação de autenticação da entidade, é necessário que primeiramente seja criada a conexão segura, que é acionado o método '*run'* da classe *XKMSConnectSSL*, a qual cria uma configuração segura de um *socket* para o servidor por intermédio do *createSSLServerSocket*. Após este espera que a entidade realize a conexão e que, realize o *login* através da classe *ServerSocket* pelo método *accept,* conforme pode ser visto no diagrama de sequência na figura 29.

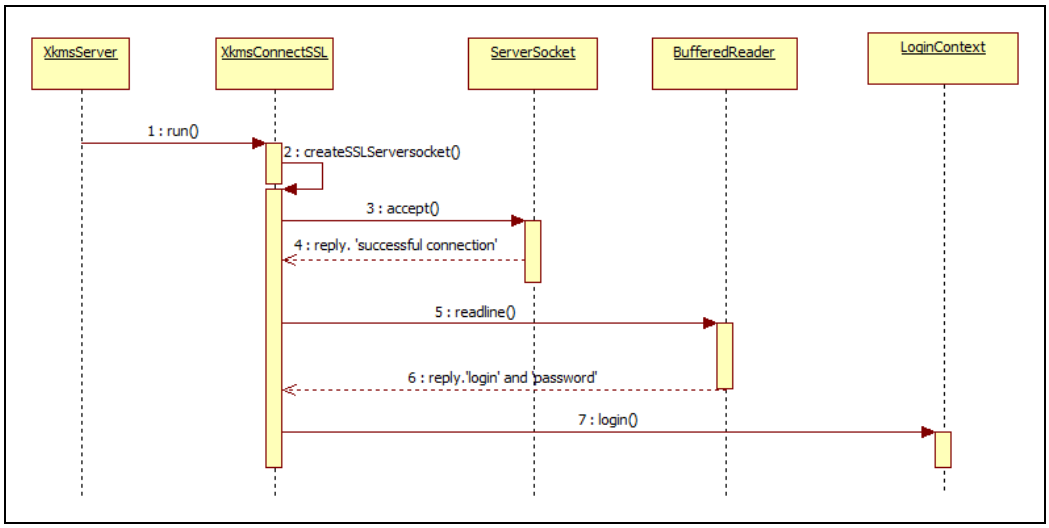

Figura 29- Diagrama de sequência para o *login*.
Fonte: o Autor.

Ao ocorrer a conexão de uma entidade com o XKMS Server, este capta as informações para o processo de autenticação através do método *readline* da classe *BufferedReader* que executa-as através do método *login* da classe *LoginContext*. Ainda terá que ocorrer a comunicação entre o agente e XKMS Server, conforme visualizado na figura 30.

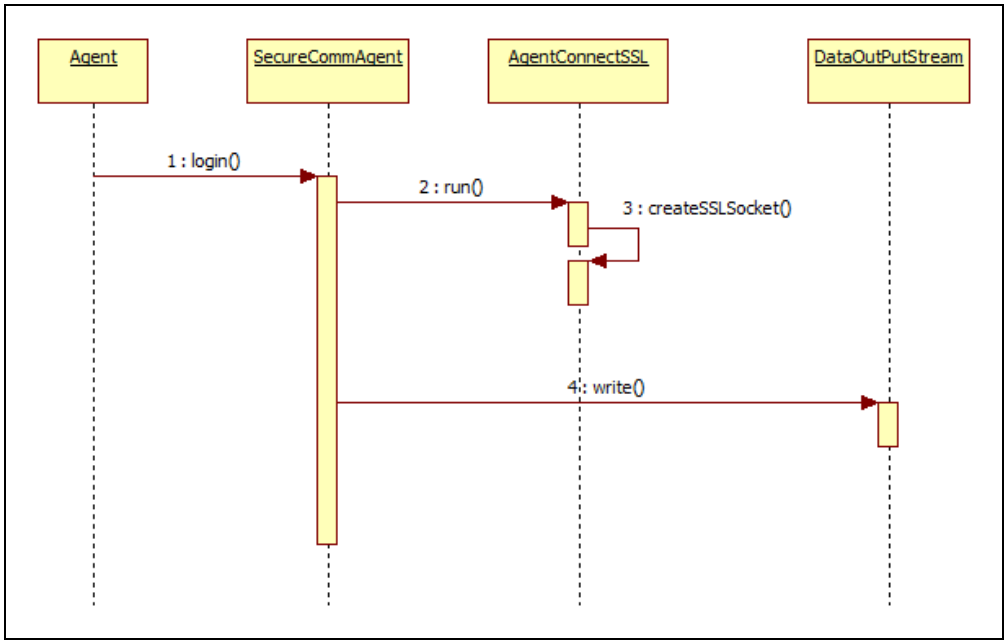

Figura 30- Diagrama de sequência do envio do login.
Fonte: o Autor

A realização do processo ocorre no momento em que o agente aciona o método *loginAgent* da classe *SecurCommAgent* para realizar operação de logar no

