Capítulo 2

Segurança em Grades Computacionais

José de Ribamar Braga Pinheiro Junior, Fabio Kon
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Abstract

Computational grids enable the execution of applications in machines geographically distributed across different administrative domains and institutions. The possibility of computers to interact in this context leads to new security problems and increases the complexity of efficient solutions for resource sharing in computational grids. The security requirements for a computational grid differ from those found on more restricted and controlled environments. This chapter introduces the most relevant security aspects for grid computing, describes existing security mechanisms used on grids, and explains how these aspects are addressed by concrete systems.

Resumo

As grades computacionais permitem a execução de aplicações em computadores geograficamente dispersos e pertencentes a instituições e domínios administrativos diferentes. A possibilidade de computadores interagirem neste contexto traz novos problemas de segurança e aumenta a complexidade de soluções eficientes para compartilhamento de recursos em grades computacionais. Os requisitos de segurança para uma grade computacional diferem daqueles presentes em ambientes mais restritos e controlados. Este capítulo descreve os aspectos de segurança mais relevantes para computação em grades, os mecanismos de segurança existentes para tratá-los e a forma como estes aspectos são tratados em sistemas concretos.

2.1. Introdução

A sociedade em que vivemos é cada vez mais dependente das novas tecnologias. A computação, em particular, surgiu para automatizar muitas tarefas que eram executadas de forma arduosa em um tempo não muito distante. A ciência, em todas suas áreas de conhecimento, é sem sombra de dúvida, uma das beneficiadas pelas tecnologias emergentes oriundas da Ciência da Computação. Biólogos, físicos, matemáticos, engenheiros, químicos, por exemplo, utilizam os computadores em suas simulações, otimizações, mineração de dados, entre outras atividades. A indústria, por sua vez, é favorecida, direta ou indiretamente, por resultados conseguidos através da informática para melhorar os seus processos, obtendo maior qualidades em seus produtos e, conseqüentemente, maiores lucros. O comércio, além das vantagens conseguidas naturalmente pela automatização de suas informações, utiliza-se da Internet para aumentar a venda de seus produtos, colocando-os disponíveis diretamente na casa ou no local de trabalho de seus clientes.

Todas essas facilidades fizeram surgir uma crescente necessidade de poder computacional. Problemas já existentes, ou mesmo novos, ainda não experimentados pela sociedade, demandam grande tempo de processamento. Altos investimentos em equipamentos são feitos para resolver problemas computacionais nas áreas de prospecção de petróleo, simulação metereológica, seqüenciamento de DNA e previsão do tempo. A importância estratégica destas atividades nos dão a idéia da importância da computação dentro da sociedade contemporânea. Os altos custos envolvidos na aquisição de computadores de alto desempenho os tornam inacessíveis para uma grande quantidade de potenciais usuários. Por outro lado, os computadores pessoais de hoje possuem a mesma capacidade de processamento dos supercomputadores de uma década atrás. A possibilidade de que estas máquinas disponibilizem seus recursos computacionais para solucionar os problemas de outros usuários e instituições em uma rede de computadores fez surgir a idéia da Computação em Grade (*Grid Computing*)[1] [Foster and Kesselman, 2003, Foster et al., 2001, de Camargo et al., 2004].

Um sistema de Computação em Grade pode ser definido como uma infra-estrutura capaz de interligar e gerenciar diversos recursos computacionais distribuídos por uma rede de computadores de maneira a oferecer, ao usuário, uso intensivo de recursos e aplicações distribuídas [Berman et al., 2003]. Os recursos disponíveis aos usuários das grades são processamento, memória, periféricos (por exemplo, instrumentos científicos e hardware especializado), componentes de software, entre outros.

As redes de computadores, em especial a Internet, no entanto, foram criadas sem a preocupação com a segurança dos dados por elas transmitidos [Garfinkel and Spafford, 1996]. Quando do surgimento das tecnologias de redes de computadores, as relações de segurança entre as partes envolvidas, que eram confiáveis e conhecidas, não obtiveram a preocupação necessária dos seus projetistas. Mais recentemente, a partir do momento que a arquitetura da Internet firmou-se como o padrão de fato das redes de computadores, estas herdaram esse problema e técnicas adicionais tiveram que ser usadas para manter a segurança das informações.

Os computadores que participam de grades computacionais são particularmente

[1] A origem do termo *Grid Computing* é uma analogia com a rede elétrica (*Power Grid*)

vulneráveis a problemas com a segurança. Em um sistema de grade, os recursos a serem protegidos estão localizados em domínios administrativos diferentes, implicando em um controle de acesso mais difícil. A Computação em Grade exige que as soluções de segurança sejam interoperáveis para permitir a integração com os sistemas de segurança já existentes. Finalmente, os sistemas operacionais e o middleware da Grade, por sua vez, podem não ser seguros o bastante, permitindo o acesso indevido a recursos.

Este capítulo foi organizado da seguinte maneira: a Seção 2.2 introduz o problema de segurança, seus fundamentos e requisitos básicos. A Seção 2.3 discute Grades Computacionais, os preceitos de segurança envolvidos e os mecanismos existentes para tratá-los. Na Seção 2.4, apresentamos estudos de casos de implementações de segurança nas principais grades existentes. Ainda nessa seção, as arquiteturas de cada um destes sistemas serão detalhadas. Finalmente, na Seção 2.5 apresentamos as tendências futuras e considerações finais.

2.2. Segurança

A preocupação com a segurança de dados surgiu no meio militar, onde proteger as informações era o principal objetivo, pois delas dependiam os resultados das batalhas. Historicamente sabe-se que muitas das vitórias ocorridas durante a segunda guerra mundial deveu-se à criptografia de dados. Esconder uma ordem que indicava um movimento das tropas poderia fazer uma diferença em centenas de vidas.

Nos dias atuais, os problemas com a proteção dos dados são mais discretos, mas nem por isso menos importantes. Com a grande participação das tecnologias de redes de computadores nas empresas, nas instituições governamentais e até mesmo nos lares de milhões de pessoas, o perigo de uma possível ameaça às informações privilegiadas se faz presente a cada momento. Já faz parte da rotina diária das empresas a realização de negócios utilizando redes privadas e até mesmo a própria Internet como meio de comunicação. Bancos disponibilizam suas transações financeiras através de redes públicas de dados e a interceptação ou modificação de informações podem levar a resultados desastrosos.

A implementação de sistemas seguros deve ser uma preocupação constante nas equipes de desenvolvimento de sistemas computacionais. Um sistema computacional é seguro se pudermos depender dele e seu software tiver um comportamento esperado [Grafinkel and Spafford, 1996]. Sabemos, no entanto, que criar um sistema computacional totalmente seguro é muito difícil (se não impossível) e devemos definir os limites de risco aceitáveis. Respondendo a questões como: qual o perímetro da rede a ser considerada, quais serviços serão disponibilizados externamente, quais são os controles necessários para o sistema e o nível de segurança pretendido, estaremos delineando o escopo do nosso problema.

Cada sistema computacional tem valores associados aos recursos que devem ser protegidos, alguns tangíveis, outros intangíveis [Lang and Schreiner, 2002]. Recursos tangíveis são aqueles aos quais podemos associar valores diretos, ou seja, podemos quantificar um preço por ele (o hardware, por exemplo). Recursos intangíveis (uma informação, por exemplo) são mais difíceis de avaliar pela dificuldade que temos em definir o quanto vale a informação. Lang [Lang and Schreiner, 2002] sugere que devemos quantificar o custo da perda pois é mais apropriado quantificar o impacto negativo do recurso

comprometido: custo da troca, danos à reputação, perda de competitividade, entre outros.

A relação entre o custo da implementação e o custo da perda do recurso nos proporciona o que chamamos de análise de risco. O objetivo da segurança é minimizar o risco. Os custos da implementação de segurança tais como custo com pessoal, hardware, software e tempo gasto não devem ser maior do que o custo associado ao risco potencial de cada possível ataque. É fácil imaginar que os requisitos de segurança necessários para um ambiente bancário devem ser diferentes daqueles esperados para um ambiente educacional. Encontrar um ponto de equilíbrio entre os custos associados à implementação de segurança e o benefício causado por essa implementação é um dos grandes desafios durante o projeto de sistemas de segurança. A seguir trataremos dos objetivos, das políticas e dos mecanismos disponíveis em um sistema de segurança.

2.2.1. Objetivos, políticas e mecanismos

Quando se projeta sistemas de segurança temos que estar atentos aos objetivos a serem alcançados. Os objetivos são definidos pelas políticas e os métodos utilizados para alcançá-los são chamados de mecanismos de segurança [Sinha, 1996].

A política de segurança define as regras gerais pelas quais é fornecido acesso aos recursos. A política serve para avaliar a segurança de um sistema e definir regras e práticas que regulamentam como devemos proteger e distribuir os recursos. As políticas de segurança não devem conter detalhes técnicos relacionados às suas proposições, mas sim especificar o que deve ser feito e o motivo. Escreve-se esta política de forma amigável e com uma linguagem clara e precisa. Uma política sempre é específica para um determinado ambiente e não pode ser definida de forma generalizada.

Os mecanismos de segurança são os meios através dos quais garantimos que uma determinada política está sendo cumprida; os mecanismos são as ferramentas disponíveis para implementar uma política de segurança. Os mecanismos de segurança podem ser desde procedimentos físicos (como impedir a entrada de usuários em uma determinada sala) ou implementados em hardware ou software (como algoritmos de criptografia ou controle de acesso).

Os objetivos comuns de um sistema de segurança são os seguintes:

- Autenticação

  A autenticação é o processo de estabelecer a validade de uma identidade reivindicada [Ramachandran, 2002]. A autenticação é o primeiro passo na segurança de um sistema computacional; junto à confidencialidade e à integridade, consiste num dos pilares da segurança.

- Não Repudiação

  A Não Repudiação consite em obter provas (ou fortes indícios) de ações realizadas no passado de forma que um indivíduo não possa negar ações que tenha realizada no sistema.

- Confidencialidade

A confidencialidade impede que os dados sejam lidos ou copiados por usuários que não possuem o direito de fazê-lo.

- Integridade de Dados

  A integridade de dados protege a informação de ser removida ou alterada sem a autorização do dono.

- Disponibilidade

  A disponibilidade é a proteção dos serviços para que eles não sejam degradados ou fiquem indisponíveis sem autorização. Isto implica em dados e sistemas prontamente disponíveis e confiáveis.

- Controle

  Um sistema computacional pode possuir diversos recursos. O controle permite que somente usuários conhecidos e que têm direitos de acesso em períodos determinados possam, devidamente, dispor dos recursos.

- Prevenção

  A prevenção é um dos elementos fundamentais em todo sistema de segurança. Garfinkel [Grafinkel and Spafford, 1996] diz que a segurança de computadores consiste em uma série de soluções técnicas para problemas não técnicos. Nós podemos gastar grandes quantias de dinheiro, tempo e esforço em sistemas de segurança mas nunca resolveremos problemas como os defeitos desconhecidos nos softwares (*bugs*) ou funcionário maliciosos. Precaver-se em relação a possíveis problemas é uma boa prática de segurança.

- Auditoria

  Ainda não se conhece um sistema de segurança perfeito, sempre é possível que usuários não autorizados possam tentar acessar o sistema, usuários legitimo tenham efetuado ações erradas ou até mesmo atos maliciosos possam ser praticados. Nesses casos é necessário determinar o que aconteceu, quando, quem foi o responsável e o que foi afetado pela ação. A auditoria deve ser um registro incorruptível de principais eventos de segurança. Através da auditoria podemos nos resguardar das ações dos usuários e até mesmo utilizar mecanismos que impossibilitem que os usuários neguem os seus atos ao utilizar o sistema (não repudiação).

2.2.2. Ameaças, vulnerabilidades e ataques

Uma ameaça pode ser definida como um possível perigo que pode explorar uma vulnerabilidade do sistema computacional [Lang and Schreiner, 2002]. Exemplos de ameaça para um sistema computacional são:

- Comprometimento da informação

  Divulgação deliberada ou intencional de informação.

- Violação de integridade

  Modificação maliciosa ou negligente ou destruição de dados.

- Negação de serviço

  Degradação ou impossibilidade de uso de um serviço.

- Repudiação de uma ação

  Falha em verificar a autenticidade de um usuário e prover um método de registrar este fato.

Uma vulnerabilidade é um ponto onde o sistema é susceptível a um ataque. Uma ameaça pode explorar uma vulnerabilidade para concretizar um ataque a um sistema. Uma vulnerabilidade geralmente surge de forma não intencional; por exemplo, para aumentar o desempenho é comum programadores evitarem a verificação de dados de entrada em programas, o que pode levar a vulnerabilidades. Um exemplo comum de vulnerabilidade é o estouro de *buffer* que tem sido muito explorado nos últimos anos [Lhee and Chapin, 2003]. Esta vulnerabilidade consiste em utilizar a falta de verificação do tamanho e do tipo dos dados de entrada de um programa, alterando o seu estado através da inserção de código malicioso, resultando num desvio para executar o código inserido.

O ataque é a efetivação de uma vulnerabilidade através de uma ameaça. O termo intruso ou atacante é comumente usado para representar aquele que tenta obter um acesso não autorizado. Os ataques podem ser passivos ou ativos [Sinha, 1996]. Os ataques passivos ocorrem quando um intruso tenta obter informações de um sistema computacional sem interferir diretamente no funcionamento deste sistema. Os ataques ativos interferem diretamente no funcionamento normal do sistema, geralmente ocasionando algum prejuízo.

O representante mais expressivo dos ataques ativos é conhecido como vírus de computador. Fred Cohen [Cohen, 1987], um dos primeiros pesquisadores que estudou o vírus de computador, o definiu inicialmente como um programa que infecta outro, modificando-o para incluir-se. Devido à facilidade de troca de arquivos pela rede, os vírus passaram a ser um problema constante de segurança. Através da rede, a contaminação por vírus de computador não mais depende do contato direto de dispositivos de armazenamento dos programas. Os vírus atuais propagam-se com velocidade extrema, através dos diversos serviços oferecidos pela Internet e suas vulnerabilidades, dentre os quais destacam-se o correio eletrônico e a Web.

Os dados, ao trafegarem pela rede, podem sofrer ataques do tipo interceptação, interrupção, modificação ou fabricação [Stallings, 2002], como pode ser visto na Figura 2.1. Na interceptação, os dados são copiados e assim perdem sua confidencialidade. A interrupção ocorre quando os dados são perdidos ou corrompidos. Na modificação, os dados são obtidos, modificados e então reenviados ao seu destino. Finalmente, na fabricação, dados e atividades são criados sem sequer terem saído da sua verdadeira origem.

2.2.3. Criptografia

A criptografia tem como objetivo esconder informações sigilosas de qualquer pessoa que não seja autorizadas a lê-las [Terada, 2000]. A criptologia é o estudo de

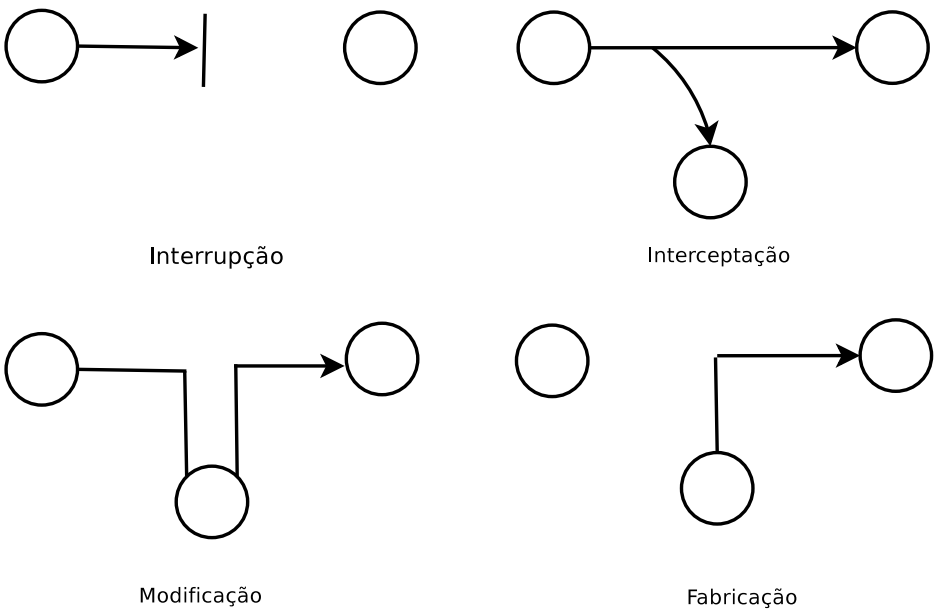

Figura 2.1. ipos de ataques [Stallings, 2002].

técnicas matemáticas relacionadas com a segurança da informação tais como confidencialidade, integridade dos dados, autenticação das entidades e autenticação da origem [Menezes et al., 1996]. A criptografia é então um mecanismo que ajuda a implementar os principais objetivos de segurança de um sistema computacional.

