Protótipo de uma Camada de Segurança para *Grids* computacionais denominada *CASGrid*

Silvio Augusto Langer[1], **Rogério Turchetti**[2], **Celio Trois**[2]

[1]Pontifícia Universidade Católica do Rio Grande do Sul - (PUCRS)
Av. Ipiranga, 6681 – 90169-900 – Porto Alegre – RS – Brasil

[2]CTISM - Universidade Federal de Santa Maria - (UFSM)
Avenida Roraima, 1000 – 97105-900 – Cidade Universitária – Santa Maria – RS – Brasil

silviolanger@gmail.com, {turchetti,trois}@inf.ufsm.br

Abstract. *This paper presents a security layer for authentication and confidentiality called CASGrid. To implement it was used asymmetric encryption and digital certificates. The CASGrid was implemented in SETolF grid. The paper is organized into two parts: first part provides authentication of users using digital certificates. Thus, certification authority was attached to grid, providing certificates. The authentication happens through the validation of client's certificate public key. The second part implements a Insurance Virtual Channel, based on session keys and asymmetric encryption. It provides integrity and confidentiality in messages exchange after the User Authentication.*

Resumo. *Este trabalho apresenta a CASGrid, uma camada de segurança para autenticação e confidencialidade, baseada em criptografia assimétrica e certificados digitais. A CASGrid foi implementada no grid SETolF, a fim de fornecer autenticação aos usuários, utilizando certificados digitais e prover a integridade e a confidencialidade na troca de mensagens após a autenticação do usuário, implementando um Canal Virtual Seguro, baseado em chaves de sessão e criptografia assimétrica. Ela fornece integridade e confidencialidade na troca de mensagens após a autenticação do usuário.*

1. Introdução

Certificado digital é um objeto eletrônico capaz de identificar uma entidade através de seus dados, sua chave pública e a assinatura de uma terceira parte confiável [Housley and Polk 2001]. Representa uma solução eficiente para garantir a segurança na comunicação e nas transações eletrônicas. Seu uso vem aumentando gradativamente, para as mais diferentes finalidades, tais como: assinaturas digitais, sigilo ou autenticação [Stallings 2008]. Independente da finalidade, a premissa que envolve o certificado digital é saber se quem está tentando comunicar-se, realmente afirma ser quem diz ser.

Se avançarmos em direção à segurança em *grids*, nota-se que existe uma intensa comunicação entre as entidades comunicantes no ambiente, onde clientes realizam solicitações e servidores respondem às mesmas. No entanto, neste cenário surgem vulnerabilidades que podem acarretar consequências indesejáveis. Sem uma rotina estabelecida para autenticação e sigilo das informações o uso dos recursos de um *grid* pode facilmente ser burlado.

Assim, a implementação de autenticação, utilizando certificados digitais é uma resposta ao problema, fornecendo um sistema mais seguro. Baseada na criptografia assimétrica, certificados digitais necessitam de uma Autoridade Certificadora (AC) responsável pela sua emissão. Um certificado, para ser usado amplamente por várias aplicações, necessita seguir um padrão, como por exemplo o X.509 [Stallings 2008]. Outrem, o uso da criptografia para prover sigilo das informações agrega ao sistema a confidencialidade e a integridade das informações.

Neste sentido, o presente trabalho propõe uma camada de segurança em *Grids (CASGrid)*, cuja finalidade é prover autenticação e sigilo entre os usuários do *Grid* SETolF (Serviço de Escalonamento Tolerante a Falhas) [Mozzaquatro et al. 2008] e seu escalonador através de certificados digitais e um canal virtual seguro. Ressalta-se que esta camada é genérica, podendo ser estendida para *grids* em geral. Na *CASGrid* a emissão dos certificados digitais dos usuários pela AC dar-se-á utilizando objetos distribuídos em java. A implementação da *CASGrid* utiliza-se da API (*Application Programming Interface*) *Bouncy Castle* [of the Bouncy Castle 2009] e a comunicação é baseada em Java RMI (*Remote Method Invocation*). Além disso, o trabalho propõe uma forte criptografia através da combinação dos algoritmos MD5(*Message-Digest algorithm 5*) e RSA (cujo nome está associado aos seus criadores *Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman*).