XKMS Server, de maneira que suas informações pessoais (*username* e *password*) são preenchidas e enviadas pelo método *run* da classe *AgentconnectSSL*. Depois, será criada através do método *createSSLSocket* uma conexão segura para a partir daí os dados serem enviados ao servidor, atreves do método *write* da classe *DataOutPutStream*.

A seguir é demonstrado um trecho do código utilizado na prototipação para o processo de verificação do *login*.

```
1  public class XkmsConnectSSL {
2     [...]
3     public void run() {
4        ServerSocket listen = null;
5        try {
6            ServerSocket listen = createSSLServerSocket();
7            Socket client = listen.accept();
8            Connection connect = new Connection(client);
9            listen.close();
10       }catch(Exception e){
11       [...]
12       }
13    }
14    class Connect extends Thread {
15       [...]
16       try {
17           in = new BufferedReader(new InputStreamReader
                     (socketclient.getInputStream()));
18           out = new DataOutputStream(socketclient.getOutputStream());
19       }catch (IOException e)
20           this.start();
21    }
22      public void run(){
23       try {
24           [...]
25           msg = in.readLine();
26           [...]
27           username = msg.getBytes();
28           password = msg.getBytes();
29           [...]
30           lc = new LoginContext("Sample", new MyCallbackHandler(username, password));
31           [...]
32           try {
33                    lc.login();
34           } catch (LoginException le) {
35                    out.write(error);
36                    [...]
37           }
38           Subject mySubject = lc.getSubject();
39           [...]
40           out.write(sendKeyConfig);
41           PrivilegedAction action = new ActionAgent(out);
42           Subject.doAsPrivileged(mySubject, action, null);
43           in.close();
44           out.close();
45           socketclient.close();
46         }catch (Exception e)
47      }
48   }
49 }
```

Como pode ser visto no trecho do código acima da classe *XkmsConnectSSL* mostra como ocorre a verificação do *login*. É necessário configurar o socket do servidor tendo como base nas suas credenciais, para tanto chama-se o método da linha 6, *createSSLServerSocket* para autenticação do agente. O servidor, em estado de espera, fica aguardando que o agente realize a conexão através do método *accept* na linha 7. Ao ocorrer à conexão, o agente tem seus dados confidenciais lidos na linha 25 e depois na linha 30 acontece a verificação de autenticidade por um determinado mecanismo. Na linha 33 se dá o início do processo de verificação, para que na linha 38, ao ser confirmado a autenticidade o usuário passe a ter uma validade para o acesso. A seguir, nas linhas 40 e 41, o servidor XKMS disponibiliza

ao agente o recurso que por ele foi configurado e executa a ação, conforme visto na linha 42.

Após ocorrer a autenticação é executado o envio do *login* do agente, que ser demonstrado no trecho da implementação a seguir:

```
1   public class AgentConnectSSL{
2       [...]
3       public AgentConnectSSL(byte[] username, byte[] password,
                        String certpass, String host){
4                [...]
5       }
6       public boolean run(){
7               [...]
8               try{
9                       socket = createSSLSocket(host);
10              }catch(Exception e)
11              try{
12                      DataOutputStream out = new DataOutputStream(socket.
                                getOutputStream());
13                      BufferedReader in = new BufferedReader(new InputStreamReader(socket.
                                getInputStream()));
14                      out.write(username);
15                      out.write(password);
16                      out.flush();
17                      respServer = in.read();
18                      in.close();
19                      out.close();
20                      [...]
21                      socket.close();
22              }catch(Exception e)
23      return success;
24      }
25  }
```

No trecho da classe de conexão do agente que ocorre em um meio seguro SSL a *AgentConnectSSL*, apresenta-se na linha 3 o construtor da classe para receber as informações pertinentes ao agente que são confidenciais e serão utilizadas para o processamento. O método *run* é responsável pelo envio do *login* do agente e encontra-se na linha 6. Já na linha 9, o método da configuração do socket do agente é feito pela *createSSLSocket* onde se tem como base o certificado do servidor. No trecho compreendido da linha 14 a 16, apresentam-se as informações que são escritas e expedidas ao servidor e na linha 17 ocorre de enviar a resposta, caso se ocorreu sucesso ou não na operação.