Historicamente a criptografia vem sendo usada pelo ser humano a mais de 2000 anos [Terada, 2000]. A Cifra de César, por exemplo, foi usada pelo Império Romano para esconder informações que eram transmitidas por mensageiros. A cifra de César consistia em deslocar de forma circular 3 posições as letras do alfabeto.

Na Figura 2.2 podemos visualizar um exemplo de uma tabela utilizada para cifrar uma mensagem (cifra de substituição). A primeira linha é o alfabeto na sua seqüência original, enquanto a segunda representa a permutação das letras. Dessa forma duas pessoas que compartilhassem esta tabela poderiam saber que a mensagem "DQJBCM"significa na verdade "GRADES". O ato de procurar a letra trocada na tabela para embaralhar a mensagem é conhecida como encripitar, enquanto a ação contrária correspondente é conhecida como desencriptar. A informação que é compartilhada entre os dois participantes (a tabela) é conhecida como chave criptográfica.

2.2.3.1. Algoritmos criptográficos

Os algoritmos criptográficos são responsáveis pela implementação dos mecanismos de segurança que tornam possível alcançar a maioria dos objetivos de sum sistema de segurança. Existem dois tipos de algoritmos de criptografia: os de chave secreta e os de chave pública (também conhecidos como simétricos e assimétricos, respectivamente). Os algoritmos de chave secreta utilizam uma chave que é compartilhada pelos participantes. Nesse tipo de algoritmo existe o problema de como fazer para compartilhar a chave sem

A	B	C	D	E	F	G	H	I	J	k	L	M
J	A	I	B	C	G	D	F	H	E	N	X	Z

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
R	k	S	L	Q	M	W	Y	P	U	O	V	T

Figura 2.2. Tabela de substituição para cifragem.

que ela fique exposta a intrusos. Nos algoritmos de chave pública cada participante possui um par de chaves: uma privada, conhecida somemente pelo detentor, e outra pública, que pode ser distribuída livremente.

A Figura 2.3 mostra como funciona um esquema de chave secreta. As duas partes, "A" e "B", compartilham uma chave secreta K e desejam se comunicar de maneira que a informação a ser transmitida seja protegida. "A" utiliza a chave que possui e um algoritmo de criptografia para cifrar[2] a mensagem e envia a mensagem no meio de transmissão que estiver disponível. Ao receber o texto cifrado, "B", com a chave que compartilha com seu par, utiliza uma função inversa a usada por "A" e obtém o texto descriptografado, igual ao original.

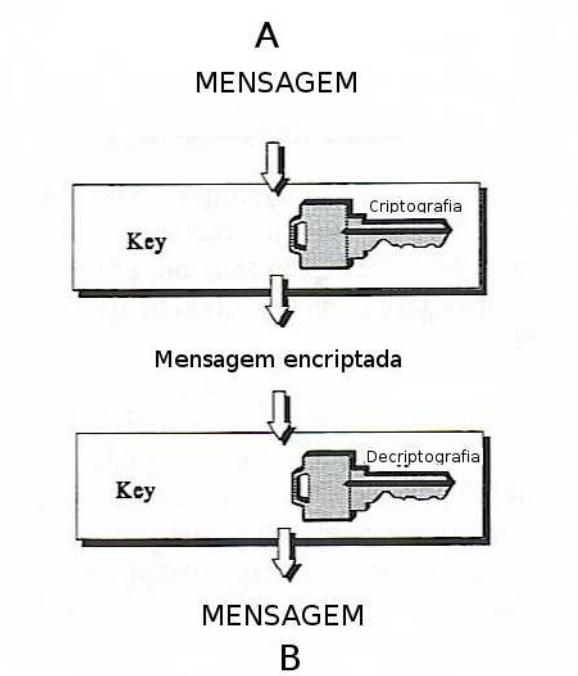

Figura 2.3. Modelo simplificado de chave secreta [NIST SP-800-12, 1995].

Os algoritmos de chave pública foram propostos por Diffie e Hellman em seu

[2]O verbo cifrar é usado no sentido de escrever em cifra, esconder.

famoso artigo *New Directions in cryptography* [Diffie and Hellman, 1976]. No modelo de chave pública, cada usuário possui um par de chaves (S,P), sendo S uma chave secreta e P uma chave pública [Terada, 2000]. Essas duas chaves são relacionadas matematicamente de tal forma que as seguintes propriedades são verdadeiras:

1. S(x) denota a aplicação da chave S no texto x e P(y) = x, ou seja, P é a função inversa de S;
2. O cálculo de S e P é computacionalmente fácil;
3. É computacionalmente difícil calcular S a partir de P;
4. Os cálculos de P() e S() são computacionalmente fáceis;
5. É computacionalmente difícil calcular S() sem conhecer a chave S.

A Figura 2.4 representa a utilização de um algoritmo de criptografia de chave pública. "A" deseja enviar uma mensagem criptografada para "B"; para fazer isso "A" utiliza a chave pública de "B" e transmite a mensagem. "B", quando recebe a mensagem de "A", é o único que consegue recuperar a mensagem por possuir a chave privada correspondente.

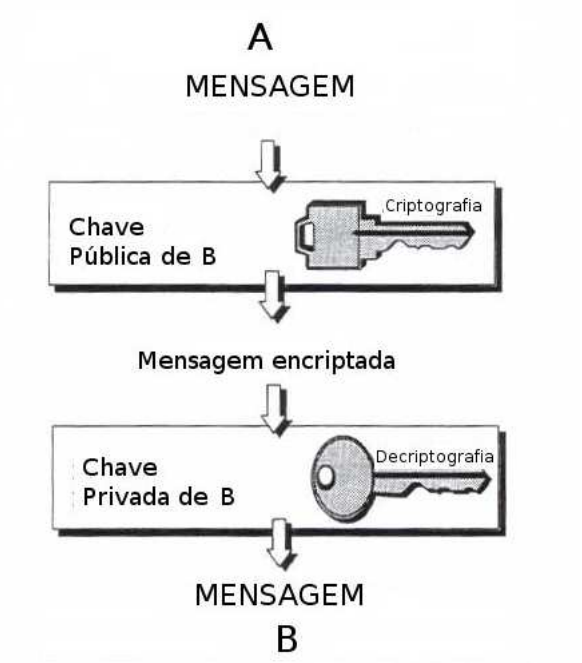

Figura 2.4. Modelo simplificado de chave pública [NIST SP-800-12, 1995].

2.2.3.2. Assinatura digital

Uma assinatura digital é mecanismo criptográfico que tem função similar a uma assinatura escrita à mão [NIST SP-800-12, 1995]. A assinatura possui duas funções primordiais à integridade e a autenticação da mensagem. Vamos considerar que dois usuários

desejam se comunicar utilizando um meio de comunicação de acesso público. A integridade impediria que um usuário mal intencionado alterasse parte da mensagem antes de ela chegar ao seu verdadeiro destino. A autenticação garantiria que a mensagem enviada tenha sido gerada verdadeiramente pelo usuário legítimo. Uma terceira função da assinatura seria a não-repudiação. A não-repudiação permite que, depois da assinatura de uma mensagem, um usuário negue que a tenha feito tal ação.

A Figura 2.5 mostra um processo de assinatura utilizando chaves públicas. Inicialmente o usuário "A" usa sua chave privada para assinar a mensagem a ser enviada para "B". Uma vez recebida a mensagem, "B" utiliza a chave pública de "A" para verificar sua origem. Para diminuir o tempo necessário para a execução do processo é possível utilizar uma função de espalhamento[3] na mensagem e logo depois aplicar o algoritmo de criptografia sobre essa saída, executando o processo inverso no destinatário.

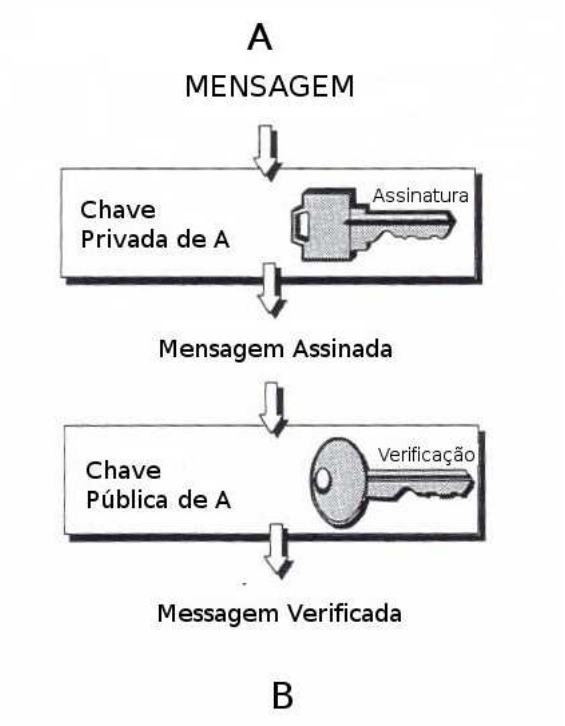

Figura 2.5. Assinatura através de uma chave pública.

2.2.3.3. Certificados digitais

Os certificados digitais são utilizados para garantir que uma determinada chave pública pertença ao seu suposto proprietário. Um certificado é um tipo especial de assinatura digital, ele é um documento que contém a chave pública de um determinado usuário e outras informações adicionais. Este documento é assinado por um terceiro elemento, uma

[3]Uma função de espalhamento (ou *hashing* é uma função matemática que gera uma saída de tamanho fixo e que possui algumas propriedades interessantes entre elas o fato de ser impraticável determinar a entrada a partir de seu hash.

autoridade confiável, atestando a autenticidade da identidade de um usuário. Na Seção 2.3.2.3 trataremos do X.509, um padrão de certificados digitais.

2.2.4. Auditoria

A auditoria é um mecanismo de segurança que tem como objetivo o registro e a análise de atividades relevantes de segurança. A auditoria pode ajudar na detecção de violações de segurança, análise de desempenho ou falhas em aplicações. Manter um registro de eventos é importante pois não é possível garantir se uma determinada política de segurança está correta ou até mesmo se foi implementada corretamente através dos mecanismos de segurança disponíveis.

A auditoria pode fornecer benefícios como o acompanhamento de ações de um usuário, reconstrução de eventos e análise de problemas. Com o acompanhamento de um usuário (ou um processo), as ações ficam associadas com quem as executou, de forma que passa a ser possível realizar uma prestação de contas (não repudiação, por exemplo). Através da reconstrução de eventos, podemos verificar em quais circunstância uma situação relevante ocorreu e quem foi o verdadeiro responsável para que ela tenha ocorrido. Uma análise minuciosa dos problemas de segurança ocorridos facilita a manutenção do sistema de segurança, permitindo que políticas, mecanismos de segurança, o software e o hardware sejam revisados de forma pró-ativa ou preventiva.

A escolha correta dos eventos que deverão ser registrados é um passo importante em um projeto de sistemas de segurança. Escolher o registro de mais eventos do que seria necessário tem um custo computacional muito alto, influenciando no desempenho do sistema como um todo e no uso de recursos. Caso a quantidade de eventos escolhidos seja insuficiente, os registros dos eventos podem ser considerados ineficazes em trazer informações que possam ser utilizadas.

A auditoria é um dos alvos preferenciais dos usuários que tentam atacar um sistema. Paralisando, destruindo ou compremetendo a integridade do sistema de auditoria, o atacante dificulta ou impossibilita ao administrador do sistema que ele e/ou seus atos indevidos sejam descobertos. Políticas de segurança como o uso de mecanismos de criptografia, cópia rotineira dos arquivos de eventos e proteção física das máquinas devem ser consideradas sempre que um sistema de segurança for implementado e isso for relevante.

A seguir apresentamos as Grades Computacionais, seus conceitos básicos, requisitos de segurança e mecanismos disponíveis para implementar segurança em Grades.

2.3. Grades Computacionais

Hoje em dia, apesar de todo avanço da computação nos últimos anos, ainda existe uma necessidade desproporcional de recursos computacionais para resolver alguns problemas específicos em diversas áreas. O custo de máquinas poderosas que possuam os recursos necessários para resolver estes problemas ainda é demasiadamente alto, particulamente para intituições de pequeno e médio porte. As grades computacionais surgiram para tentar dar alento a esta questão; elas são sistemas distribuídos que permitem escalonar processos em máquinas espalhadas por diversas organizações e domínios administrativos.

As grades computacionais estão atualmente em evidência; diversos pesquisadores no mundo todo, nas mais diversas áreas da Computação, investem recursos humanos e financeiros em pesquisas sobre o tema. A indústria já começa a investir no desenvolvimento de sistemas que viabilizem este modelo. Grandes empresas como IBM, Microsoft, NEC, Sun e Oracle investem tempo e dinheiro para desenvolver padrões e sistemas de Grades Computacionais. Algumas questões ainda necessitam de muito trabalho para considerarmos que o desenvolvimento de sistemas de grades esteja totalmente consolidado, sobretudo a Segurança. Entre as forças relacionadas ao desenvolvimento de Grades computacionais [de Camargo et al., 2004] podemos destacar:

- Executar em diversas plataformas sobre os diversos sistemas operacionais pré-existentes, não exigindo assim nenhum tipo de substituição;
- permitir o compartilhamento eficiente de recursos computacionais, tais como processador, memória, disco, outros tipos de hardware e software;
- oferecer qualidade de serviço tanto aos usuários das aplicações quanto aos proprietários dos recursos;
- prover suporte a aplicações paralelas;
- oferecer autonomia para usuários e administradores, evitando impor políticas rígidas de utilização e oferecimento dos recursos;
- ser escalável, podendo englobar de poucas máquinas até possivelmente milhões;
- possuir baixo custo de instalação;
- não causar sobrecarga perceptível ao usuário; e
- ser fácil de usar, permitindo que aplicações pré-existentes sejam facilmente adaptadas ao novo contexto.

O padrão arquitetural *Grid* [de Camargo et al., 2004] apresenta uma arquitetura para grades computacionais (Figura 2.6). Este padrão arquitetural tem como usos conhecidos os sistemas Globus [Foster and Kesselman, 1997], Condor [Epema et al., 1996], InteGrade [Goldchleger et al., 2003] e OurGrid [Andrade et al., 2003] descritos na seção a seguir. Nesse padrão, o agente de acesso é o ponto de acesso primário do usuário à Grade e deve ser executado em cada nó de onde os usuários submeterão aplicações à Grade. Através dele o usuário pode, ao submeter aplicações, definir seus requisitos, monitorar e coletar os resultados das execuções. O serviço de provisão de recursos deve ser executado em cada nó que disponibiliza seus recursos para a Grade; ele gerencia e provê informações sobre a disponibilidades destes recursos. O serviço de provisão de recursos é responsável por permitir a execução de aplicações no nó, reportar erros e devolver os resultados do processamento ao usuário ou a outros nós da aplicação distribuída. O serviço de monitoramento de recursos é responsável por monitorar os estados dos recursos e prover informação sobre o estado destes recursos a outros serviços da Grade. O serviço de escalonamento gerencia os recursos disponíveis na Grade e é responsável, baseado nas informações recebidas do serviço de monitoramento, por escolher nós aptos a executar uma aplicação cuja execução foi solicitada.

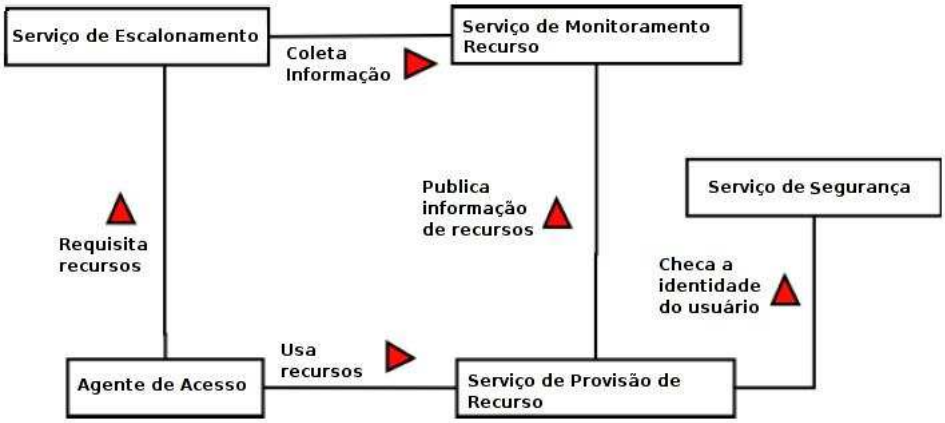

Figura 2.6. Arquitetura de uma Grade computacional.

O serviço de segurança é responsável por três tarefas principais:

1. Proteger os recursos compartilhados, de forma que os usuários que compartilham seus recursos na Grade não sofram os efeitos de aplicações maliciosas.
2. Gerenciar as identidades dos usuários, permitindo relações de confiança e responsabilidade.
3. Proteger as trocas de informações entre os participante da Grade, provendo confidencialidade e integridade.