Este artigo está organizado da seguinte maneira. A Seção 2 discute trabalhos relacionados em segurança para *grids* e modelos de autenticação. A Seção 3 irá detalhar a *CASGrid* e a integração com o SETolF, bem como experimentos práticos realizados. A Seção 4 encerra o artigo com as considerações finais.

2. Trabalhos Relacionados

Uma proposta que se assmila a *CASGrid* é a Peer-to-Peer Security Layer (P2PSL) [Detsch et al. 2006], a qual trata de questões de segurança em aplicações P2P, permitindo que cada *peer* especifique seus requisitos de segurança, de forma distinta para cada canal de comunicação. Os requisitos de segurança são satisfeitos por módulos que implementam diferentes técnicas de segurança. Esta camada de segurança é feita com base em perfis, cada um servindo a diferentes necessidades de segurança.

Para realizar a comunicação segura, a P2PSL utiliza-se de um par de chaves que pode ser publicada de duas formas. A primeira delas, descentralizada usando PGP (*Pretty Good Privacy*), a segunda, centralizada, utilizando uma AC. Sempre que um peer receber uma mensagem assinada por uma chave desconhecida, ele solicita a AC a chave pública correspondente ao *peer* remoto armazenando-a localmente para uso futuro. O experimento da P2PSL foi realizado utilizando OurGrid.

A semelhança que a P2PSL possui com este trabalho deve-se ao fato da *CASGrid* focar-se exclusivamente na relação entre cliente e escalonador, o que pode ser considerado uma arquitetura *peer-to-peer*. A semelhança com este trabalho também é encontrada na utilização da troca de chaves com a AC, embora a P2PSL não utilizar de certificados e na troca de chaves entre usuário e SETolF ao estabelecer o Canal Virtual Seguro.

A segunda proposta [Foster et al. 1998], trata do *Grid Security Infrastructure* (GSI), o qual utiliza-se de certificados para prover autenticação, tal como a CASGrid o faz. O GSI permite uma autenticação única do usuário na grade, onde o GRAM (*Globus Resource Allocation Manager*) verifica se o usuário pode executar o recurso em questão.

Caso o usuário tenha o acesso permitido, é criado um *Job Manager*, que é responsável por iniciar e monitorar a tarefa submetida [Licht 2006]. Seu principal objetivo, é fazer com que os serviços disponibilizados pela grade e que estiverem disponíveis na rede, não precisassem de privilégios locais especiais para executarem, reduzindo assim o risco de ataques [Licht 2006].

O uso de certificados digitais para autenticação também é validado pelo DCOCE (Digital Certificate Operation in a Complex Environment), projeto da Universidade de Oxford [Norman 2004] que explorou as vantagens e desvantagens da utilização de certificados digitais para usuários como meio de autenticação on-line em um ambiente de ensino superior e concluiu que seu uso é viável e escalável.

3. A Camada de Segurança *CASGrid*

Em *grids* de computadores a entrada e saída de usuários compõe uma rotina a qual necessita ser monitorada, a fim de oferecer segurança a quem acessa o serviço. Neste contexto, segundo Pedroso [Pedroso 2006], um aspecto crítico é a autenticação de usuários em diferentes domínios administrativos. Esta Seção discorre sobre a emissão de Certificados Digitais para prover a autenticação entre usuários e SETolF, e posteriormente estabelecer um Canal Virtual Seguro entre os mesmos.

3.1. Autenticação na *CASGrid*

Para estabelecer a autenticação entre usuário e SETolF faz-se necessário possuir um certificado assinado, o qual será emitido pela AC do SETolF. Ao solicitar um certificado, o cliente envia sua chave pública para a AC. A autoridade certificadora, usando sua chave privada, juntamente com a chave pública do cliente (utilizando algoritmo RSA com 1024 *bits*), gera um certificado.