Há ainda a implementação do mecanismo de segurança que volta-se para o gerenciamento de chaves que usa o *timestamp* e realiza uma comunicação segura ao realizar alterações que são consideradas significativas e que o servidor XKMS realiza o tratamento do tempo de vida útil das chaves que encontram-se no repositório.

Na classe *Security* ao realizar o método *setup* ocorre o registro da chave pública. É o local onde ocorre a ativação das funções responsáveis por gerarem o par de chaves criptografadas através do método *generateRSAKeyPair* , também gera os valores do algoritmo *Diffie-Hellman* pelo método *generateDHKeyPair* e ainda gerar o temporizador que cuida da data de criação da chave pública (*timestmap*) pelo método *generateTimestamp.*

O servidor XKMS realiza uma troca de informações sobre o registro com o agente por intermédio do método *registerTimeKey*, onde o servidor recebe juntos a chave pública e o *timestamp* de criação que depois envia ao agente para ser armazenado em sua base de dados, a sua chave pública e *timestamp* que diz quando a chave que foi registrada irá expirar.

Há no servidor XKMS um método responsável por efetuar a troca de informações de registro, que é *TimeKeyRegistration,* que recebe as informações do agente, realiza o cálculo da validade das chaves pelo método *generateValidityTmp* que se baseia no *timestamp* que é fornecido pelo agente, e ainda, envia a chave pública do servidor junto com o *timestamp* da chave registrada e as armazena no repositório das chaves. Como pode ser visualizada no diagrama de sequência, na figura 31, que trata do registro da chave pública.