As grades computacionais ainda apresentam alguns desafios para serem resolvidos. A Segurança em Grades Computacionais, sobretudo, é um tema relevante dentro do contexto anteriormente delineado. Nas subseções a seguir apresentamos os requisitos e os mecanismos de segurança disponíveis relevantes às grades.

2.3.1. Requisitos de segurança

Sistemas de Grades Computacionais são inerentemente mais vulneráveis a ameaças de segurança do que sistemas tradicionais. Em uma grade existe, potencialmente, uma quantidade grande de usuários, recursos e aplicações sendo administradas por diferentes domínios administrativos. Neste ambiente, as regras de segurança definidas em cada sítio podem ser significantemente diferentes. As políticas definidas para um usuário local podem ser ineficientes em um ambiente de Grade, seja porque são permissivas em excesso ou porque são proibitivas demais, forçando aos participante um isolamento pouco produtivo.

As organizações que participam de uma grade computacional geralmente possuem investimentos em sistemas de segurança já existentes [Foster and Kesselman, 2003]. Em uma grade pode existir a necessidade da convivência de vários protocolos de segurança. Os mecanismos diferentes podem exigir interoperabilidade segura entre máquinas e ambientes heterogêneos.

A forma de autenticação e autorização em uma Grade é um requisito importante na definição de um sistema de segurança para a Grade. Pelos requisitos definidos anteriormente, a autenticação na Grade deve permitir a interoperabilidade entre diversos

mecanismos de segurança; isso implica que temos que transformar um mecanismo de segurança em operação de autenticação [Foster and Kesselman, 2003]. Da mesma forma, a autorização poderia fornecer suporte a diversos mecanismos de controle de acesso.

A possibilidade de delegação de direitos de acesso aos elementos de uma Grade é um requisito importante a ser definido. A delegação ameniza o trabalho do administrador em ter que ceder direitos a membros da Grade. Sempre que um dono de recurso confiar em alguém para usar este recurso, este poderá delegar este direito, por um tempo determinado, a um terceiro desde que o sistema assim permita. A delegação de direitos, no entanto, deve ser usada de uma forma restrita. Uma corrente de redelegação muito grande pode fazer com que o recurso acabe sendo usado indevidamente por usuários maliciosos. Os sistemas de segurança de grades devem conter dispositivos para minimizar este problema.

No ambiente de grades surge um problema que não é comum em ambiente tradicionais: devemos garantir que um recurso não seja provido por um atacante. Um usuário mal intencionado, por exemplo, poderia disponibilizar recursos na Grade com a intuito de obter informações privilegiadas. Uma boa política de autorização diminui o risco deste problema acontecer. Em geral o "princípio do menor privilégio" é uma boa opção para a definição de políticas de segurança na Grade. Ou seja, não convém dar ao usuário mais direitos do que ele necessita, mesmo que isso gere um aumento no custo da administração do sistema.

A auditoria é um dos requisitos importantes a ser considerado em uma grade. A diversidade de usuários e de recursos aumenta a probabilidade de surgirem ameaças; eliminar as vulnerabilidades de um sistema nem sempre é possível, principalmente em uma ambiente dinâmico. Os administradores da Grade devem ter mecanismos de rastreamento que permitam verificar ameaças ou erros.

A confidencialidade e a integridade são requisitos importantes para as grades computacionais. Possuir mecanismos que impeçam, ou pelo menos dificultem muito, a modificação ou a obtenção de dados é muito importante. Por questão de desempenho, estas possibilidades devem ser opcionais para os membros.

Estes são os requisitos básicos de segurança necessários para uma grade computacional. A seguir apresentamos os mecanismos de segurança disponíveis para implementar estes requisitos.

2.3.2. Mecanismos de segurança

Nesta seção apresentamos os mecanismos de segurança disponíveis para implementar segurança em grades. Estes mecanismos são apropriados para utilização em sistemas distribuídos e sua aplicação no ambiente de grades computacionais é, em geral, possível. Os mecanismos a seguir descritos são usados em sistemas de rede há anos e passaram ao longo do tempo pr várias melhorias. Esses mecanismos, apesar de tudo, possuem falhas que serão apontada em cada subseção.

2.3.2.1. Kerberos

O Kerberos [Kohl and Neuman, 1993] é um protocolo de autenticação em rede que permite a autenticação de usuários e serviços. Este protocolo oferece confidenciabilidade,integridade e assinatura para as mensagens transmitidas sobre a rede. O protocolo foi criado na década de 80 pelo MIT (Massachusetts Institute of Technology) (`www.mit.edu`) e até hoje ainda é bem utilizado em sistemas distribuídos. Durante anos ele vem sendo aprimorado e encontra-se na sua versão 5. O sistema possui uma implementação disponível sobre forma de código aberto (`http://web.mit.edu/kerberos/www/`).

O Kerberos possui uma estrutura de dados associada a chaves criptográficas, denominada *ticket*. O *ticket* é apresentado sempre que houver a necessidade de autenticação em um serviço ou aplicação. O *ticket* é criptografado e contém informação do identificador do cliente e uma chave aleatória gerada pelo Kerberos.

A arquitetura do Kerberos é constituída por três módulos. O servidor de autenticação (AS - *Autentication Server*), o servidor de concessão de *tickets* (TGS - *Ticket Granting Server*) e um banco de dados. O AS é responsável pela autenticação do usuário na rede. O TGS é um servidor que gera credenciais para comunicação entre clientes e servidores de aplicação. O banco de dados é responsável por armazenar as chaves secretas dos usuários e dos serviços que deverão se autenticar junto ao Kerberos.

O Kerberos usa chaves secretas compartilhadas para proteger a comunicação e permitir autenticação de usuários e de servidores de aplicação. Cada usuário possui uma senha para autenticação. Uma chave criptográfica derivada desta senha é armazenada no banco de dados. Da mesma forma, o servidor de aplicação possui uma chave criptográfica para autenticação.

O processo de autenticação, apresentado na Figura 2.7, é descrito a seguir. Inicialmente um cliente envia para o AS uma solicitação de autenticação. O AS responde com uma credencial criptografada com a chave do cliente. A credencial consiste em um *ticket* para o TGS, denominado TGT (*Ticket-Grainting Ticket*), e uma chave de sessão. Uma chave de sessão é uma chave secreta compartilhada entre o Kerberos e um cliente. Este *ticket* é composto pela identidade do cliente e uma cópia da chave da sessão e é assinado com a chave do servidor TGS. Este *ticket* é usado para requisitar credenciais ao TGS.

Uma vez autenticado, o usuário pode solicitar chaves de sessão para servidores de aplicação. De posse da credencial conseguida no processo de autenticação, o cliente solicita ao TGS uma credencial para um determinado servidor de aplicação; o TGS responde com duas mensagens: um *ticket* criptografado com a chave da sessão TGS para o cliente e outro criptografado com a chave secreta do servidor de aplicação (TA e TB, na figura, respectivamente); ambos os *tickets* possuem uma chave idêntica. O cliente envia ao servidor de aplicação a credencial obtida e este responde com uma mensagem criptografada com a chave compartilhada entre o cliente e o servidor de aplicação, o que confirma sua autenticação.

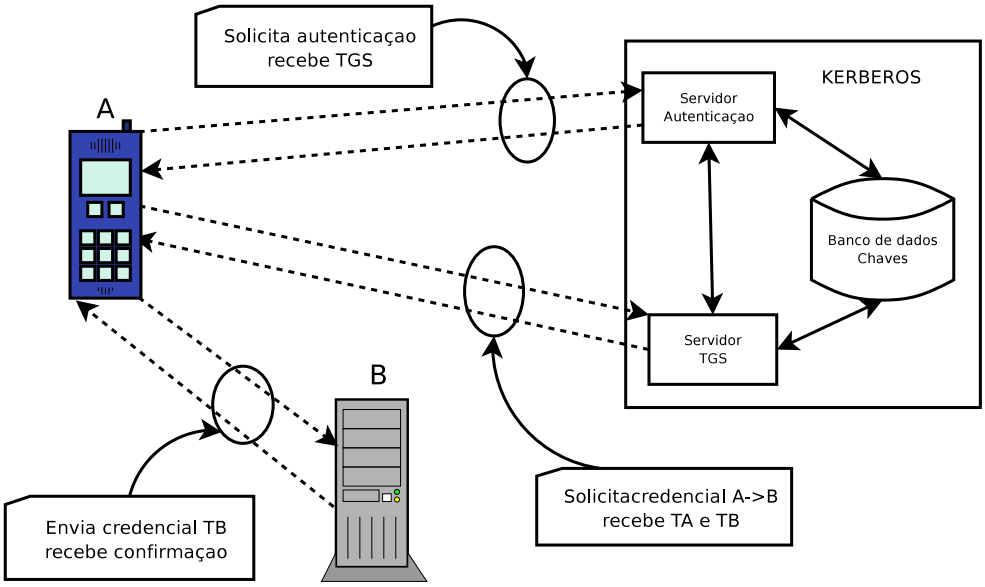

Figura 2.7. Protocolo do Kerberos.

Devido à impossibilidade de todos usuários estarem cadastrados no mesmo servidor, o Kerberos possui a noção de domínios administrativos (*realms* na nomenclatura Kerberos). Esta divisão muitas vezes é feita sobre limites organizacionais. Cada domínio possui sua própria base de dados e seus próprios servidores. Para um usuário utilizar o Kerberos, ele deve estar cadastrado em um domínio. O identificador de um usuário possui a forma NOME_USUÁRIO@DOMÍNIO_USUÁRIO.

Existe também a possibilidade de um usuário de um domínio poder se autenticar em um outro domínio administrado por uma instituição diferente. Para obter chaves de um domínio remoto, o usuário deve solicitar ao seu próprio TGS um *ticket* aceito pelo TGS remoto. Se o TGS remoto tiver seu registro no TGS local, o TGS local entregará ao usuário um *ticket* do domínio remoto. Deste ponto em diante, o usuário usa esse *ticket* para solicitar credenciais para os servidores presentes em outros domínios. Uma outra opção permitida pelo Kerberos é a hierarquia de domínios, de maneira que para contatar um serviço em um outro domínio, pode ser necessário contatar TGSs remotos em um ou mais *realms* imediatos. O nome de cada domínio é armazenado no *ticket*.

Existem algumas opções avançadas disponíveis no protocolo Kerberos através de opções presentes no ticket. A seguir uma breve descrição de cada uma delas.

- INITIAL

  Indica que um *ticket* foi criado usando o protocolo do AS e não via TGS. No Kerberos é possível um cliente solicitar uma credencial a um servidor diretamente do AS. Isso às vezes é interessante para alguns tipos de aplicação de modo a garantir um tempo curto entre a autenticação e a utilização do recurso.

- INVALID

  Indica que o *ticket* é inválido. Servidores de aplicação devem rejeitar este *ticket*. Estes *tickets* podem se tornar válidos através de uma solicitação de validação de *ticket* ao TGS.

- RENEWABLE

  Os *tickets* no Kerberos, por questão de segurança, possuem tempo de vida limitado (cerca de 8 horas). Um *ticket* com este sinal ativado pode ser renovado pelas aplicações (desde que ele já não tenha expirado). Ele possui dois tempos de expiração, quando o *ticket* deve expirar e o tempo a partir do qual ele não mais pode ser renovado.

- POSTDATED

  Um sistema em lote que deve executar em um horário pré-determinado pode necessitar de um **ticket** para uso futuro.

- PROXYABLE

  Indica que o cliente pode permitir que um determinado serviço se comporte como ele. Para limitar o mau uso deste **ticket**, somente endereços de rede listados podem apresentar um *ticket* deste tipo.

- FORWARDABLE

  Esta opção é um caso específico da *PROXYABLE*, a diferença é que, neste caso, o serviço assume a identidade do cliente.

O Kerberos possui algumas limitações bem conhecidas [Bellovin and Merritt, 1990]:

- O Kerberos faz cache de seus *tickets* no diretório temporário do sistema operacional hospedeiro. Ou seja, a segurança do protocolo depende da segurança do sistema do arquivos da máquina onde o Kerberos está sendo executado;
- Como o Kerberos utiliza o tempo para impedir que as mensagens que trafegam na rede sejam retransmitidas ao destinatário (ataques do tipo *replay*), o protocolo exige que as máquinas estejam sincronizadas. Isto pode gerar uma negação de serviço para alguns clientes legítimos;
- As aplicações podem ser modificadas e usadas para capturar senhas dos usuários.

2.3.2.2. SESAME

O SESAME [Sesame, 2003] (*Secure European System for Applications in a Multi-vendor Environment*) é um projeto de desenvolvimento e pesquisa europeu que oferece tecnologia de assinatura única com controle de acesso distribuído baseado em papéis e troca de dados criptografados para sistemas distribuídos. O sistema tem sido aprimorado ao longo dos anos e a sua versão corrente do SESAME é a V4.

A arquitetura do SESAME provê as seguintes características:

- Autenticação simples ou mútua;
- Confidencialidade e integridade na comunicação de dados;
- Controle de acesso baseado em papéis;
- Delegação de direitos;
- Serviço de auditoria;
- Suporte a vários domínios.

De forma resumida, o SESAME trabalha da seguinte forma. Inicialmente o usuário se autentica em um servidor de autenticação e recebe um pacote especial de identificação (*token*) que prova sua identidade. O usuário apresenta este *token* a um servidor de atributos de privilégios que disponibiliza um certificado de controle de acesso. Sempre que for necessário acessar um recurso protegido, o usuário apresenta este certificado ao detentor do recurso que toma suas decisões de acordo com os seus atributos de segurança ou outras informações adicionais, como por exemplo uma lista de controle de acesso. O SESAME permite que os direitos de acesso de um determinado usuário possa ser delegado a um terceiro.

A Figura 2.8 ilustra os componentes do SESAME. O SESAME está organizado em três grupos: os componentes do cliente (*Client-side components*), os componentes do servidor de segurança do domínio (*Domain Security components*) e, finalmente, os compontes do servidor (*Server-side components*) [Tanenbaum and Steen, 2002].

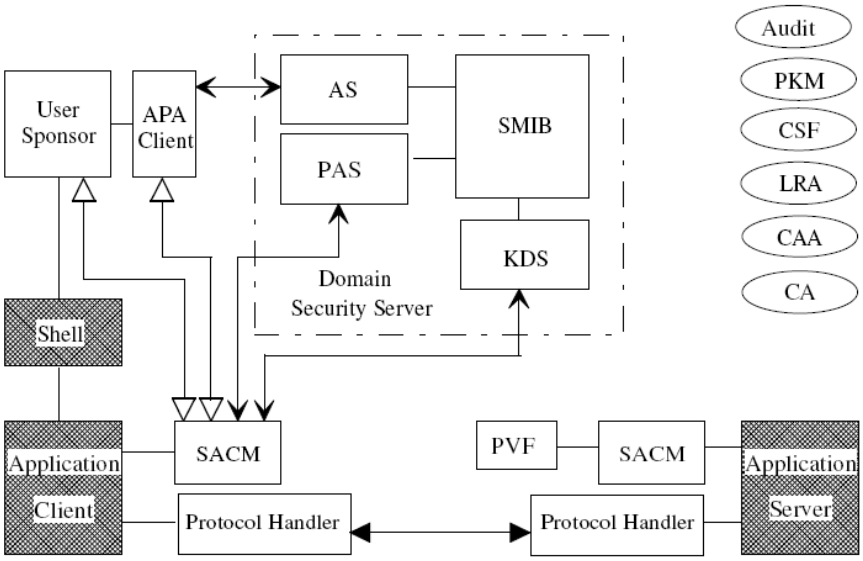

Figura 2.8. Arquitetura do SESAME [Sesame, 2003].

Componentes do servidor de segurança

Os componentes do servidor de segurança são agrupados em uma máquina conhecida como DSS (*Domain Security Server*). Sua função é permitir a autenticação, autorização e distribuição de chaves. O servidor de autenticação (AS - *Authenticator Server*) é responsável pela autenticação de usuários e programas. O SMIB (*Security Menagement Information Base*) é uma base de dados que armazena chaves privadas (das aplicações) e informações relevantes à implementação da segurança no servidor. O PAS (*Privilege Attribute Server*) providencia uma lista de certificados que determinam os direitos dos clientes às aplicações. Ao fazer uma requisição, um usuário recebe um certificado de privilégio (PAC - *Privelege Attribute Certficate*).

O PAC é o elemento central para o SESAME. Ele possui informações sobre uma sessão em particular e é assinado pelo PAS. O PAC permite a delegação temporária dos seus direitos a um outro usuário ou servidor. A delegação de um PAC permite também que uma determinada entidade seja anônima (ou melhor, pseudo-anônima). Para fazer isto, o cliente obtém um PAC delegado que não contém sua identidade e o delega para um terceiro que atua com seus direitos. Desta forma, o elemento final, o servidor, não tem conhecimento sobre a identidade do seu cliente.