Durante o processo de autenticação, é informada a identidade do usuário que será associada a esta requisição. Além disto, ao informar esta identidade, é criado um registro no escalonador associando a identidade com o *host* do cliente. Este registro será utilizado futuramente para verificar se o cliente já foi autenticado no Servidor. A requisição gerada combinada com o certificado auto-assinado da Raiz, são os elementos que irão prover a identidade de um certificado digital [Adams and Lloyd 2004].

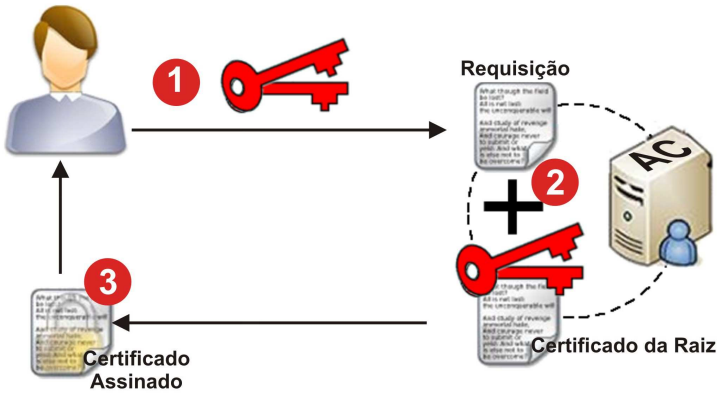

Figura 1. Esquema da Geração de Certificados Assinados na *CASGrid*

A Figura 1 mostra resumidamente a geração do certificado requerido pelo usuário. Os passos são detalhados a seguir: **(1)** o usuário envia seu par de chaves para a AC com

a finalidade de gerar a requisição do certificado. **(2)** A AC gera o certificado Raiz auto-assinado a partir das chaves pública (cliente), privada (AC) e identidade (cliente). **(3)** Ao final deste processo, o cliente possuirá um certificado assinado, o qual tem a finalidade de permitir o acesso na comunicação deste com o SETolF.Ao final deste processo, o cliente possuirá um certificado assinado pela AC, permitindo o acesso na comunicação deste com o SETolF.

O certificado gerado obedece o padrão X.509. O padrão não dita o uso de um algoritmo específico, mas recomenda o RSA (*Ronald Rivest, Adi Shamir e Leonard Adleman*). Todos os certificados X.509 obedecem o padrão internacional ITU-T X.509 [Licht 2006]. Atualmente este padrão está na versão 3 e dita que um certificado contenha estas informações, as quais são criptografadas. A Figura 2 apresenta o certificado gerado através da execução de um experimento prático utilizando a *CASGrid*.

```
  [0]         Version: 3
         SerialNumber: 1260317708722
             IssuerDN: CN=Autoridade Certificadora SETolF
           Start Date: Tue Dec 08 22:15:08 BRST 2009
           Final Date: Tue Dec 08 22:15:18 BRST 2009
            SubjectDN: CN=Langer
           Public Key: RSA Public Key
             modulus: 862fb2af428d494c5f30a7a493e0fe0856fabddf3ba5ac9454982af147c2f14d8fd95782c689353503ed7f2b2bad9c4fe8821c20
112295d112a80f3d9409294df3444b3a322a4e1e9e5cb1e7d6f7dadbfe7c802e23a43b9e9593b96d122ca6bc9d6e2143ea08f6c2fbf2ff41becf581919caf
c5a2e040fa9ef74b79f8590af
    public exponent: 10001

  Signature Algorithm: MD5WithRSAEncryption
            Signature: 21783c5be1752cebc096f9e90139a287c068a447
                       ba70fc616262c9ca8159f6bf662d05c053104ccf
                       6d75534646245e397a63e46ede6e72ce5c472ee4
                       745cb3e67c88dca2fa14ede1deaeb4b8d319e33c
                       23aba52a7428c07d5e4deba8f35f10e177910db9
                       5a9e9fe805c23fc1f6d7f66f609dec1c913a9f34
                       6baaf02d90555312
```