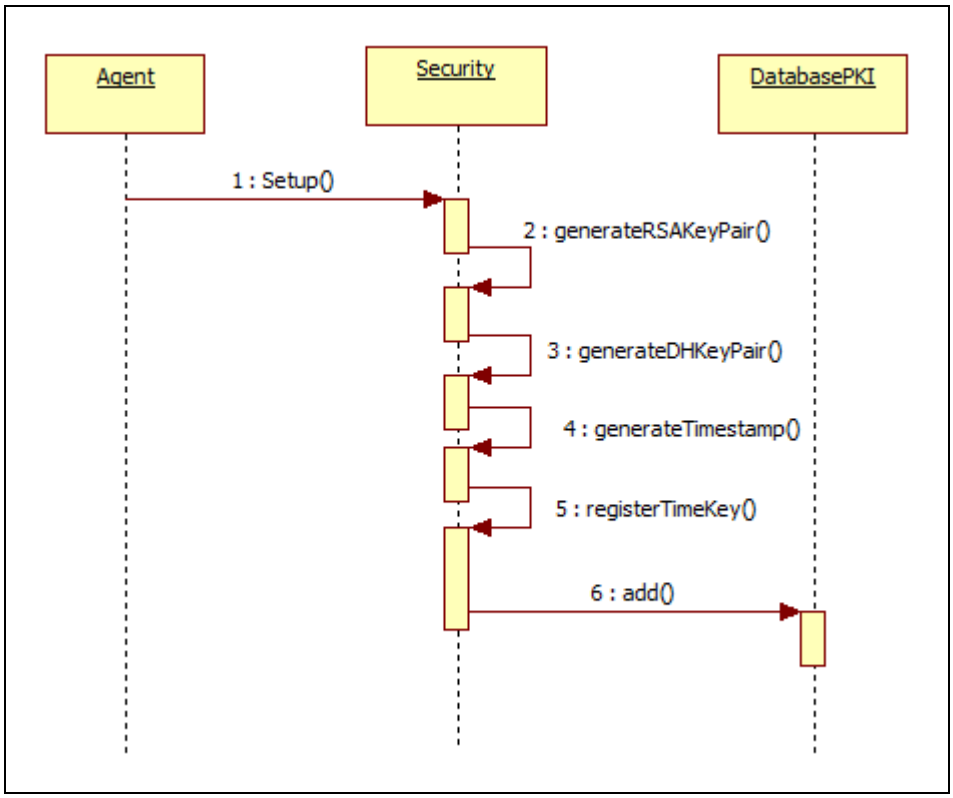

Figura 31-Diagrama de sequência do registro das chaves públicas.
Fonte: o Autor

Um usuário da solução, para que possa enviar uma mensagem de maneira segura, é necessário que previamente seja verificada a validade da chave pública do destinatário através do método *verifyValidityKey.* Caso haja na base de dados do PKI do emissor, a chave do destinatário, o método capta o prazo de expiração da chave e verificar se ainda é uma chave válida ou não. Mas o emissor pode não ter a chave do destinatário armazenada, então pode solicitar ao servidor XKMS, pelo método *locateKey* responsável por solicitar a chave e *timestamp* com a validade da chave. Ao perceber que o prazo das chaves públicas expirou, o agente pode fazer outra requisição de uma chave ao servidor XKMS. A figura 32 demonstra o diagrama de sequência do envio de mensagem segura.

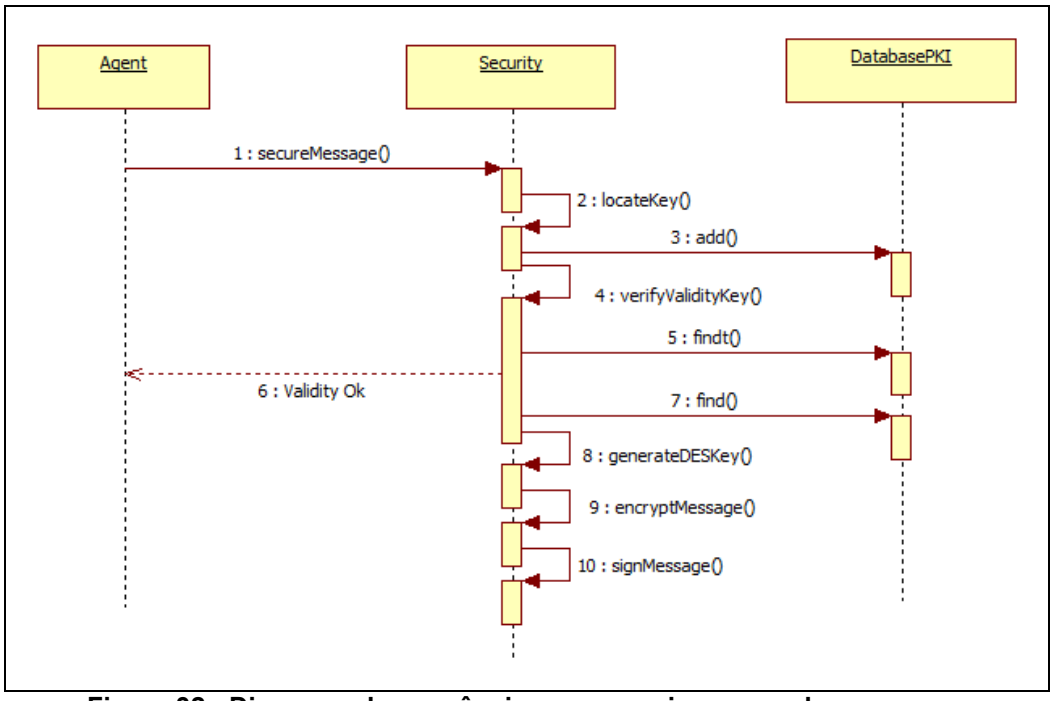

Figura 32 - Diagrama de sequência para o envio seguro de mensagem.
Fonte: o Autor