Componentes do lado do cliente

No lado do cliente, o SESAME possui os seguintes componentes: o fiador do usuário (*User Sponsor*), o cliente de autenticação e privilégio (*APA Client - Authentication and Privilege Access Client*) e o gerenciador de contexto de segurança (SACM - *Secure Association Context Managemement*). O primeiro permite ao usuário entrar, sair e trocar os seus papéis de acordo com a necessidade de um determinado contexto. O programa que executa a função do fiador do usuário possui uma relação de confiança e interage com o sistema operacional hospedeiro do cliente para implementar algumas de suas funções. Estas funções incluem, por exemplo, gerar chaves secretas a partir de uma senha para comunicações com o servidor do domínio de segurança. O cliente de privilégio e autenticação, implementa uma biblioteca que permite ao usuário e às aplicações interagirem com o servidor de segurança do domínio. E, por último, o gerenciador de contexto da segurança é responsável por iniciar e manter as informações necessárias para a comunicação segura com o servidor de aplicações.

Os privilégios atribuídos aos clientes durante a requisição de um PAC são definidos durante o processo de *logon*. O usuário pode especificar valores padrão associados a um papel que ele assuma ou atributos específicos. No primeiro caso, todos os privilégios associados ao seu identificador e ao seu papel padrão são atribuídos. No caso do cliente indicar somente um papel especifico, tomará somente os privilégios daquele papel. Caso deseje, o cliente pode solicitar somente atributos específicos, o que pode ser interessante no caso de uma delegação.

Componentes do lado do servidor

Assim como o cliente, o servidor também possui um gerenciador de contexto de segurança (SACM), com função análoga. O servidor possui ainda, um componente cuja a função é validar e extrair todas as requisições necessárias das requisições de entrada, o PVF (*PAC Validation Facility*). Assim, por exemplo, o PVF descriptografa chaves de

sessões, extrai os direitos de acessos dos PACs recebidos do cliente e verifica assinaturas.

2.3.2.3. Infra-estrutura de Chaves Públicas

Uma infra-estrutura de chaves públicas (ICP ou PKI - Public Key Infrastructure) reúne um conjunto de hardware, software, políticas, e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados de chaves públicas [Adams and Lloyd, 2002]. O ICP foi concebido como um ambiente seguro e eficiente para oferecer serviços de autenticação e segurança baseados nas técnicas de criptografia de chave pública (ver subseção 2.2.3.1).

Entre as tarefas de uma ICP estão o registro de entidades, a inicialização, a certificação, a recuperação e atualização de chaves. Os três primeiras tarefas são conhecidas como matricula da entidade. O registro corresponde ao processo de identificação da entidade, esta tarefa depende das políticas definidas pela ICP. O registro de uma entidade que represente um banco deve ser algo tratado com mais rigor do que a definição de um simples usuário, por exemplo. A inicialização define o mecanismo que será associado a entidade que se registrou, esta ação corresponde etapa da criação das chaves públicas. O processo de inicialização pode ser efetuado pelo ICP ou até mesmo pela própria entidade registrada. A certificação é a conclusão do processo de matricula de uma entidade. Nessa tarefa será criado o Certificado de Chave Pública. O ICP é responsável também pela recuperação e atualização das chaves. A recuperação ocorre quando, por qualquer motivo considerado válido pela ICP, a recuperação da chave for necessária. Os eventos que poderiam originar a perda de chave poderiam ser a quebra de mecanismos de armazenamento, esquecimento, entre outros. Quando um determinado certificado expira é necessário a criação de novas chaves para a entidade. A ICP pode gerar chaves novas e emitir um novo certificado.

Um Certificado de Chave Pública é gerado sempre com um prazo de validade muito grande. Porém, em certas situações, é necessário que um certificado seja considerado não válido e seja revogado. Cabe ao ICP revogar o certificado antes do prazo de expiração e publicar esta informação. O ICP possui uma lista de certificados revogados que é utilizada para verificar um certificado cuja a data de expiração não foi alcançada.

O ICP é composto dos seguintes elementos:

Entidade Final

A Entidade Final corresponde aos elementos que usam a infra-estrutura. As Entidades Finais são representadas pelos usuários, roteadores, servidores, e outros elementos que possam utilizar um certificado de chave pública.

Autoridade Certificadora

A Autoridade Certificadora (AC) é elemento central da infra-estrutura de chaves públicas. Ela é responsável pela emissão de certificados de chaves públicas e de listas de certificados revogados. A Autoridade Certificadora pode, por questões administrativas, delegar muitas das suas tarefas a outros elementos da arquitetura como a Autoridade de Registro ou um Emissor de Lista de Certificados Revogados.

Autoridade de Registro

A Autoridade de Registro (AR) é uma interface entre a Autoridade Certificadora e o usuário do ICP. Essa entidade é opcional na arquitetura, podendo ser usado para a recepção de pedidos de registros de um Entidade Final, verificação da autenticidade dos requerentes, entre outras. A Autoridade de Registro, no entanto não pode emitir um certificado de chave pública, pois esta tarefa é exclusiva da Autoridade de Registro. A opção de delegar algumas tarefas da AC para a AR pode ser interessante por questões administrativas e de segurança. Uma empresa pode, por exemplo, terceirizar as tarefas administrativas de uma ICP com restrições como a impossibilidade de registro de Entidades Finais.

Emissor de Lista de Certificados Revogados

O Emissor de Lista de Certificados é responsável pela emissão e controle das Listas de Certificado Revogados. Este componente é opcional e recebe a delegação de suas tarefas da Autoridade de Registro.

Repositórios de Certificados

O Repositório de Certificados é responsável por armazenar e recuperar certificados e lista de certificados revogados. Um Repositório de Certificados refere-se a qualquer método de armazenamento e recuperação de associadas a uma Infra-estrutura de Chave Pública, podendo ser implementado através de diversos protocolos como: LDAP (Ligthweigth Directory Access Protocol), FTP (File Transfer Protocol) ou até mesmo HTTP (Hyper Text Transfer Protocol.

Hierarquia de Certificados

As Autoridades Certificadoras poderão ser interligadas formando uma estrutura hierárquica como visto na Figura 2.9. No modelo de hierarquia, as Autoridades certificadoras emitem certificados para as outras de um nível mais baixo. Em uma estrutura hierárquica todas as ACs conhecem a chave pública da AC que está na raiz da hierarquia.

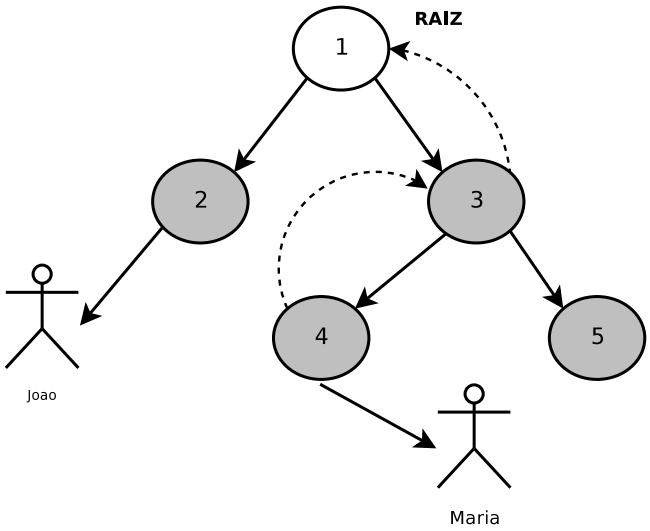

Figura 2.9. Hierarquia de Autoridade Certificadora.

A estrutura hierarquia permite que os certificados possam ser validados através do

caminho dos certificados da CA raiz (número 1 na figura). Vamos supor, de acordo com a figura, que João quisesse verificar o certificado de Maria. Para fazê-lo ele teria que validar o certificado de chave pública de Maria. Depois dessa verificação João teria que validar o certificado da Autoridade Certificadora de Maria, 3 na figura. Até que alcance a raiz que possui chave pública conhecida. A linha tracejada apresenta na figura o que chamamos de caminho de certificação.

Uma outra opção de interligação de Autoridades é a estrutura mista. Neste tipo de interligação várias Autoridades Certificadores se autenticam mutuamente, criando várias relações de confiança. Neste tipo de estrutura não existe a hierarquia nas relações de confiança entre as autoridades certificadoras. O principio de funcionamento é análogo ao anterior.

X.509 O X.509 faz parte de um conjunto de recomendações da IEFT para certificados de chave pública. A recomendação X.509 vem ao longo dos anos sofrendo uma modificações encontrando-se atualmente na sua versão 3. Um certificado X.509 é composto de informações assinadas pelo seu emissor sobre um determinado sujeito que são armazenadas em campos obrigatórios e opcionais. Para validar a autenticidade deste certificado verificamos a assinatura do emissor e, caso a assinatura seja considera verdadeira, as informações contidas no certificados serão consideradas confiáveis.

A Figura 2.10 representa um certificado de identidade X.509 na sua versão 3. Ele é composto de campos opcionais e obrigatórios como descritos a seguir.

Versão

Indica a versão do X.509 (atualmente 3). **Número de Série**

O número de série é um identificador único atribuído pelo emissor.

Algoritmo e Parâmetros

Contém o identificador do Algoritmo usado para assinar o certificado e seus parâmetros

Emissor

O Emissor é o nome único usado para identificar o certificado x.509.

Não antes de e Não depois de

Indicam o período de validade do presente certificado.

Sujeito

No Sujeito contém o identificador do Sujeito objeto desse certificado que possui a chave pública representada neste certificado.

Algoritmos, Parâmetros e Chave pública do Sujeito

Estes campos contém as informações relativas a chave pública do sujeito. Estes campos indicam o algoritmo e os parametros utilizados para gerar a chave pública.

Identificadores únicos do Emissor e do Sujeito

Esses campos são usados caso seja necessário reutilizar os nomes dos nomes do

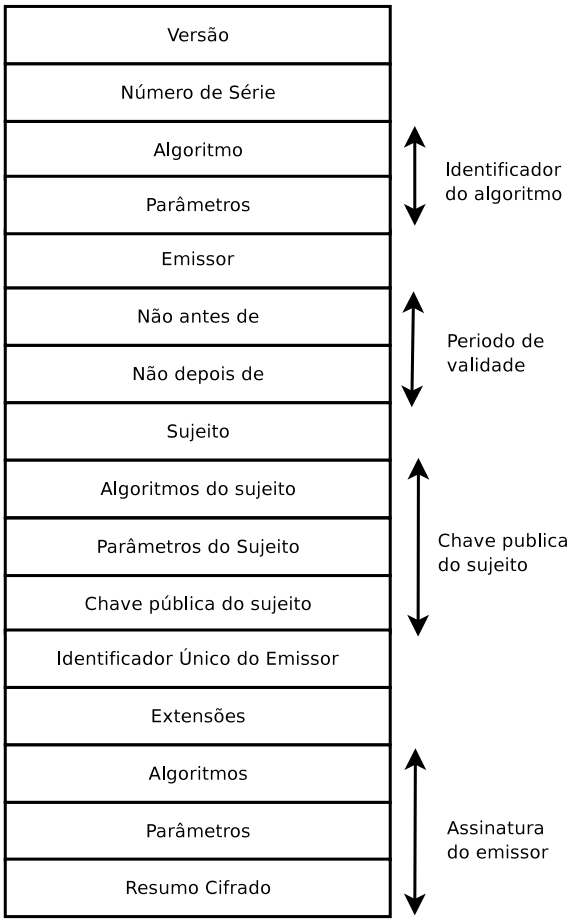

Figura 2.10. Campos que representam um Certificado de identidade X.509.

Sujeito e do Emissor.

Extensões

Esse campo é utilizado para permitir extensões do certificado. Através desse campo é permitido ao emissor implementar funcionalidades adicionais no certificado de acordo com as suas necessidades. Este campo poderia, por exemplo, ser usado para fornecer informações de autorização ao Sujeito do certificado.

Algoritmos, Parâmetros e Resumo Cifrado

Os campos Algoritmos, Parâmetros e Resumo Cifrado são usados para indicar quais algoritmos e parâmetros foram usados para assinar este certificado.

Como foi descrito anteriormente, o campo de extensão pode ser usado para permitir que informações de autorização a um certificado de identidade. Existe pelo menos dois bons motivos para isto não ser feito desta forma. O primeiro motivo é que o certificado de identidade possui um tempo de vida maior que o de autorização. Uma autorização pode deixar de existir e nem por isso um sujeito de um certificado de identidade deveria ter o seu certificado de identificação revogado.

Além do certificado de identidade, a recomendação X.509 permite a definição de

certificados de autorização. Os certificados de autorização contém atributos que especificam privilégios de um grupo, usuário, papel (role) ou outra informação de autorização associada ao proprietário desse certificado [Xavier, 2004].

2.3.2.4. Redes de Confiança (SPKI/SDSI)

O desenvolvimento do SDSI/SPKI foi motivado pela alta complexidade das infraestruturas de chave públicas, em especial o X.509. O SDSI foi projetado no MIT por Ronald Rivest e Butler Lampson [Rivest and Lampson, 1996]. O SDSI é uma infraestrutura de chaves públicas com espaço de nomes locais, o que dá a característica de descentralização. O SPKI foi desenvolvido por Carl Ellison e outros [Ellison et al., 1999] e é um sistema de autorização flexível e simples. Os dois projetos se uniram e formaram o SPKI/SDSI um sistema de autenticação e autorização que combina o espaço de nomes locais do SPKI e o sistema de autorização do SPKI.

O SPKI/SDSI usa como forma de representação as *S-expressions*. Uma s-expression é uma convenção para representar estruturas de dados ou expressões no formato de texto, como usado nas estruturas LISP. As *S-expressions* encapsulam elementos com parênteses cujos elementos podem ser cadeias de caracteres ou outra *S-expression*.

Uma chave pública representada sobre a forma de *S-expression* pode visto na Figura 2.5 a seguir. A primeira linha é o identifica que a *S-expression* é de uma chave pública. A segunda linha representa os algorimos de criptografia usados dentro dessa chave pública: algoritmo de chave pública RSA, padrão de criptografia PKCS1 e algoritmo de *hashing* MD5. Finalmente, o valor (e #11#) indica o valor do expoente da chave RSA. Os valores "n| ANeWQ0..."representa a chave criptográfica.

```
(public key
        (rsa pkcs1 md5 (e #011#)
                (n |ANeWQ0+7nhwMzuahgLPPMbOi6jUP0RPgZTzpLAhJ6qm/1DT1LVRYF78izo5z
                JqtTCB/yYoSExEEM2e8Anx7trz+wK6U4HdBcEwexjkXXo3BM9D433bpVm8iM61y
                8FMaYH743gvectGZ3BBBGNzH6KHAERXjW0te2y9UpT1GzWart|FMaYH743gvect
                GZAA|)
        )
)
```

Figura 2.11. Chave Pública SPKI/SDSI.

Nomes no SPKI/SDSI

No SDSI/SPKI a identificação é feita através de chaves públicas e não por um nome. O argumento para tal consideração é que o nome não é um identificador único (o caso dos homônimos) e em alguns casos podem também mudar. É bem comum as pessoas mudarem seus nomes, seja porque não gostam ou porque contraem matrimônio. Uma

chave pública, por outro lado, não pode ser considerada única, porem é pouco provável a coincidência da escolha aleatória de uma chave pública. Um tamanho de chave pública comumente usado possui cerca de 309 caracteres (1024 bits).

O SDSI associa uma chave pública a um nome no espaço de nomes local do usuário. Um nome pode estar associado a zero ou mais identificadores e somente serão conhecidos localmente, ou seja, não necessitam ser globalmente único. Um nome básico no SDSI é uma *S-expresssion* com dois elementos: a palavra *name* e o nome escolhido para identificá-lo. Por exemplo joão poderia identificar maria dentro do seu espaço de nome da seguinte forma: joão: (*name* maria).

O nome tem significado somente para quem o definiu, porém para ser identificado globalmente deve ser associado a uma chave pública. Assim se a chave pública definida na Figura 2.11 fosse usada para identificar maria, a *S-expression* que representa o nome completo SDSI visto na Figura 2.12.

```
(name
(public key
        (rsa pkcs1 md5 (e #011#)
                (n |ANeWQ0+7nhwMzuahgLPPMbOi6jUP0RPgZTzpLAhJ6qm/1DT1LVRYF78izo5z
                JqtTCB/yYoSExEEM2e8Anx7trz+wK6U4HdBcEwexjkXXo3BM9D433bpVm8iM61y
                8FMaYH743gvectGZ3BBBGNzH6KHAERXjW0te2y9UpT1GzWart|FMaYH743gvect
                GZAA|)
        )
)
maria)
```

Figura 2.12. Nome completo SPKI/SDSI.