Figura 2. Certificado do Usuário gerado pela AC

As especificações do certificado são detalhadas a seguir:

- *Version*: a versão do formato do certificado.
- *Serial number*: uma sequência alfanumérica que identifica de forma única o certificado;
- *IssuerDN*: identifica quem é a AC. Neste trabalho identificada como Autoridade Certificadora SETolF fazendo referência a finalidae da AC.
- *Validity (Start Date / Final Date)*: esta é uma informação crucial num certificado. Todo certificado é válido somente por um intervalo de tempo. Desta forma, um certificado não é válido antes da data inicial e nem tampouco é válido após a data final. É importante afirmar que, na geração do certificado, pode se considerar o limite temporal para a validade do certificado.
- *SubjectDN*: é para quem o certificado foi gerado. É o proprietário do certificado.
- *Public key info*: é a chave pública associada ao certificado. Esta é a chave que será extraída para prover a autenticação do usuário.
- *Signature algorithm*: cada certificado possui uma assinatura digital criptografada. Este campo indica qual algoritmo criptográfico foi utilizado.
- *Signature*: é a assinatura digital criptografada do certificado. É gerada a partir das informações do próprio certificado e é o que impede que as informações do certificado sejam modificadas.

A finalidade dos certificados digitais neste trabalho, é prover a autenticação dos usuários do SETolF, que escalona processos em *grids* visando manter o funcionamento

normal do sistema mesmo na presença de falhas. O SETolF proporciona ao usuário diversas funcionalidades para o processamento de aplicações distribuídas. Assim, um aspecto importante a considerar neste trabalho, refere-se a extração da chave pública contida no certificado do usuário. O papel desta chave é essencial para que seja efetuada a autenticação, visto que durante a geração do certificado esta mesma chave é associada ao certificado gerado.

A sequencia necessária para realizar a autenticação, conforme ilustra a Figura 3, a qual é detalhada a seguir: **(1)** Cliente envia o certificado criptografado com a chave pública do SETolF. **(2)** O escalonador do SETolF, descriptografa o certificado e o envia para a AC, criptografando-o novamente com a chave pública da AC. **(3)**AC, verifica se o certificado é válido e em seguida compara a data atual com a data de validade do certificado. Caso estas premissas se confirmem, a comunicação entre cliente e servidor poderá ser estabelecida e o envio de tarefas ao escalonador poderá ser executada. Caso contrário, não haverá comunicação. **(4)** AC retorna ao SETolF a resposta. **(5)** Em caso de resposta afirmativa, o cliente está autenticado e pode enviar suas tarefas ao SETolF.

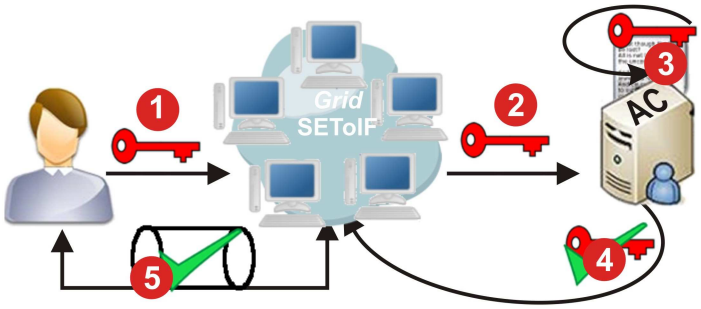

Figura 3. Autenticação utilizando a chave pública do certificado na *CASGrid*

A Figura 3, também elucida a necessidade de confiar na AC, pois todo processo descrito ocorre mediante a confirmação da veracidade da associação entre a chave pública do certificado e deste propriamente dito. Uma vez que os certificados ficam armazenado na AC, a validação do processo obrigatoriamente passa pela AC. Neste sentido, o trabalho desconsiderou possíveis falhas que possam ocorrer na AC. Ainda nesta perspectiva, um problema que pode trazer conseqüências no sistema de autenticação da *CASGrid* está no *spoofing* da entidade certificadora. *Spoofing* é uma técnica na qual o endereço real do atacante é mascarado, de forma a evitar que ele seja encontrado ou fazer-se passar por outra entidade [Cole 2001]. A solução para este problema seria a aplicação de um canal seguro com o uso de criptografia entre o SETolF e a AC.