A seguir será demonstrado um trecho do código de registro da chave pública com o *timestamp.*

```
1   public class Security {
2       [...]
3       public void setup() {
4             [...]
5             rsaKeyPair = generateRSAKeyPair();
6             dhKeyPair = generateDHKeyPair();
7             timeCreateKey = generateTimestamp();
8             if (!agent.getAID().equals(xkmsServer)) {
9                   [...]
10                  timeKeyResponse = (TimeKeyRegistrationResponse) registerTimeKey();
11                  databasePKI.add(new Entry(timeKeyResponse.getTimestamp(),
                          xkmsServer, timeKeyResponse.getPublicKey()));
12            }
13      }
14      private TimeKeyResponse registerTimeKey() {
15             [...]
16             keyOf.setPublicKey(new prototype.security.xkmsServer.ontology.PublicKey());
17             addEncryptedPublicKey(keyOf, rsaKeyPair.getPublic());
18             message = new ACLMessage(ACLMessage.INFORM);
19             message.setSender(agent.getAID());
20             [...]
21             message.addReceiver(xkmsServer);
22             agent.getContentManager().fillContent(message, keyOf);
23             agent.send(message);
24             message = agent.blockingReceive();
25             keyOf = (KeyOf) agent.getContentManager().extractContent(message);
26             timeKeyResponse = new TimeKeyRegistrationResponse();
27             timeKeyResponse.setPublicKey(retrieveEncryptedPublicKey(keyOf));
28             [...]
29             addEncryptedTimestamp(timeOf, timeCreateKey);
30             message = new ACLMessage(ACLMessage.INFORM);
31             message.setSender(agent.getAID());
32             [...]
33             message.addReceiver(xkmsServer);
34             agent.getContentManager().fillContent(message, timeOf);
35             agent.send(message);
36             message = agent.blockingReceive();
37             timeOf = (TimeOf) agent.getContentManager().extractContent(message);
```

```
38              timeKeyResponse.setTimestamp(retrieveEncryptedTimestamp(timeOf));
39              return timeKeyResponse;
40      }
41      [...]
42  }
```

Pode-se visualizar da linha 5 a 7 o método gerador do par de chaves (pública e privada), além do gerador dos parâmetros do algoritmo *Diffie-Hellman* e o gerador do *timestamp*. Na linha 10, é invocado o método *registerTimeKey* pelo método *setup*. Para ocorrer o envio da chave pública ao servidor XKMS, pode-se visualizá-los da linha 16 a 23. Mas antes de ocorrer o envio ao servidor é preciso que a chave pública seja criptografada, linha 17. Da linha 29 a 35 demonstra-se os passos para envio do *timestamp* da criação da chave para o servidor. Na linha 11 tem-se a recuperação e o armazenamento no banco de dados do agente, das informações do valor da chave pública do XKMS Server que encontram-se nas linhas de 24 a 27 e seu *timestamp* das linhas 36 a 38.

O trecho do código a seguir apresenta a verificação do tempo de serviço da chave (*timestamp)*.

```
1   public class Security {
2         [...]
3         public boolean verifyValidityKey(ACLMessage message) {
4               [...]
5               receiver = (AID) message.getAllReceiver().next();
6               receiverValidityKey = databasePKI.findt(receiver);
7               [...]
8               currentTime = generateTimestamp();
9               if(receiverValidityKey.before(currentTime))
10              return false;
11              return true;
12        }
```

A classe Security contem o método *VerifyValidityKey* que pode ser visto na linha 3. Na linha 5 aparece a identificação do destinatário e será aproveitado para capturar o *timestamp*. Caso haja na base de dados do emissor a chave pública do destinatário, há a captura do *timestamp* da chave na linha 6. Na linha 8 obtem-se a captura do tempo decorrente e posteriormente é realizada a comparação com a validade da chave na linha 9, para daí determinar se a chave é uma chave pública valida.

Há também na classe Security os métodos *locateKey* e *secureMessage* que são responsáveis respectivamente por requisitar a chave pública ao servidor e enviar a mensagem de maneira segura. No trecho do código a seguir podemos visualizar o envio de mensagem segura.

```
1    public class Security {
2         [...]
3         public void secureMessage(ACLMessage message) {
4                [...]
5                receiver = (AID) message.getAllReceiver().next();
6                receiverPublicKey = databasePKI.find(receiver);
7                [...]
8                if(receiverPublicKey == null) {
9                      timeKeyResponse = (TimeKeyLocateResponse) locateKey(receiver);
10                     receiverPublicKey = timeKeyResponse.getPublicKey();
11                     databasePKI.add(new Entry(receiverValidityKey, receiver,
                                receiverPublicKey));
12               }
13               encryptMessage(message, generateDESKey(), receiverPublicKey);
14               signMessage(message);
15        }
16        private TimeKeyResponse locateKey(AID holder) {
17               [...]
18               request = new ACLMessage(ACLMessage.REQUEST);
19               request.setSender(agent.getAID());
20               request.addReceiver(xkmsServer);
21               [...]
22               agent.getContentManager().fillContent(request, keyOf);
23               secureMessage(request);
24               agent.send(request);
25               response = agent.blockingReceive();
26               recoverMessage(response);
27               keyOf = (KeyOf) agent.getContentManager().extractContent(response);
28               timeKeyResponse.setPublicKey(retrievePublicKey(keyOf));
29               request = new ACLMessage(ACLMessage.REQUEST);
30               request.setSender(agent.getAID());
31               request.addReceiver(xkmsServer);
32               [...]
33               agent.getContentManager().fillContent(request, timeOf);
34               secureMessage(request);
35               agent.send(request);
36               response = agent.blockingReceive();
37               recoverMessage(response);
38               timeOf = (TimeOf) agent.getContentManager().extractContent(response);
39               timeKeyResponse.setTimestamp(retrieveTimestamp(timeOf));
40               return timeKeyResponse;
41        }
42        [...]
43   }
```