Certificados SPKI/SDSI

Como uma solução totalmente distribuída o SDSI/SPKI permite uma grande flexibilidade nas definições de certificados e delegação. Primeiramente, o SPKI/SDSI é sistema igualitário, ou seja, não existe hierarquia entres os participantes. Cada usuário é responsável por gerenciar seus próprios certificados, ou seja, é uma autoridade certificadora. Assim eles possuem liberdade para gerar certificados e delegar acessos aos seus recursos.

Existem dois tipos de certificados no SPKI/SDSI: o Certificado de Nome (*Name Certs*) e o Certificado de Autorização. *Auth Certs*. O Certificado de Nome providencia autenticidade de um nome local, ou seja, ele certifica que o nome criado dentro do espaço de nomes do emissor é válido. O certificado de autorização concede uma autorização de acesso a um recurso pertencente ao emissor ao sujeito do certificado.

Um certificado de nome é composto por quatro campos: *issuer*, *identifier*, *subject* e *validity specification* [Clarke et al., 1999]. O *issuer* é a chave pública que assina o certificado. O *identifier* identifica o nome local que se está definindo. O *subject* é representado por uma chave pública ou um nome (chave seguida de um ou mais iden-

tificadores). Caso o *subject* não seja iniciado por uma chave pública é considerado que o nome pode ser encontrado dentro do espaço de nomes local. O *validity specification* corresponde ao período de validade do certificado é considerado válido; o *validity specification* também pode ser usado para verificação em lista de revogação de certificado. O Certificado de Nome em *S-Expression* tem a seguinte forma: (cert (issuer (name K N)) (subject S) <valid>). Onde: K é o *issuer*, N é o *identifier*, S é o *Subject* e <valid> é o *validity specification*. O exemplo a seguir cria um certificado de nome para maria no espaço de nome de joão.

O SPKI permite a definição de grupos. Cada grupo possui um nome e um conjunto de membros. Um grupo pode referenciar também outros grupos. Para criar um grupo, o dono do grupo emite, para cada membro do grupo, um certificado definindo o nome local do grupo. A possibilidade de criação de grupo facilita o controle de acesso aos recursos, com esta estrutura pode-se autorizar (ou desautorizar) todas as pessoas do grupo apenas fazendo isto para o grupo.

Um Certificado de Nome consiste de cinco campos: *issuer*, *subject*, *delegation* , *tag* e *validity specification*. Os dois primeiros tem função análoga ao Certificado de Nome explanado anteriormente, sendo que o Subject pode também indicar um grupo.. O campo *delegation* indica se o certificado pode ser delegado ou não. O *tag* especifica que tipo de autorização (ou autorizações) o sujeito do certificado receberá. O *validity specification* tem função análoga ao Certificado de Nome.

Através da indicação do campo *delegation* um Certificado de Autorização quando criado pode permitir que o sujeito do certificado possa delegar seus direitos. A Figura 2.13 mostra um exemplo de delegação que poderia ocorrer num ambiente com SPKI/SDSI. A impressora I (ou um sistema de gerenciamento que a represente) emite um certificado com delegação para o gerente G permitindo o direito de imprimir - I(I,D)G. O gerente, por sua vez, delega este direito a um funcionário F que confia, gerando um novo certificado para o funcionário - G(I,D) e entrega o certificado que gerou junto com o certificado recebido da impressora. Um estagiário não muito confiável solicita o direito de imprimir para a funcionária, esta o faz mas com a ressalva de não propagar o direito de delegação. O estagiário para imprimir entrega toda cadeia de confiança que foi gerada no processo e sua assinatura. A impressora confirma todas as assinaturas e garante o direito de impressão para o estagiário.

A próxima subseção apresentará estudos de caso de implementações de grades computacionais representando o estado da arte na área.

2.4. Estudo de casos

Nesta seção apresentamos estudos de caso de algumas grades computacionais existentes. Uma breve descrição de cada arquitetura é mostrada. A seguir, decrevemos a implementação de segurança de cada uma delas.

2.4.1. Globus

O Globus [Foster and Kesselman, 1997, Globus, 2004] é atualmente o projeto de maior impacto na área de Computação em Grade. O Projeto Globus envolve muitas ins-

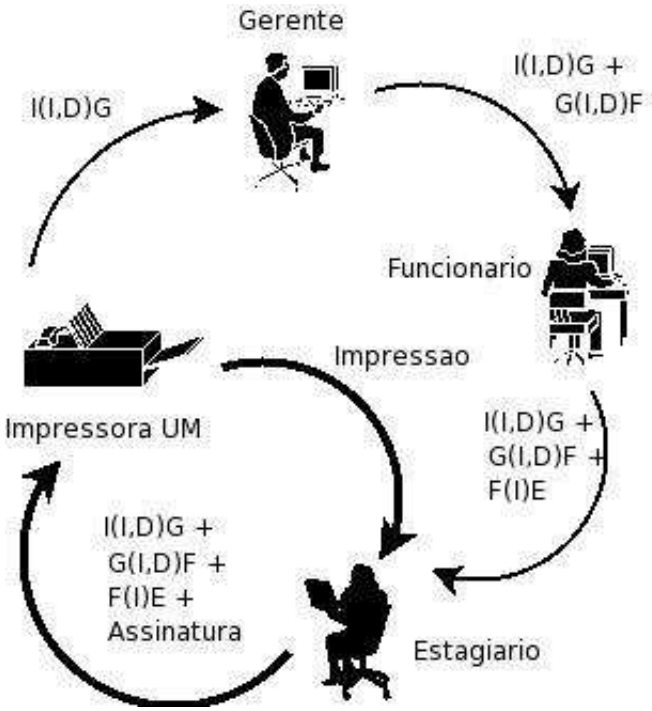

Figura 2.13. Delegação de um Certificado.

tituições de pesquisa da Europa, da Ásia e pricipalamente dos Estados Unidos, dentre elas podemos citar: Laboratório Nacional de Argonne (ANL), Universidade de Chicago, Universidade do Sul da Califórnia (USC) e o Laboratório de Computação de Alto Desempenho da Universidade do Nordeste de Illinois e o Imperial College na Inglaterra. Grandes intituições da indústria também apoiam o projeto tal como: IBM, Microsoft e Cisco.

O projeto Globus inclui o desenvolvimento de um sistema de Computação em Grade denominado Globus Toolkit. O *Globus Toolkit* (GT), é um pacote que permite a construção de sistemas de grade de modo incremental. O termo *toolkit* é utilizado pelo fato dele permitir a configuração e personalização de grades especializadas e aplicações. O Globus Toolkit está atualmente na versão 4.0, que apresentamos nesta subseção.

A versão atual do Globus Toolkit é uma implementação do OGSA (*Open Grid Services Architecture*), uma arquitetura padrão que visa a construção de uma infra-estrutura para grades baseada em serviços [Foster et al., 2002]. Serviços são entidades que disponibilizam suas funcionalidades a usuários e aplicações através da rede. No modelo OGSA, qualquer objeto é representado por um serviço, tais como: dispositivos de armazenamento, redes, programas, entre outras entidades.

A base da definição de serviços para o OGSA é denominada *Grid Service*. O *Grid Service* é uma extensão de um *Web Service* que mantém informações do estado dos serviços. A definição de estados permite a distinção entre a instância de um serviço de outras que providenciam uma mesma interface.

As interfaces básicas e o comportamento de um *Grid Service* é definido através

do WSRF (*Web Service Resource Framework*) [Foster and Czajkowski, 2005]. O WSRF é uma evolução do OGSI (*Open Grid Service Infrasctructure*) que é um padrão utilizado nas versões anteriores do GT. O WSRF separa as funcionalidades do OGSI em diversas especificações e inclui algumas novas especificações que surgiram. O WSRF define uma família de especificações para acessar recursos usando serviços Web.

A seguir apresentamos o GSI, uma implementação de arquitetura de segurança baseada no OGSA.

2.4.1.1. GSI - Grid Security Infrastructure

O Globus disponibiliza um serviço de segurança denominado GSI (*Globus Security Infrastructure*) [Foster et al., 1998], uma implementação da arquitetura de segurança baseada no OGSA. Assim como outros serviços do Globus, o GSI é uma especificação abstrata que pode ser implementada sobre diferentes mecanismos de segurança.

A Figura 2.14 mostra o resumo da arquitetura do GSI [Ian, 2005]. O GSI utiliza diversos padrões para implementar funcionalidades como autenticação, delegação proteção de mensagens e autorização.

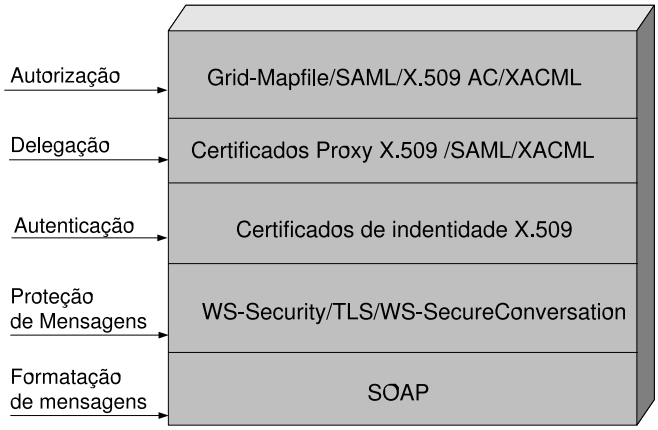

Figura 2.14. Arquitetura do GSI.

A autenticação do GSI é feita através de credenciais X.509. O X.509 é utilizado no GSI para permitir o estabelecimento de confiança entre domínios de segurança diferentes. Além da autenticação X.509, presente também nas versões anteriores do Globus Toolkit, a versão 4 permite a autenticação de usuários utilizando identificador de usuário e senha. Porém quando este método é usado perde-se a possibilidade de usar mecanismos de segurança que incluem confidenciabilidade e integridade de dados.

Para permitir a delegação dinâmica o GSI estende o conceito de *proxy* do X.509 que permite ao usuário atribuir uma nova identidade X.509 ao usuário e então delegar alguns de seus atributos de segurança a esta nova identidade [Welch et al., 2004]. Este mecanismo permite a criação de novas identidades e credenciais sem a intervenção do administrador da rede. A vantagem de se estender o padrão X.509 é a possibilidade de

utilização de bibliotecas já existentes com uma modificação bem pequena.

A tabela 2.1 resume as diferenças entre o certificado X.509 e o certificado x.509 com proxy[Welch et al., 2004]. No campo *Issuer* o Certificado Proxy é identificado pelo chave pública do certificado ou por outro Certificado X.509 Proxy. Esta opção permite que o certificado seja criado dinamicamente por possuidor de certificado, sem a intervenção das CAs. Ao contrário do padrão X.509, onde o sujeito de um certificado é definido pela CA, o campo *Subject Name* do certificado permite a definição de sujeitos dentro do escopo de nomes do emissor do certificado. A restrição do escopo deve ser feita para que os certificados sejam únicos. A chave pública do Certificado X.509 Proxy é distinta da chave pública do seu emissor.

Tabela 2.1. Comparação entre X.509 padrão e X.509 proxy estendido pelo Globus Toolkit.

Atributo do Certificado	**X.509**	**X.509 Proxy**
Issuer/Signer	Autoridade Certificadora	Certificado de chave pública ou Certificado de outro Proxy
Subject Name	Definido pela Autoridade certificadora	Definido no espaço de nomes do dono do certificado
Delegação do Emissor	——	Define as condições dos direitos delegados
Key pairs	Par de Chave única	Par de chave única

O Certificado de Proxy acrescenta duas características adicionais ao Certificado X.509 padrão: autenticação única e delegação. A autenticação única dispensa a digitação freqüente de uma senha para provar a autenticidade de um usuário. A delegação, como já definido anteriormente, permite que o servidor se conecte a qualquer recurso em nome do cliente.

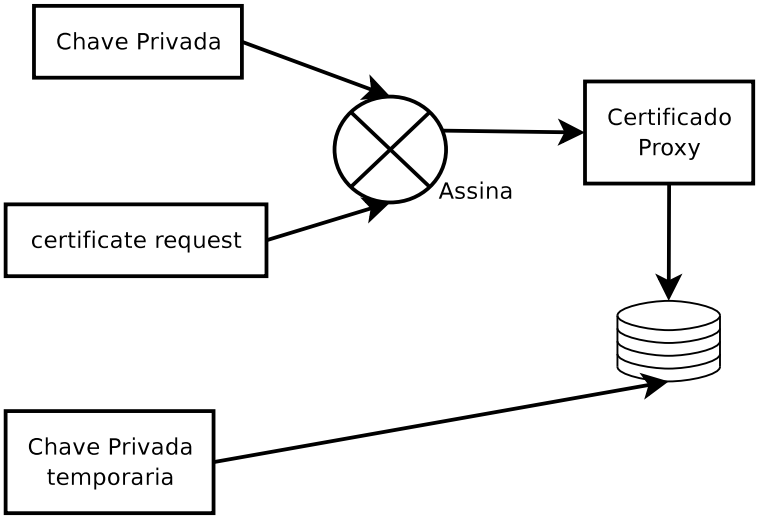

Figura 2.15. Protocolo de Autenticação Única.

A autenticação única permite o conceito de autenticação na Grade para reduzir o número de vezes em que o usuário precisa acessar sua chave privada [4]. Com o Certificado

[4]O acesso a chave privada é uma ação de risco por isso ela é geralmente protegida por senha, em um ambiente de grade isso pode não ser viável.

Proxy, o usuário autentica uma vez para criar o Certificado Proxy (que possui um par de chaves diferente do seu criador) e o usa inúmeras vezes sem precisa usar a chave privada do seu criador. Para aumentar a segurança o Certificado Proxy possui um tempo de vida curto, e o compromentimento de sua chave não chega ser um problema grave de segurança.

A autenticação única no Globus Toolkit é mostrada na Figura 2.15. Inicialmente um novo par de chave pública e privada é criado e associado a um *certificate request*[5]. A seguir, o usuário usa sua chave privada para assinar o Certificado Proxy recém criado. Finalmente, o Certificado Proxy é armazenado de forma protegida em disco.

A delegação de direitos no Globus Toolkit também pode ser feita sobre um canal de comunicação [Welch et al., 2004] como pode ser visto na Figura 2.16. Inicialmente, os dois envolvidos na delegação negociam um canal seguro de comunicação. O delegado, aquele que recebe a delegação, cria um par de chaves e um *certificate request*. O delegado então envia o *request certificate* sobre o canal seguro. Ao receber o *request certificate*, o delegador define as restrições nos direitos doados e usa a chave privada associada ao seu Certificado Proxy para gerar o novo Certificado Proxy. O delegador envia o Certificado sobre o canal seguro de rede. Ao receber o certificado, o delegado coloca em um lugar seguro, junto a chave privada criada no inicio do processo.

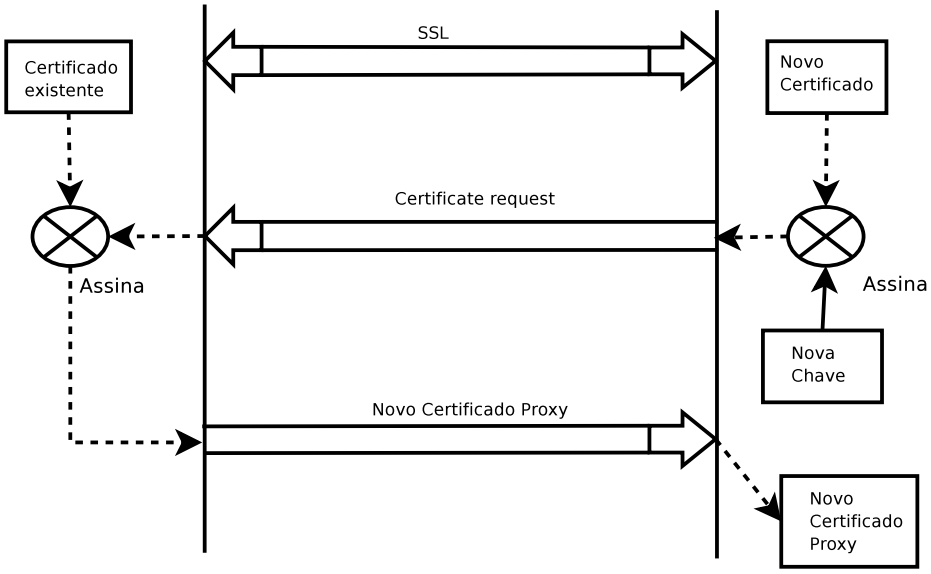

Figura 2.16. Delegação de Certificado Proxy

Para permitir a comunicação entre as federações, o GSI usa *gateways* que fazem tradução entre credenciais X.509 e outros mecanismos. O KCA (ou KX.509) (`http://www.citi.umich.edu/projects/kerb_pki`), desenvolvido na **University of Michigan** é um sistema baseado no Kerberos que permite que usuários adiquiram certificados X.509 usando *tickets* Kerberos. O usuário se autentica normalmente no Kerberos e solicita uma credencial para acessar o serviço do KCA. O KCA recebe o *ticket*, determina

[5]Especificação padrão para codificar requisições de certificados, incluindo o nome da pessoa que requisita o certificado e sua chave pública.

a identidade do usuário, gera um certificado de tempo curto (*short-lived* X.509) e envia de volta ao usuário que solicitou.