O diagrama de classes, conforme mostra a Figura 4, ilustra a implementação da *CASGrid* com o relacionamento entre as classes e como o conceito de objetos distribuídos foi aplicado ao trabalho. Para efetivar esse conceito, utilizou-se duas interfaces para que clientes pudessem acessar métodos na AC, e esta, por sua vez, acessar métodos do Cliente, bem como outras duas interfaces para que o SETolF possa acessar os métodos da AC e vice-versa.

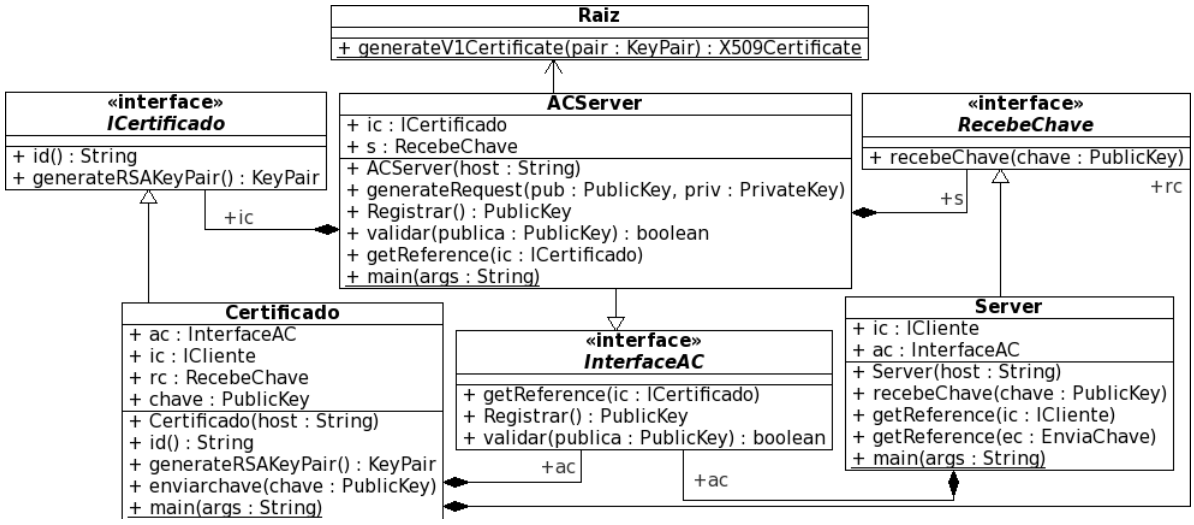

Figura 4. Diagrama de classes da autenticação na *CASGrid*

Classe Raiz: Classe responsável pela geração do certificado auto-assinado. A AC depende do certificado gerado por esta classe para a assinar o dos demais certificados solicitados pelos usuários. O método *generateV1Certificate()* recebe por parâmetro o par de chave geradas pelo método *generateRSAKeyPair()* da própria classe. Ambos são instanciados pela classe *ACServer*.

Classe ACServer: Responsável pela emissão dos certificados assinados e de registrá-los no repositório de certificados válidos. Para tanto, no método construtor *ACServer* o servidor remoto da AC é registrado enviando sua referência ao cliente. A emissão dos certificados dá-se pelo método *Registrar()* o qual invoca o método *generateRSAKeyPair()* da classe Certificado responsável pela criação do par de chaves do cliente.