Na linha 6, se constata que caso a chave pública do destinatário esteja presente na base de dados do emissor, é realizada a captação da mesma. Se a mesma não existir na base de dados, daí se chama o método *locateKey* que é acionado no método *secureMessage* na linha 9. Da linha 18 a 24 o *locateKey* faz a solicitação da chave pública do servidor e o *timestamp* da linha 29 a 35. Da linha 25 a 28 a chave pública e *timestamp* das linhas 36 a 39, são recebidas e armazenamento na base de dados do agente, conforme a linha 11. A mensagem é criptografada na linha 13, pois já encontra-se com a chave pública e em seguida é assinada para finalmente ser enviada na linha 14.

7. CONCLUSÃO E TRABALHOS FUTUROS.

Neste trabalho foram explorados problemas relevantes a autenticação e autorização de acesso, em virtude da mudança na utilização de sistemas computacionais considerados tradicionais para um novo paradigma computacional, que se encontra sendo explorado nesta última década.

No capítulo 2, foi realizada uma revisão bibliográfica do que é considerado de mais importante em todos os sistemas, a segurança das informações, de maneira que foram visualizadas algumas das muitas violações que a acometem. Para que se possa efetuar a segurança de um sistema, deve se verificar o cumprimento dos requisitos que estão voltados para tal finalidade. Assim sendo, se constatou que uma das primeiras maneiras que pode ser efetuada a segurança de um sistema é por intermédio da autenticação e autorização de controle de acesso de maneira que possa evitar a ocorrência de alguns ou muitos problemas em um sistema.

O terceiro capítulo explana sobre as muitas maneiras que podem ocorrer o gerenciamento de identidades, através da conceitualização e de alguns modelos existentes. Foi visualizado que alguns protocolos são utilizados exclusivamente para o gerenciamento de identidades. Além de terem sido observados alguns trabalhos voltados ao gerenciamento de identidades que serviram como estado da arte para a realização deste trabalho.

O sistema computacional que foi mostrado no capítulo 4, tratou sobre a computação nas nuvens e as suas características peculiares e essenciais relataram-se também seus modelos de serviços, além de demonstrar algumas das principais ferramentas que são utilizadas para instalação de sistemas nas nuvens.

No capítulo 5 foram apresentadas duas propostas que são voltadas à execução de autenticação. Estas autenticações seriam realizadas em duas maneiras distintas: centralizada e distribuída.

Finalmente no capitulo 6 foram realizadas as validações deste trabalho.

Com a elaboração desta proposta, conseguiu-se obter algumas contribuições consideradas como importante para o trabalho, conforme elencada a seguir:

- Na implementação optou-se pela realização da autenticação realizada de maneira distribuída, pelo fato desta apresentar uma maior tolerância a falhas.

- Desenvolveu-se uma proposta de mecanismo seguro que usufrui um servidor XKMS para efetuar uma comunicação realizada de maneira segura, através da utilização de um gerenciador de chaves, para que somente entidades que pertencem ao círculo de confiança possam ter acesso ao sistema de maneira segura.

- Efetuar a transposição de autenticação, de maneira que sejam mantida a relação de confiança e o compartilhamento de serviços existentes.

- Participação de nuvem, no círculo de confiança através da demonstração de certificação digital que demonstre que esta é participante de uma federação.

7.1. Trabalhos futuros

Como sugestões de trabalho futuro, tem-se o desenvolvimento do gerenciamento de identidade para computação em nuvem móvel, através do smartphone como parte confiável.

Realizar a integração e incorporação de SOA aos sistemas computacionais nas nuvens para aumentar a segurança dos mesmos.

Procurar resolver a autenticação realizada de maneira centralizada, de maneira que se passe o sistema a apresentar uma maior tolerância a falhas.