O GSI implementa troca segura de mensagens na rede de duas formas: no níveis do transporte e da mensagem. A comunicação no nível do transporte usa o TLS (Transport Layer Security)[Dierks and Allen, 1999] para a criação de comunicação segura fim a fim. A comunicação no nível da mensagem usa padrões de segurança baseado em *Web Services* [Booth et al., 2004] e SOAP (Simple Object Access Protocol) [W3C, 2000].

O TLS é baseado no protocolo *Secure Socket Layer* (SSL) desenvolvido pela empresa Netscape Corporation. O TLS é um protolo de transporte seguro que provê privacidade e integridade e é usado para encapsular mensagens em protocolos de alto nível. O TLS permite que as partes envolvidas realizem autenticação mútua e negociação de algoritmos de criptografia, criando um canal seguro de comunicação. O TLS foi projetado de forma a ser um protocolo eficiente quanto ao uso de CPU e o uso da rede, o que faz com que ele seja uma boa opção de uso em grades computacionais.

O *Globus Toolkit* implementa as especificações *WS-Security* e *WS-SecureConversation* para permitir comunicação segura no nível de mensagem. O padrão *WS-Security* é um arcabouço que permite a criptografia e assinatura de mensagens sobre SOAP. Ele especifica um mecanismo que associa um *token* de segurança às mensagens. Para o padrão *WS-Security*, o token é uma estrutura de dados extensível, codificado em binário, usado para representar de forma transparente os vários mecanismos de segurança. A integridade das mensagens é garantida através de *XML Signature* e a confidenciabilidade usando *XML Encription* [Foster and Kesselman, 2003]. O *WS-SecureConversation* gerencia a criação de contextos de segurança e cria chaves que podem ser usadas neste contexto. Este contexto pode então ser usado para proteger mensagens subseqüentes sem que o processo de autenticação seja repetido a cada momento.

A autorização no *Globus Toolkit* é feita através do uso de SAML (*Security Assertions Markup Language*) e de Grid mapfiles. O SAML define a sintaxe e semântica de afirmações feitas sobre um sujeito por uma entidade. O GSI usa essa linguagem para permitir a comunicação com autoridades certificadoras e receber as informações a respeito das autorizações que um cliente tem sobre um determinado serviço. O uso do Grid-mapfile é mantido por compatibilidade a versão 2 do GT e associa um determinado usuário a nomes qualificados que aparecem em certificados X.509. Por exemplo, a entrada "/O=Grid /OU=Ime /OU=simpleCA-ime.usp.br /OU=localdomain /CN=Jose Braga" jrbraga associaria o usuário jrbraga ao correspondente identificador X.509.

O Globus Toolkit introduz o servidor CAS (*Comunity Authorization Server*) que é responsável pelo gerenciamento de políticas de acesso para os recursos da comunidade [Pearlman et al., 2002]. Dependendo da política da comunidade, o CAS pode ser controlado por um ou mais administradores. O CAS permite que as tarefas de administração possam ser distribuídas. Um administrador pode, por exemplo, ser responsável pelo registro de usuários e recursos em um servidor CAS, mas não ter o direito de associá-los a grupos que possuem direitos específicos, podendo essa autorização ser feita por um outro administrador.

A Figura 2.17 representa um usuário solicitando ao servidor CAS uma requisição

de capacidade (*capability*). Essa capacidade permite a ele executar um conjunto de ações específicas. O servidor CAS, caso a política de acesso definida para esse usuário permita, devolve ao usuário a capacidade solicitada. De posse dessa autorização, o usuário pode usar a capacidade de acordo com os direitos permitidos ou delegá-la a um terceiro.

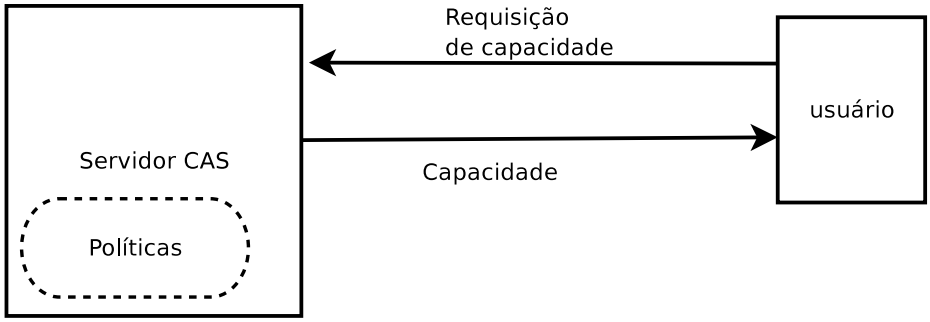

Figura 2.17. Usuário faz uma requisição de capacidade ao servidor CAS.

A Figura 2.18 mostra um usuário submetendo uma capacidade a um provedor do recurso. O usuário faz uma requisição ao dono do recurso provando que possui direitos de acesso. A seguir, o provedor verifica se a capacidade apresentada pelo usuário e se as políticas definidas localmente permitem o uso do recurso. Caso isso seja possível, o usuário finalmente recebe o direito de utilizar o recurso.

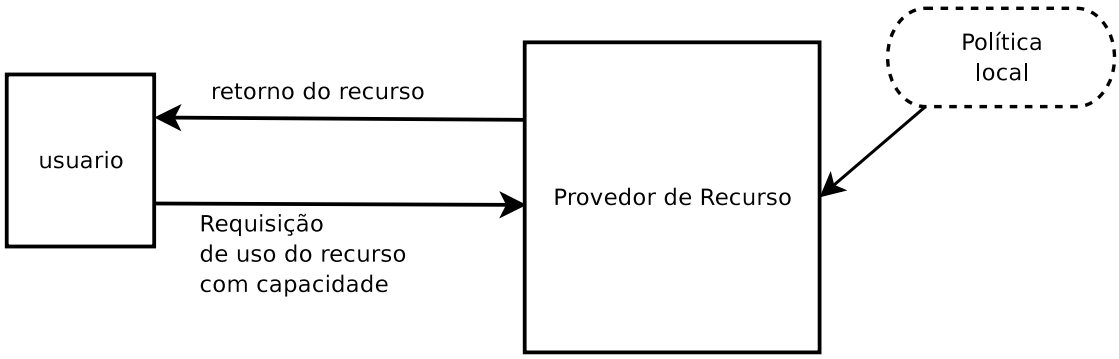

Figura 2.18. Cliente requisitando uso de recurso.

2.4.2. Legion/Avaki

O Projeto Legion [Lewis and Grimshaw, 1996] foi desenvolvido na Universidade da Virginia com o objetivo de prover abstrações que permitissem uma visão única para milhares de computadores interligados em um sistema distribuído. Desde o início do projeto, o Legion foi criado com o intuito de permitir a transferência de sua tecnologia para a indústria; isto ocorreu em 2001 através da empresa AVAKI fundada pelos seus idealizadores. A companhia detém os direitos sobre o Legion que passou a ser chamado de Avaki.

No Legion, todas as entidades da Grade são representadas por objetos (no sentido de Orientação a Objetos), tais como: computadores, usuários, equipamentos, dispositivos de armazenamento e aplicações. Estes objetos comunicam-se entre si através de chamadas

a métodos assíncronos. Cada método possui uma assinatura própria que descreve os seus parâmetros e valores devolvidos. Um objeto é descrito por um conjunto de interfaces que define suas classes. O Legion utiliza uma IDL (*Interface Definition Language*) para descrever seus objetos e é um sistema reflexivo onde as próprias classes são objetos.

Os objetos Legion são persistentes e podem estar associados a dois estados: ativo ou inativo. Os objetos ativos ficam presentes na memória, possuem um processo associado e podem responder a chamadas a seus métodos. Os objeto inativos são armazenados em algum dispositivo de armazenamento através de sua representação seriada (OPR - *Object Persistent Representation*). Cada OPR é endereçado por um endereço de representação persistente (OPA - *Object Persistent Address*).

Cada objeto Legion possui um identificador LOID (*Logical Object Identifier*) e um endereço de objeto (LOA - *Legion Object Address*). O LOID é um endereço único, atribuído pela classe do objeto no momento de sua criação. O LOA provê um ou mais protocolos de rede utilizados para a comunicação concreta com um objeto. Por exemplo, o LOA poderia utilizar o protocolo TCP indicando o endereço IP e a porta onde o objeto está localizado.

O Legion define as interfaces e funcionalidade para um conjunto de classes bases denominadas objetos do núcleo. Essas classes básicas oferecem serviços básicos para os objetos que as herdam. Os objetos do núcleo são:

- *Legion Object e Legion Class* - definem um conjunto de métodos que todos os objetos e classes devem implementar. O *Legion Object* define o método $may_I()$ usado para controle de segurança das chamadas de execução.
- *Legion Hosts* - Quando um objeto *Legion Host* é ativado ele cria um processo para contê-lo. Cada recurso computacional pode estar associado a mais de um objeto *Host*. O *Legion Hosts* tem como responsabilidade criar e gerenciar processos para objetos Legion ativos.
- *Legion Vaults* - representam o armazenamento persistente dos demais objetos Legion. Quando um objeto é seriado, sua representação (OPR) é armazenada em disco por objetos *Legion Vaults*.
- *Implementation Objects* - encapsulam aplicações a serem executadas no Legion. O Legion utiliza metadados para apresentar as características importantes de uma aplicação, tais como: formato da aplicação (por exemplo, binário ou byte-code), plataforma e requisitos de hardware.

2.4.2.1. Segurança

O identificador de objetos LOID possui um campo que armazena uma chave pública, que vai ser a base para a autenticação e da autorização do Legion. Através do LOID é possível se comunicar de forma segura com um objeto obtendo a chave criptográfica armazenada. Como a chave pública no Legion é associada ao nome do objeto, um atacante não pode subistituir uma nova chave em um identificador de objeto conhecido, por que

se uma parte do LOID é alterada, incluindo a chave, um novo LOID é criado e não será conhecido.

O Legion usa o paradigma de usuário e senha para permitir a autenticação do usuário. Para se autenticar, o usuário fornece um identificador de conta (*login*) e uma senha. O nome do usuário é associado ao que o Legion chama de objeto de autenticação que vai agir como o procurador do usuário na Grade. O objeto de autenticação possui a chave privada, a senha criptografada e o perfil do usuário no seu estado persistente. A senha oferecida pelo usuário é comparada à senha armazenada no estado persistente do procurador para negar ou permitir o acesso à Grade. O estado de autenticação do objeto é armazenado de forma persistente no disco na máquina.

Uma outra forma de autenticação no Legion é através do LDA (*Legion Data Access*). O LDA cria um mapeamento entre o identificador do usuário no Unix e o LOID de um objeto de autenticação para acessar a dados da grade usando o protocolo NFS. Quando um cliente NFS monta um DAP ele mapeia o espaço de nomes no sistema de arquivo local do seu host, providenciando um acesso aos dados da Grade de forma transparente. O DAP fornece suporte ao mecanismos de segurança do Legion, o controle de acesso é feito através de credenciais assinadas e as interações com a Grade pode ser assinada.

O Legion usa credenciais para permitir a delegação de direitos. O Legion define uma forma de transferência de direitos através de credenciais para objetos. O credencial é uma lista de direitos assinada pelo usuário ou pelo seu procurador. O recurso verifica a credencial e segue a cadeia de confiança até conferir a assinatura, decidindo se garantirá o direito ao recurso.

A autorização do usuário no Legion é feita através de lista de controle de acesso (LCA). Para cada função ou ação associada a um objeto existe uma lista de permitidos e negados (*allow* e *deny*). Cada lista contém o nome de outros objetos ou um diretório que contém uma lista de objetos. Caso um diretório contenha objetos de autenticação (um objeto de autenticação é o representante do usuário na Grade) então ele atua como grupo de usuários. Se um objeto estiver presente simultaneamente nas duas listas, o objeto será negado de efetivar a ação.

O protocolo SSL é usado para permitir a comunicação segura no Legion. Os dados são protegidos no Legion de três formas: privada, protegida e nenhuma. Na primeira, mais restritiva, todas as mensagens são completamente criptografadas, garantindo confidencialidade e autenticidade. No modo protegido, as mensagens são protegidas de modificação através de funções de resumo (*hashing*). E finalmente, o Legion permite que as mensagens possam ser transmitidas sem criptografia, com exceção das credenciais.

O Legion providencia um mecanismo de *sandboxing* para a proteção das máquinas hospedeiras. O *sandbox* faz um isolamento do identificador real do usuário e do sistema de Grade na maquina hospederia. O *sandboxing* do Legion permite a definição de identificadores genéricos, portanto que não estão associados a nenhuma conta na máquina.

2.4.3. Condor

O sistema Condor[6] foi criado no início dos anos 80 na Universidade de Wisconsin como um sistema para computação distribuída caracterizado por deixar o controle das máquinas sob a responsabilidade dos seus proprietários. O Condor (Condor high-througput computing system) é um sistema de computação batch distribuído para computação intensiva, com mecanismos para gerenciamento de tarefas, políticas de escalonamento, gerenciamento e monitoramento de recursos, etc.

A disponibilidade de grande montante de recursos tolerante a falhas por prolongados períodos de tempo (high throughput) associada a computação oportunista (aproveitamento de recursos ociosos) são características fundamentais do Condor. Essas características são implementadas através dos mecanismos de *ClassAds*, migração e *checkpoint* de tarefas e chamadas de sistema remotas.

A linguagem ClassAd provê meios eficientes para comparar os recursos oferecidos com as solicitações de recursos recebidas, ou seja. na submissão de tarefas. O Checkpointing periódico de tarefas provê tolerância à falhas e permite migração de tarefas entre máquinas para escalonamento preemptivo de baixa custo [Krueger, 1988]. Condor implementa chamadas de sistema remotas, um mecanismo de sandbox móvel, para redirecionar, quando executando tarefas em máquinas remotas, chamadas de callback relacionadas a tarefas de I/O para a máquina que submeteu a tarefa. Com esse mecanismo os usuários não precisam disponibilizar arquivos de dados em estações de trabalho remotas antes da execução do Condor nessas máquinas.

Condor-G [Frey et al., 2002] é um encontro das tecnologias Condor e Globus projects. Protocolos para comunicação inter-domínio e acesso padronizado a diversos sistemas batch remotos são contribuições do Globus. O Condor-G incorpora do Condor os conceitos de submissão e alocação de tarefas, recuperação de erros e criação de um ambiente de execução amigável. No relacionamento com o Globus Toolkit, o Condor-G pode ser usado como um serviço confiável para gerenciamento e submissão de tarefas para um ou mais sites enquanto o sistema Condor high-throughput pode ser usado como um serviço de gerenciamento da estrutura subjacente para um ou mais sites, com o Globus Toolkit como elo entre eles.

2.4.3.1. Segurança

O Condor possui uma biblioteca chamada CEDAR que permite a clientes e servidores negociar e usar diferentes protocolos de autenticação tais como Kerberos, GSI, chaves públicas, autenticação via redes ou nós confiáveis e sistema de arquivos. Essa biblioteca implementa um conjunto de interfaces de autenticação simples e extensível. A biblioteca CEDAR possui a capacidade de negociar integridade de dados e algoritmos de privacidade separado do protocolo de autenticação. A Figura 2.19 mostra a Arquitetura de Segurança do Condor.

A autenticação no Condor baseada em sistema de arquivos é implementada da seguinte forma. Um processo *daemon* utiliza a propriedade do arquivo para decidir a

[6] `http://www.cs.wisc.edu/condor/`

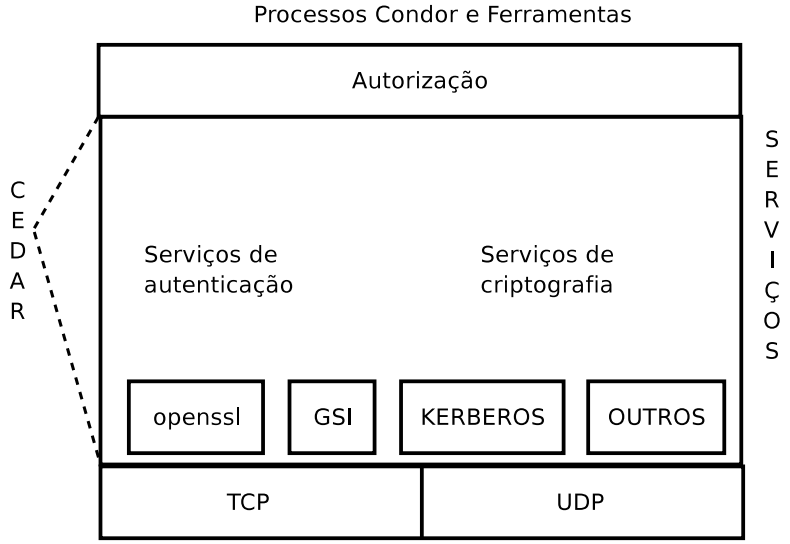

Figura 2.19. Arquitetura de Segurança do Condor.

identidade. Quando um usuário deseja entrar na grade, o processo solicita a escrita de um arquivo em um diretório de escrita temporária. Comparando o dono do arquivo e o nome que o usuário indicou, ele decide sobre a autorização.