Em seguida, é instanciado o método *generateRequest* que recebe por parâmetro a chave pública do cliente, gerando uma requisição. Nessa requisição estão os dados necessários para identificar o portador da chave pública correspondente, que deverá ser incorporada ao certificado. Após criar o certificado assinado do cliente, o método inclui o certificado gerado no repositório de certificados válidos, associando a chave pública do cliente ao respectivo certificado.

Classe Certificado: Esta classe tem por objetivo receber o certificado gerado e registrado pela classe ACServer. Para tanto no método construtor da classe invoca o método *Registrar()*utilizando-se da interface *InterfaceAC*. Este certificado é assinado utilizando-se a combinação dos algoritmos MD5 e RSA com 1024 *bits*.

A classe possui o método *String id()*, que objetiva informar o *common name* do certificado do cliente, o qual é instanciado no método *Registrar()* da classe *ACServer*. O método *generateRSAKeyPair()* também é instanciado no método referido da classe *ACServer*. A classe ainda é responsável pelo envio da chave pública do certificado com o objetivo de autenticar o cliente ao SETolF.

Classe Server: A classe representa o SETolF. Para tanto, no método construtor *Server* o servidor remoto do SETolF é registrado enviando sua referência ao cliente. Sua função é de autenticar o cliente, através da validação da chave enviada pelo cliente. Para tanto, o método *recebeChave* recebe por parâmetro a chave extraída do certificado. Para enviá-la a AC o método invoca *validar* da classe ACServer.

InterfaceAC: prove a ligação do cliente com a AC, que objetiva a manipulação dos objetos residentes no servidor, considerando a impossibilidade do cliente copiá-los. Nesta interface é feita a declaração das funções invocadas remotamente. Para isso, a interface deve ser estendida da classe *Remote*. O único método ao qual o cliente terá acesso na AC será o *Registrar()*.

Classe ICertificado: de forma inversa a InterfaceAC, esta classe permite a ligação da AC com o cliente, permitindo que esta manipule objetos residentes no cliente. A classe ICertificado estende os métodos *id()*, *generateKeyPair()* para que a AC tenha acesso aos mesmos.

RecebeChave: permite que a classe *Certificado* possa acesssar o método *recebeChave* da classe *Server*.

enviaChave: estabelece a comunicação entre a classe *ACServer* e *Server* permitindo que o SETolF possa acessar o método *validar* da classe *ACServer*.

3.2. Canal Virtual Seguro

O Canal Virtual Seguro (CVS) implementado pela *CASGrid*, tem a função de prover o sigilo na troca de mensagens entre cliente e SETolF. É importante reforçar que este canal somente existirá se o cliente estiver autenticado, caso contrário o mesmo não será constituído e o envio de tarefas ao escalonador não será possível. O CVS para prover tal funcionalidade utiliza-se de chaves simétricas e assimétricas.

Uma maneira de impedir que uma pessoa não-autorizada tenha acesso às informações que cruzam a rede, é tornar o conteúdo das mensagens ilegível, só podendo ser compreendido pelo destinatário pretendido. Na criptografia, a mensagem a ser criptografada é conhecida como texto original. Para criptografar este texto, é utilizado um algoritmo parametrizado por uma chave [Tannenbaum 2003]. Para descriptografar este texto, é necessário que o destinatário conheça a chave pública da origem.

O CVS, conforme Figura 5, utiliza-se de cifras simétricas para criptografar os dados a serem enviados. Sua escolha deve-se ao fato de ser mais rápido em relação a outros algoritmos de cifras simétricas, proporcionando mais segurança e velocidade na comunicação [of Standards and Tecnology 2001]. No entanto, um ponto crítico na criptografia simétrica diz respeito ao compartilhamento de chaves, pois uma interceptação das mesmas, no momento da distribuição, comprometeria toda a segurança na comunicação [Stallings 2008]. Esse impasse é solucionado pela *CASGrid* com a adição do uso da criptografia assimétrica.