8.Referencias

Anselmo Junior, Ari Silveira. **Gerenciamento de identidades como um serviço para ambientes de computação em nuvem.** Dissertação (mestrado) – Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação. Santa Catarina, 2011.

Apache. **Docs,** 2013. Disponível em: http://httpd.apache.org/docs/2.2/ssl/#mod-ssl.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H.,Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. **Above the clouds: A berkeley view of cloud computing.** Technical report, EECS Department, University of California, Berkeley, 2009.

Braden, Bob. Clark, David. Crocker, Steve. Huitema, Christian**. Security in the Internet Architecture.** Relatório do workshop da Internet Architecture Board - IAB. 1994.

Burnett, Steve; Paine, Stephen. **Criptografia e segurança: o guia oficial RSA**. Tradução de Edson Furmankiewicz. Campus, Rio de Janeiro - BR. 2002.

Buyya, Rajkumar; Ranjan, Raajiv. Calheiros, Rodrigo N.. **InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services.** In Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2010, pages 13{31. Springer, 2010.

Camargo, Edson Tavares de. **Transposição de autenticação em arquiteturas orientadas a serviços através de identidades federadas.** Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica. Universidade Federal de Santa Catarina, Florianópolis, SC, 2006.

Campos, André L. N. **Segurança da informação controlando os riscos,** Editora Visual Books, Florianópolis, 2006.

Celesti, Antonio; Tusa, Francesco; Villari, Massimo; Puliafito, Antonio. **How to Enhance Cloud Architectures to Enable Cross-Federation.** In Proceedings of the 3rd International Conference on Cloud Computing, IEEE CLOUD 2010, pages 337{345, Miami, Florida, 2010. IEEE Computer Society. *IEEE CLOUD, page 337-345. IEEE,* (*2010*)

CONSEGI - Congresso Internacional Software Livre e Governo Eletrônico (3. : 2010 : Brasília) - **Amãpytuna: computação em nuvem: serviços livres para a sociedade do conhecimento.** -- Brasília : FUNAG, 2010. 135 p. : il.

CSA - Guia da *Cloud Security Alliance* Versão 2.1, **2009 - *Security Guidance for Critical Areas of Focus in Cloud,* *2009.***

Damiani, E., di Vimercati, S. D. C., e Samarati, P.. **Managing multiple and dependable identities.** In IEEE Internet Computing , pages 29–37. IEEE. (2003)

Dantas, Marcus Leal. **Segurança da informação: uma abordagem focada em gestão de riscos.** 152 p. – Olinda: Livro Rápido, 2011.

Dokras, Satchit. Hartman. Bret, Mathers. Tim, Fitzgerald. Brian, Curry. Sam, Nystrom. Magnus, Baize. Eric, Mehta. Nirav. **Cloud WP, 2009 - O papel da segurança na computação em nuvem confiável**. White paper da RSA divisão de segurança da EMC.

Echavarria, Isabel Cristina Satizábal. **Contribución a la Validación de Certificados em Arquitecturas de Autenticación e Autorización**. Tesis (Doctorado. Em Ingenieria Telemática). Universitat Politécnica de Catalunya. Barcelona, 2007.

Feliciano. Guilherme, Agostinho. Lucio, Guimarães. Eliane, Cardozo. Eleri. **Gerência de Identidades Federadas em Nuvens: Enfoque na Utilização de Soluções Abertas**, Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2011.

Ferreira, Aurélio B. de Hollanda. **Novo Dicionário da Língua Portuguesa.** 2. ed. Rio de Janeiro: Nova Fronteira, 1986. 1838 p.

Hovav, Anat and Berger Ron "**Tutorial: Identity Management Systems and Secured Access Control,**" Communications of the Association for Information Systems, Vol. 25, article 42, 2009.

IBM - International Business Machines (IBM), disponivel em http://www.ibm.com/developerworks/br/cloud/library/cl-openstack-cloud/ acessado em agosto 2012.

Izquierdo, Victor Antonio. **"Afinal o que é Segurança da Informação?".** Disponível em http://www.relacionamentodigital.com/afinal-o-que-e-seguranca-da-informacao, 2007. Acessado em 02.fev.2013.

John, Anil. **Reality of XACML PEP-PDP Interoperability**, disponivel em http://www.aniltj.com/blog/2008/09/28/RealityOfXACMLPEPPDPInteroperability.aspx, 2008, acessado em set. 12.