A execução de uma tarefa no Condor é necessariamente cooperativa. Certas informações são conhecidas unicamente pela máquina de execução, tais como quais sistemas de arquivos, redes e bases de dados podem ser acessados, enquanto somente a máquina de submissão conhece em tempo de execução os recursos necessários para a tarefa. Essa cooperação conhecida como *split execution* é realizada no Condor pelos componentes shadow e sandbox.

Condor provê o mecanismo chamado *shadow*, para proteção dos usuários, no site de submissão. O shadow providencia todo o necessário para especificação da tarefa em tempo de execução: o executável, os argumentos, o ambiente, arquivos de entrada, etc. Nenhum deles conhecido fora do agente de execução até o momento real da execução.

O *sandbox* no site de execução deve proteger proprietários de máquinas, impedindo acesso indevido aos recursos. O sandbox é formado por dois componentes distintos: sand, responsável por criar o ambiente apropriado à execução da tarefa e o componente box, responsável pela proteção dos recursos de possíveis danos que uma tarefa maliciosa possa causar.

Em versões iniciais, Condor restringia a execução de tarefas a uma parte limitada do sistema de arquivos através do comando *chroot* do Unix. Atualmente, as tarefas são executadas sem restrições no sistema de arquivos, mas com restrições no login. Essa abordagem possibilita contudo, em uma máquina multi-cpu executando múltiplas tarefas simultaneamente, o seqüestro de uma tarefa de usuário por outro usuário malicioso, porque compartilham um mesmo ID de usuário, ainda que de uma conta nobody padrão com poucos privilégios. Condor evita esse problema alocando dinamicamente IDs de usuário para cada tarefa em execução. No Unix, isso exige intervençaõ do administrador,

enquanto em ambiente Windows, Condor aloca usuários em tempo de execução. Condor também usa o conceito de domínio de ID de usuário para um conjunto de máquinas compartilhando uma mesma base de dados de usuário.

2.4.4. InteGrade

O InteGrade [7] é um sistema de Grade motivado principalmente pela necessidade de aproveitamento de recursos computacionais compartilhados (computadores pessoais, estações de trabalho), em geral subutilizados – com predomínio de períodos ociosos – para execução de aplicações com grande demanda por tais recursos. As características do InteGrade incluem suporte para uma gama de aplicações paralelas e mecanismos que buscam minimizar a percepção, pelos proprietários dos recursos compartilhados, de qualquer possível perda de qualidade de serviço [Goldchleger, 2004].

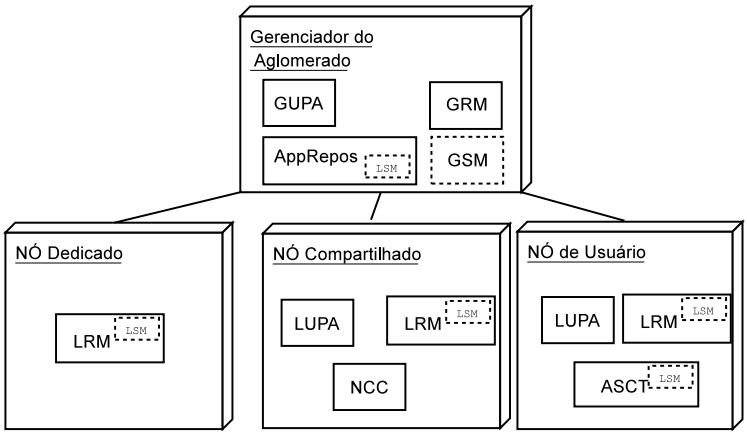

Figura 2.20. Arquitetura do InteGrade com os módulos de segurança.

Uma grade do InteGrade é constituída de aglomerados (*clusters*) de computadores organizados de forma hierárquica (Figura 2.20). Nesta seção apresentamos uma visão resumida da sua arquitetura através da descrição dos componentes de um aglomerado e dos módulos implantados nestes componentes.

- Gerenciador do aglomerado: responsável por gerenciar o aglomerado e pela comunicação com gerenciadores de outros aglomerados. Este nó pode ser distribuído para balanceamento de carga ou replicado para tolerancia a falhas.
- Nó dedicado: reservado para as aplicações da grade.
- Nó compartilhado (fornecedor de recursos): tipicamente uma estação de trabalho, disponibiliza parte de seus recursos para execução de aplicações dos usuários da grade.
- Nó de usuário: pertence ao usuário da grade que submete aplicações à grade.

O *Local Resource Manager* (LRM) e o *Global Resource Manager* (GRM) cooperam no gerenciamento dos recursos de um aglomerado. O LRM é executado nos nós do

[7] `http://gsd.ime.usp.br/inteGrade`

aglomerado, coletando informações sobre o estado do nó (uso de CPU, disco, memória, recursos de rede, etc) e as envia periodicamente ao GRM. Este último executa no nó gerenciador do aglomerado e usa as informações enviadas pelos LRMs para escalonamento das aplicações submetidas à grade.

LRM e GRM também colaboram no protocolo de alocação de recursos e execução de aplicações. O GRM seleciona nós candidatos para executar uma aplicação submetida pelo usuário da grade, com base nos requisitos da aplicação e na disponibilidade de recursos. Estas últimas informações são coletadas pelos LRMs nos nós fornecedores de recursos.

De modo semelhante à cooperação LRM/GRM, os módulos *Local Usage Pattern Analyzer* (LUPA) e *Global Usage Pattern Analyzer* (GUPA) também trabalham cooperativamente na análise de padrões de uso das máquinas para contribuir para um escalonamento eficiente na grade [Goldchleger, 2004].

O módulo *Application Repository* (*AppRepos*) permite o armazenamento e recuperação de aplicações.

2.4.4.1. Segurança

Os módulos que implementam os serviços de segurança são o LSM (*Local Security Manager*) e o GSM (*Global Security Manager*). Ambos utilizam a API GSS para obter os contextos de segurança entre o repositório de aplicações e os módulos que com ele interagem. O GSS, por sua vez, tem seus serviços implementados através do Kerberos. A implementação atual do repositório seguro utiliza a versão 5 do Kerberos desenvolvida pelo MIT[8] para a linguagem C. Para a linguagem Java, essa implementação utiliza a API GSS Java.

O GSM é responsável por iniciar e gerenciar os contextos. Cada cliente, através do LSM, possui um contexto de segurança com o GSM. Todas as trocas de mensagens entres os módulos são feitas utilizando esses contextos. O LSM pode efetuar quatro operações básicas: assinar e verificar uma mensagem enviada com o contexto que ela possui, ou ainda, assinar e verificar uma mensagem enviada sob um outro contexto. Mais especificamente, o repositório usa o GSM (via o LSM) para verificar e assinar os arquivos executáveis das aplicações dos seus clientes, enquanto estes clientes usam o LSM para assinar e verificar arquivos durante o armazenamento e recuperação do repositório.

O protocolo de armazenamento de uma aplicação do InteGrade é apresentado na Figura 2.21. O ASCT utiliza o LSM para assinar o arquivo e solicitar ao repositório de aplicações o seu armazenamento (1,2,3). O repositório de aplicações verifica a assinatura deste binário através do LSM (4), que o faz através do GSM (4.1). Uma vez o arquivo verificado com sucesso, o repositório de aplicações calcula um resumo do binário através de uma função *hash* conhecida[9], por exemplo, MD5, (5) e o armazena a seguir no sistema de arquivos (6). Ao final do processo de armazenamento em disco, o repositório de

[8] http://web.mit.edu/kerberos/www

[9] Para garantir que essa chave não possa ser gerada pelo atacante, o GSM adiciona ao arquivo uma chave que é obtida pela API GSS. Essa é uma técnica bastante utilizada em sistemas de segurança [Terada, 2000].

aplicações envia a identificação (ID) assinada da aplicação (7), que por sua vez também é verificada (8). Caso qualquer uma das operações falhar é gerada uma exceção que é devidamente tratada e registrada em arquivo de *log*.

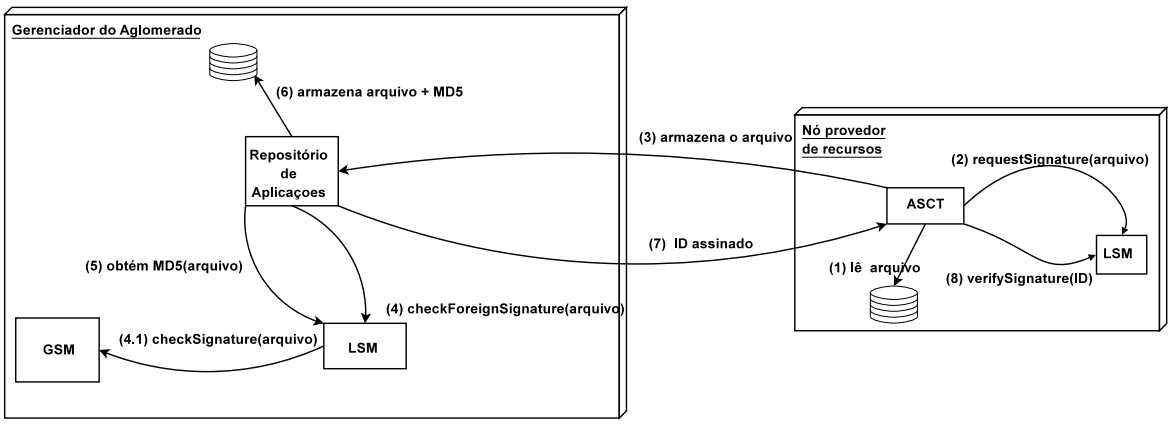

Figura 2.21. Protocolo de armazenamento de um executável de uma aplicação.

A Figura 2.22 apresenta o protocolo de recuperação de um arquivo. Uma vez que um determinado LRM recebeu a solicitação de execução de uma aplicação [Goldchleger, 2004], ele deverá requerer o binário compatível com a sua plataforma. De posse do ID da aplicação, o LRM o assina e indica ao repositório de aplicações o arquivo executável desejado (1,2). O Repositório obtém o arquivo desejado e verifica sua integridade através da função de *hash* (3,4). Antes de enviar o arquivo executável ao LRM, o repositório assina o binário através do LSM (5), que repassa essa função para o GSM (5.1), pois a assinatura deve ser feita através do contexto criado para o LRM que fez a requisição. Ao receber o arquivo, o LRM verifica sua assinatura (6,7) e o armazena em disco para execução (8).

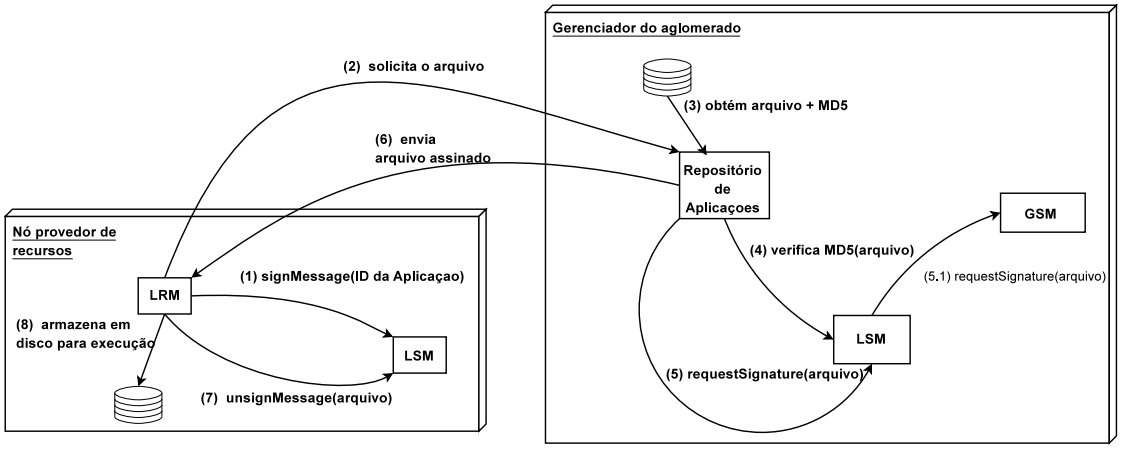

Figura 2.22. Protocolo de recuperação de um binário de uma aplicação.

Os protocolos acima descritos garantem a autenticidade e a integridade dos arqui-

vos executáveis submetidos à grade. Todas as mensagens trocadas entre o repositório de aplicações e seus clientes são devidamente assinadas e criptografadas. Pessoas mal intencionadas terão mais dificuldade ao utilizar técnicas de ataque conhecidas para modificar, interceptar ou fabricar dados, prejudicando assim o uso da grade. O uso de função de *hash* no sistema de arquivos é útil para tentar impedir a modificação das aplicações através de falhas de segurança no sistema de arquivos do gerenciador do aglomerado.

2.4.5. OurGrid

O OurGrid utiliza o MyGrid para implementar um sistema de Grade baseado numa rede *peer to peer*. O MyGrid é um sistema que teve como premissa de projeto contruir um sistema simplificado para executar aplicações sobre recursos computacionais distribuídos. No MyGrid o próprio usuário pode instalar um grade computacional com os recursos que dispõe. A instalação do MyGrid não requer nenhum privilégio especial de administrador.

O MyGrid define duas categorias de máquinas. A maquina base é ponto de acesso à Grade. Através da desta máquina, o usuário pode adicionar submeter e monitorar aplicações e ainda adicionar outras máquinas a Grade. As máquinas de *grid*, por sua vez, são as máquinas responsáveis pela execução de aplicação na Grade. As máquinas de *grid* e as máquinas base não necessitam compartilhar nenhum sistema de arquivo, bastando que sejam acessíveis pela máquina pelos usuários.

O MyGrid define a *Grid Machine Interface*, um conjunto mínimo de serviços que precisam estar disponíveis para que uma dada máquina possa ser adicionada ao Grid do usuário [Santos-Neto and Cirne, 2005]. Os serviços são transferências de arquivo, execução remota e interrupção de execução múltipla. O Mygrid provê as seguintes implementações para estes serviços:

1. *Grid Script* - usa aplicações básicas do sistema operacional;
2. *User Agent* - pequena aplicação desenvolvida em java;
3. Globus Proxy - direciona as operações necessárias para serviços implementados no Globus (GSI, GRAM e GridFTP).

O OurGrid é formado por três componentes básicos: MyGrid Broker, OurGrid *Peer* e um Sanboxing baseado no Xen denominada Swan. O MyGrid Broker provê um alto nível de abstração da grade. O Ourgrid é o componente responsável por gerenciar as máquinas que pertencem a um determinado domínio administrativo e obter acesso a máquinas em outros domínios [MyGrid/OurGrid, 2005]. E finalmente, o SWAN é uma solução de *sandboxing*[10] baseado na máquina virtual Xen*ho03:xen*.

2.4.5.1. Segurança

O OurGrid possui duas formas de autenticação: acesso direto e via OurGrid Peer. No acesso direto os usuários obtem os recursos pela máquina grid através de autenticação

[10]O *sandboxing* é uma tecnologia que cria ambientes de execução confinados para executar aplicações inseguras.

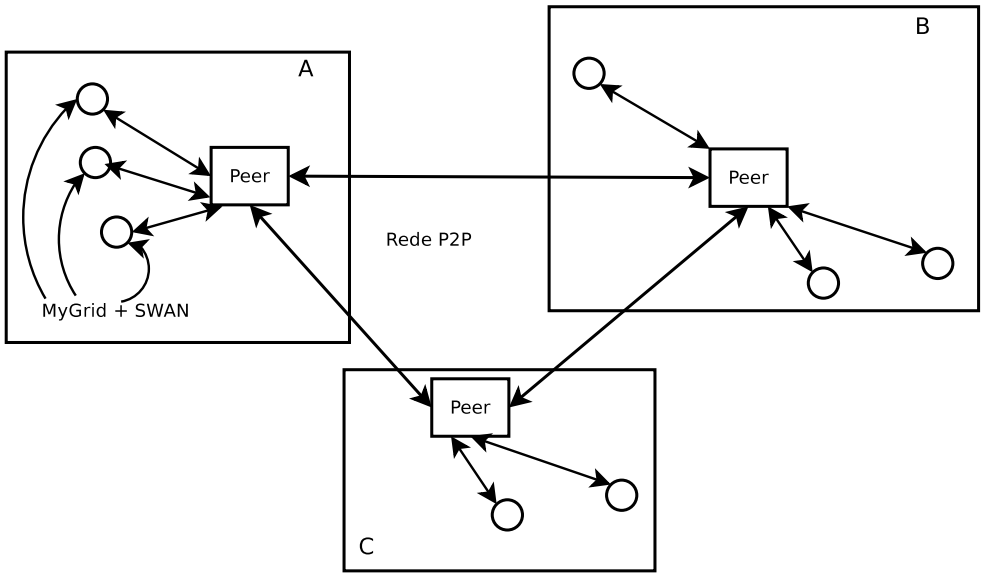

Figura 2.23. Arquitetura do OurGrid.

no sistema operacional via login e senha. A outra forma de autenticação é via certificados certificados digitais no formato X.509. Nesse modo de autenticação, somente módulos confiáveis podem comunicar entre si.