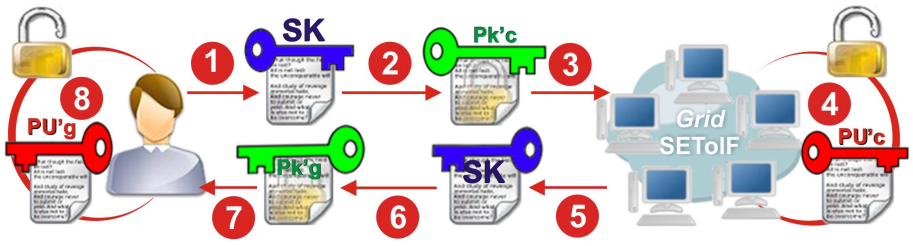

Figura 5. Canal Virtual Seguro da *CASGrid*

Por ser computacionalmente mais cara, a criptografia assimétrica geralmente é usada em conjunto com a simétrica, agindo justamente onde esta é mais fraca, ou seja,

no compartilhamento do segredo, bem como na manutenção das chaves. Desta forma, o canal seguro é implementado conforme mostra a Figura 5 e detalhada a seguir: **(1)** Ilustra o texto claro ou, neste caso, um arquivo *.class* sendo encriptado com a chave simétrica (**SK**), ou chave secreta. **(2)** A chave simétrica é encriptada com a chave privada do cliente (**Pk'c**). **(3)** Ilustra o envio do dado criptografado para o escalonador. **(4)** Escalonador descriptografa o dado enviado com a chave pública do cliente (**PU'c**), acessando remotamente o método *decifra()*. Em seguida distribuí o arquivo para processamento pelos *peers*. **(5)** Após receber o resultado dos *peers* o escalonador aplica a chave simétrica (**SK**) do escalonador sobre o dado a ser enviado ao cliente. **(6)** A chave simétrica do escalonador é encriptada pela chave privada (**PK'g**) do escalonador. **(7)** Escalonador envia o dado ao Cliente. **(8)** Cliente recebe o dado criptografado e descriptografa o mesmo com a chave pública do escalonador (**PU'g**) acessando remotamente o método *decifra()* do escalonador.

A Figura 6 mostra o Diagrama de Classes que implementa o Canal Virtual Seguro. Nas classes *Server* e *Cliente* as chaves simétricas são geradas utilizando o algoritmo AES de 128 *bits* e o par de chaves gerado pelo métodos *generateKeyCliente()* e *generateKeyServer()* utilizam o algoritmo RSA com 1024 *bits*. A classe *printHexa* tem a função de gerar um *array* de *bytes* para mostrar o dado ilegível. Além disso, para tornar possível a comunicação remota entre as classes utilizou-se duas interfaces.