Leancode. http://leancode.com/2007/02/23/openid-protocol-diagram/ acessado em março de 2013 às 14:50h

Mell, Peter. Grance, Timothy. **The NIST Definition of Cloud Computing (Draft) - NIST Special Públication 800-145 (Draft)** Recommendations of the National Institute of Standards and Technology, 2011.

Mello, E. R., Wangham, M. S., Fraga, J. S., and Camargo, E. S. (2006). **Segurança em serviços web**. In Minicursos do SBSeg, VI Simpósio Brasileiro Segurança da Informação e Sistemas Computacionais - 2006 - Santos, SP.

Menezes, A. J.; Oorschot, P. C. V.; Vanstone, S. A. **Handbook of Applied Cryptography**. CRC Press Boca Raton, 1997.

MIT. Massachusetts Institute of Technology. **Kerberos: The Network Authentication Protocol**. Cambridge, 2012. Disponível em: http://web.mit.edu/kerberos/www. Acessado em: junho de 2012.

Moraes, Falkner de Arêa Leão. **Segurança e confiabilidade em IDS baseados em agentes.** Dissertação (Mestrado) - Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís, 2009

Nurmi, Daniel; Wolski, Rich; Grzegorczyk, Chris; Obertelli, Graziano; Soman, Sunil; Youseff, Lamia; Zagorodnov, Dmitrii. **"The eucalyptus open-source cloud-computing system"** em Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid-Volume 00, 2009.

OASIS - **SECURITY ASSERTION MARKUP LANGUAGE –SAML. V2.0**. Oasis, 2007. Disponível em: <http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf>. Acesso em: 12 set. 2012

OMG - **Unified Modeling Language TM (OMG UML)** ***Version 2.5 FTF – Beta 1,*** *disponivel em http://www.omg.org/spec/UML/2.5/Beta1/PDF/ acessado em novembro de 2012.*

OpenID. Disponivel em http://openid.net/, e acessado em fevereiro de 2013.

Openstack - **OpenStack Starter Guide (diablo)** Nov 11, 2011, disponível em: http://docs.openstack.org/diablo/openstack-compute/starter/openstack-starter-guide-diablo.pdf. acessado em março de 2012.

Ramos, Karla Darlene Nepomuceno. **PAPÍLIO: Proposta de um Algoritmo de Criptografia Baseado no Algoritmo de Viterbi e Codificação Convolucional.** Dissertação (Mestrado em Sistemas e Computação) Departamento de Informática e Matemática Aplicada. Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, RN, 2002.

RFC 3127 (rfc3127) - **Request for comments. Authentication, Authorization, and Accounting: Protocol Evaluation.** Disponível em: http://www.elook.org/computing/rfc/rfc3127.html. Acessado em fev. 2013.

Sakuragui, Rony Rogério Martins. **Gerenciamento de Identidades com Privacidade do Usuário em Ambiente Web.** Tese (Doutorado em Engenharia da Computação e Sistemas Digitais) – Departamento de Engenharia de Computação e Sistemas Digitais. Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 2012.

Santos, Daniel Ricardo dos, Westphall, Carla Merkle. **“Uma aplicação de privacidade no gerenciamento de identidades em nuvem com uApprove”.** Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2011.

Seabra, Eduardo Manuel Moreira. **Gestão de Identidades e privacidade em redes de próxima geração.** Dissertação (Mestrado Engenharia Electrónica e

Telecomunicações) – Departamento de Electrónica, Telecomunicações e Informática Universidade de Aveiro, Aveiro, Portugal, 2009.

Stair, Ralph M.; Reynolds, George W. ; **Princípios de sistemas de informação: uma abordagem gerencial.** São Paulo: Pioneira Thomson Learning, 2006.

Stallings, William. "**Criptografia e segurança de redes**." Tradução Daniel Vieira; revisão técnica graça Bressan, Ákio Barbosa e Marcelo Succi. 4ª.ed. –São Paulo: Pearson Prentice Hall, 2008.

StarUML, 2005. **StarUML** Disponivel em http://staruml.sourceforge.net/docs/api-doc/index.html. Acessado em novembro de 2012.

Wangham, M. S., de Mello, E. R., da Silva Böger, D., Gueiros, M., and da Silva Fraga, J. (2010). **Gerenciamento de Identidades Federadas.** In Minicurso - SBSeg 2010 - Fortaleza - CE.