A autenticação do OurGrid é feita em duas fases [Santos-Neto and Cirne, 2005]. A primeira garante que o usuário tem permissão para solicitar serviços às maquinas grid. A segunda parte garante que o usuário não está solicitando serviços a uma falsa maquina grid. Dessa forma os usuários (através do broker), os OurGrid Peers gerenciam uma lista de certificados usadas para validar a tentativa de acesso.

O OurGrid utiliza os certificados para permitir comunicação segura entre o MyGrid Broker e o OurgridBroker. A segurança na comunicação é fornecida através do uso de RMI baseado em SSL (Secure Socket Layer), que garante comunicação criptografada.

Detsch [Detsch et al., 2004] propõe um arcabouço de segurança para redes P2P chamado P2PSLF (Peer-to-Peer Security Layer Framework) que foi implementado no OurGrid. Este arcabouço, ainda em desenvolvimento, define serviços de autenticação e confidencialidade para sistemas P2P. O P2PSFL foi implementado para ser completamente independente da aplicação do usuário. O arcabouço tem características bem interessantes como a modularidade, definição de requisitos de segurança por *peer* e redefinição dos requisitos dinamicamente.

A Figura 2.24 apresenta o resumo da arquitetura do P2PSLF. O arcabouço permite a definição de grupos de *peers* que compartilham requisitos de segurança. Cada *peer* possui perfis associados que representam os sentidos das mensagens entre os *peers* e os requisitos definidos. No exemplo da Figura 2.24, o *Peer* 1 exige autenticação quando houver troca de mensagens com os *Peers* 2 e 4. O *Peer* 1 também define que as mensagens com o *Peer* 3 serão autenticadas quando o sentido for do 1 para o 3 e autenticadas e confidenciais no sentido inverso. O *Peer* 3, por sua vez, envia mensagens autenticadas com confidencialidade para os *Peers* 1 e 4 e mensagens somente autenticadas no sentido

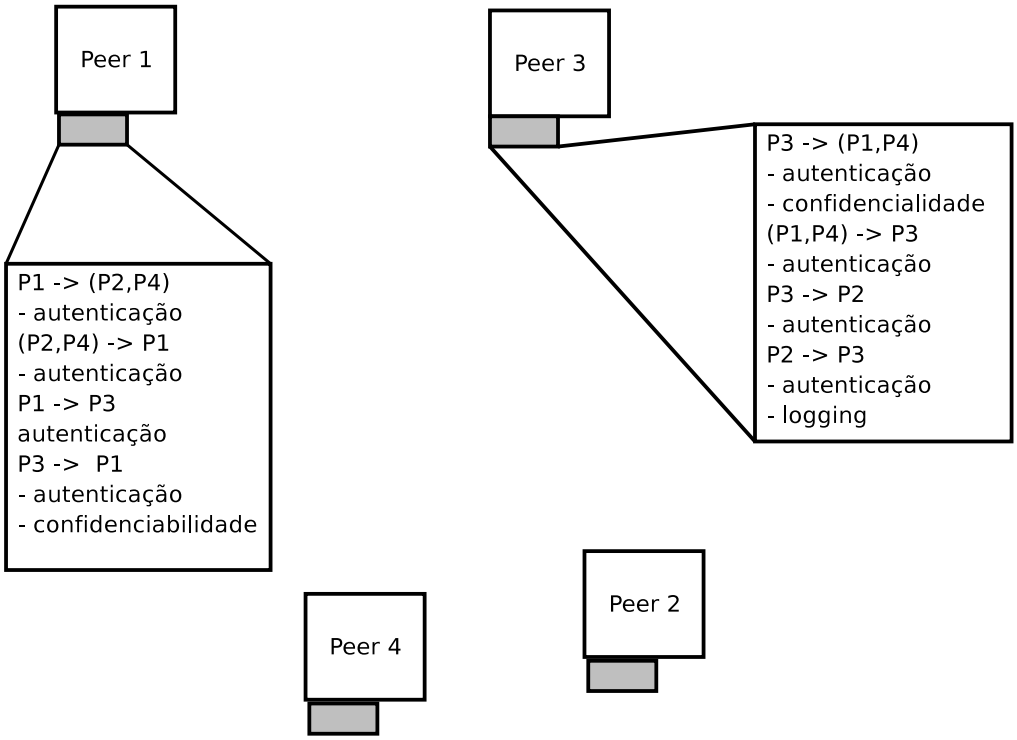

Figura 2.24. Arquitetura do P2PSFL.

contrário. E finalmente, o *Peer* 3 envia mensagens com autenticação para o *Peer* 2 e o mesmo *Peer* autentica e registra eventos ocorridos quando recebe mensagens do *Peer* 2.

Em cada um dos *peers* presentes na arquitetura há um módulo de configuração que é responsável pela descoberta de *peers* que formam o grupo. Assim que ingressa a rede, o módulo faz uma requisição especial de busca para descobrir as configurações de segurança dos *peers* remotos. Baseado nas informações devolvidas, o administrador pode reconfigurar os requisitos de segurança definidos para aquele nó.

2.5. Considerações finais

Neste capítulo fizemos uma breve introdução sobre segurança. Descrevemos os conceitos básicos de segurança e as suas ferramentas mais usadas. Apresentamos as Grades Computacionais, suas características, seus requisitos de segurança básicos e os mecanismos de segurança disponíveis para implementar seus requisitos. A seguir, levantamos o estado da arte em segurança de Grades Computacionais ao fazer um estudo de caso de implementações de segurança em grades computacionais.

As Grades Computacionais introduzem novos desafios na definição de soluções para seus requisitos de segurança. A característica distribuída e dispersa administrativamente requer a definição de mecanismos que sejam escaláveis, dinâmicos e confiáveis. A necessidade de que a autenticação e a autorização em uma grade seja feita de maneira separada direciona as soluções de segurança por um caminho diferente daquele necessário para tratar de sistemas convencionais.

As implementações de segurança nos sistemas de grades aqui apresentadas pos-

suem algumas caracteríticas em comum. Entre elas, a necessidade de criar implementações fléxiveis que permitam utilizar mecanismos diferentes de segurança. Sistemas como o Condor permitem que a autorização seja feita por mecanismos que vão desde uma infra-estrutura de chaves públicas até uma simples criação de um arquivo em um diretório temporário. O uso de *sandboxes*, protocolos padrão (como o SSL), utilização de delegação através de *proxies* são outros exemplos de caminhos compartilhados entre essas implementações de segurança.

2.5.1. Tendências futuras

Diversas tecnologias, tais como CORBA, JRMI e Serviços Web, surgiram ao longo dos últimos anos e poderiam ser usadas para padronizar uma infra-estrutura básica para grades computacionais e para segurança. Observamos, no entanto, que existe uma tendência (talvez pela própria caracteristica naturalmente heterogênia das grades computacionais) da convivência de padrões distintos. As grades terão que se adequar a essa realidade, sob a pena de ficarem isoladas, o que é um contra-senso pelos princípios definidos paras as grades computacionais.

O uso de *proxy* de certificados parece ser um bom caminho a ser seguido quando o problema for a delegação. O Globus parece ter norteado a questão com a definição da extensão dos certificados X.509. Eles apresentam uma boa solução para *login* único na rede, a delegação e a criação de identidades de uma maneira rápida e segura. Acreditamos no entanto, que a dependência ainda existente em um elemento central, a autoridade certificadora, ainda diminui a escalabilidade desta solução.

Alguns tipos de Grade podem requerer a autenticação de grupo. A autenticação de grupo permite que um conjunto de elementos compartilhem de forma segura chaves criptográficas. Dessa forma, é permitido que cada membro do Grupo possa obter privacidade em função do segredo que somente os membros do grupo conhecem. Essa tecnologia poderia ser usada par permitir, por exemplo, a transmissão de vídeos em difusão de forma segura.

Uma outra tendência a ser considerada são as federações. Uma federação é o agrupamento de elementos com afinidade. Um determinado congregado ao se filiar a uma federação, recebe uma identidade associada a esta federação. Com o processo de filiação, o congregado terá direitos a recursos disponíveis na federação a que pertence. Além de receber uma nova identidade, a federação facilita a administração da Grade.

O uso de *sandboxing* parece ser uma tendência como solução de isolamento de aplicações maliciosas em Grades Computacionais. A maioria dos sistemas de grade aqui apresentada utilizam a tecnologia de *sandboxing* para permitir que as aplicações sejam executads em um domínio de execução diferente. Esse isolamento, no entanto, é geralmente muito restritivo impedindo que as aplicações da grade tenham acessos a recursos. Além de tudo, por trás desta tecnololgia, geralmente há um ambiente que oferece uma sobrecarga considerável para os usuários (locais e da Grade).

Sabemos no entanto que não existe uma solução única e genérica para a segurança em Grades Computacionais. Na verdade até apostamos que sua característica dinâmica e flexível faça com que as Grades Computacionais sempre estejam em permanentemente

desenvolvimento. Sempre que houver uma nova tecnologia de segurança, as grades deverão se adequar a elas, sem ter que esquecer de implementar soluções para a vasta quantidade de mecanismos legados ainda em uso. Finalmente, o mundo acadêmico ainda tem uma trilha muito longa para percorrer na definição de sistemas que se adequem aos novos ambientes criados no contexto da Computação em Grade.

Referências

[Adams and Lloyd, 2002] Adams, C. and Lloyd, S. (2002). *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Andrade et al., 2003] Andrade, N., Cirne, W., Brasileiro, F., and Roisenberg, P. (2003). OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 61–86. Springer Verlag. Lect. Notes Comput. Sci. vol. 2862.

[Bellovin and Merritt, 1990] Bellovin, S. M. and Merritt, M. (1990). Limitations of the kerberos authentication system. *SIGCOMM Comput. Commun. Rev.*, 20(5):119–132.

[Berman et al., 2003] Berman, F., Fox, G., Hey, A. J. G., and Hey, T. (2003). *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). *Web Services Architecture*. World Wide Web Consortium.

[Clarke et al., 1999] Clarke, D., Elien, J.-E., Ellison, C., Fredette, M., Morcos, A., and Rivest, R. L. (1999). Certificate chain discovery in spki/sdsi. To be published, November 1999.

[Cohen, 1987] Cohen, F. (1987). Computer viruses: theory and experiments. *Comput. Secur.*, 6(1):22–35.

[de Camargo et al., 2004] de Camargo, R. Y., Goldchleger, A., Carneiro, M., and Kon, F. (2004). Grid: An Architectural Pattern. In *The 11th Conference on Pattern Languages of Programs (PLoP'2004)*, Monticello, Illinois, USA.

[Dierks and Allen, 1999] Dierks, T. and Allen, C. (1999). RFC 2246: The TLS protocol version 1. IETF RFC Publication. Status: PROPOSED STANDARD.

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654.

[Ellison et al., 1999] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Bell, S., and Ylonen, T. (1999). SPKI Certificate Theory. Internet RFC #2693.

[Epema et al., 1996] Epema, D., Livny, M., van Dantzig, R., Evers, X., and Pruyne, J. (1996). A worldwide flock of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, 12:53–65.

[Foster and Czajkowski, 2005] Foster, I. and Czajkowski, K. (2005). Modeling and managing state in distributed systems: the role of ogsi and wsrf. In *Proceedings of the IEEE*, volume 93, pages 604–612.

[Foster and Kesselman, 1997] Foster, I. and Kesselman, C. (1997). Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 2(11):115–128.

[Foster and Kesselman, 2003] Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc.

[Foster et al., 2002] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Global Grid Forum, Open Grid Service Infrastructure Working Group.

[Foster et al., 1998] Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). A Security Architecture for Computational Grids. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 83–92.

[Foster et al., 2001] Foster, I., Kesselman, C., and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of Supercomputer Applications*, 15(3):200–222.

[Frey et al., 2002] Frey, J., Tannenbaum, T., Foster, I., Livny, M., and Tuecke, S. (2002). Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5:237–246.

[Garfinkel and Spafford, 1996] Garfinkel, S. and Spafford, G. (1996). *Practical UNIX & Internet Security*. O Reilly & Associates, Inc.

[Globus, 2004] Globus (2004). `http://www.globus.org`.

[Goldchleger, 2004] Goldchleger, A. (2004). Integrade: Um sistema de middleware para computação em grade oportunista. Tese de mestrado, IME/USP.

[Goldchleger et al., 2003] Goldchleger, A., Kon, F., vel Lejbman, A. G., and Finger, M. (2003). InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. In *Proceedings of the ACM/IFIP/USENIX Middleware'2003 1st International Workshop on Middleware for Grid Computing*, pages 232–234, Rio de Janeiro.

[Grafinkel and Spafford, 1996] Grafinkel, S. and Spafford, G. (1996). *Practical UNIX and Internet Security*. O'Reilly & Associates, Inc.

[Ian, 2005] Ian, V. W. (2005). Globus toolkit version 4 grid security infraestruture: A standards perspective. www-unix.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf.

[Kohl and Neuman, 1993] Kohl, J. and Neuman, C. (1993). The Kerberos network authentication service (v5). Internet RFC #1510.

[Krueger, 1988] Krueger, P. E. (1988). *Distributed scheduling for a changing environment*. PhD thesis, Madison, WI, USA.

[Lang and Schreiner, 2002] Lang, U. and Schreiner, R. (2002). *Developing Secure Distributed Systems with CORBA*. Artech House, Inc., Norwood, MA, USA.

[Lewis and Grimshaw, 1996] Lewis, M. J. and Grimshaw, A. (1996). The Core Legion Object Model. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC '96)*, pages 551–561, Los Alamitos, California. IEEE Computer Society Press.

[Lhee and Chapin, 2003] Lhee, K.-S. and Chapin, S. J. (2003). Buffer overflow and format string overflow vulnerabilities. *Softw. Pract. Exper.*, 33(5):423–460.

[Menezes et al., 1996] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. (1996). *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA.

[MyGrid/OurGrid, 2005] MyGrid/OurGrid (2005). `http://www.ourgrid.org`.

[NIST SP-800-12, 1995] NIST SP-800-12 (1995). An introduction to computer security: The NIST handbook. Special Publication SP 800-12, National Institute of Standards and Technology (NIST).

[Pearlman et al., 2002] Pearlman, L., Welch, V., Foster, I., Kesselman, C., and Tuecke, S. (2002). A community authorization service for group collaboration. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 50–59, Washington, DC, USA. IEEE Computer Society.

[Ramachandran, 2002] Ramachandran, J. (2002). *Designing security architecture solutions*. John Wiley & Sons, Inc., New York, NY, USA.

[Rivest and Lampson, 1996] Rivest, R. L. and Lampson, B. (1996). SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession.

[Santos-Neto and Cirne, 2005] Santos-Neto, E. and Cirne, W. (2005). *Minicurso: Livro Texto*, chapter Grids Computacionais: Da Computação de Alto Desempenho a Serviços sob Demanda, pages 15–60. Sociedade Brasileira de Redes de Computadores.

[Sesame, 2003] Sesame (2003). Secure European System for Applications in a Multivendor Environment. https://www.cosic.esat.kuleuven.ac.be/sesame/.

[Sinha, 1996] Sinha, P. K. (1996). *Distributed Operating Systems: Concepts and Design*. Wiley-IEEE Press.

[Stallings, 2002] Stallings, W. (2002). *Network Security Essentials: Applications and Standards*. Prentice Hall Professional Technical Reference.

[Tanenbaum and Steen, 2002] Tanenbaum, A. S. and Steen, M. V. (2002). *Distributed Systems. Principles and Paradigms*. Prentince Hall.

[Terada, 2000] Terada, R. (2000). *Segurança de Dados. Criptografia em Redes de Computadores*. Editora Edgard Blücher Ltda.

[W3C, 2000] W3C (2000). *Simple Object Access Protocol (SOAP) 1.1*. URL: http://www.w3c.org/TR/SOAP.

[Welch et al., 2004] Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., and Siebenlist, F. (2004). X.509 proxy certificates for dynamic delegation. NIST Gaithersburg MD, USA.

[Xavier, 2004] Xavier, F. C. (2004). Um sistema de autorização baseado em uma infraestrutura de gerenciamento de privilégio. Tese de mestrado, IME/USP.