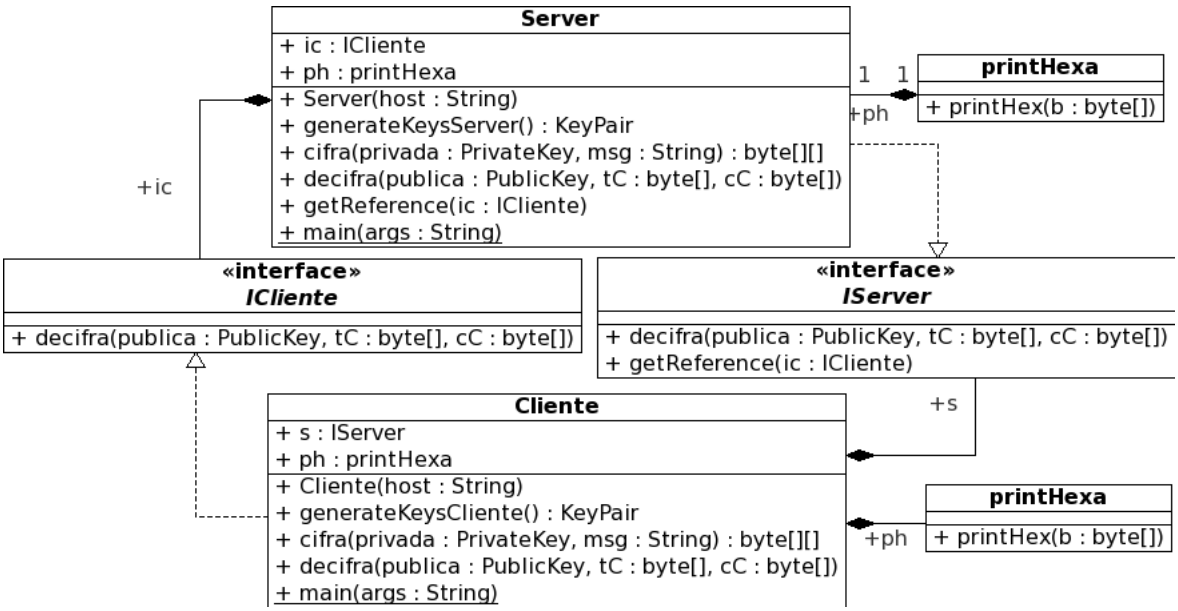

Figura 6. Diagrama de classes do Canal Virtual Seguro da *CASGrid*

Classe Server: Representa o escalonador do SETolF. Estabelecido o CVS e agrega dois métodos que serão responsáveis pela criptografia e a descriptografia dos dados recebidos do cliente. A comunicação do escalonador com o cliente é realizada através de RMI (*Remote Method Invocation*), sendo que para criptografar os dados do escalonador, no método construtor da classe, é invocado o método *cifra()* recebendo por parâmetro o texto claro e a chave privada do escalonador.
No método construtor da classe também é realizada a descriptografia dos dados recebidos pelo cliente. Para poder ler os dados criptografados do cliente é invocado

remotamente o método *decifra* da classe *Cliente*. Este método recebe por parâmetro a chave pública, o texto cifrado e a chave cifrada do cliente. O resultado da ação deste método é o texto claro, possibilitando o escalonador ler os dados enviados pelo cliente.

Classe Cliente: Esta classe é responsável pela criptografia dos dados do cliente e descriptografia dos dados recebidos do escalonador. A criptografia dos dados do cliente é realizada no método construtor da classe, invocando o método *cifra* recebendo por parâmetro o texto claro e a chave privada do cliente. O processo para realizar a criptografia segue o mesmo procedimento descrito na classe *Server*.

Para descriptografar os dados recebidos do escalonador é invocado remotamente o método *decifra* da classe *Server*, recebendo por parâmetro a chave pública, o texto cifrado e a chave cifrada do escalonador. O resultado da ação deste método é o texto claro, possibilitando o cliente ler os dados enviados pelo escalonador.

Classe IServer: Interface cujo objetivo é prover a ligação do cliente com o escalonador, permitindo a manipulação dos objetos residentes no SETolF, considerando a impossibilidade do cliente copiá-los. Nesta interface é feita a declaração das funções invocadas remotamente. Para isso, a interface deve ser estendida da classe *Remote*. Um dos métodos ao qual a classe *cliente* terá acesso no escalonador é o *decifra*, que permite que o cliente acesse a chave pública do escalonador. O método *getReference*, por sua vez, faz com que o cliente envie sua referência ao escalonador.

Classe ICliente: De forma inversa a IServer, esta classe permite a ligação do SETolF com o cliente, permitindo que esta manipule objetos residentes no cliente. A classe estende apenas o método *decifra* para que o escalonador tenha acesso a chave pública do cliente.

3.3. Experimento para demonstrar o CVS em funcionamento

Após o desenvolvimento do Diagrama de Classes do CVS, houve a necessidade de realizar um experimento para mostrar que, de fato, o CVS estava realizando sua função de forma efetiva. Apesar da dificuldade de demonstrar a eficiência do canal, a forma encontrada para realizar este experimento foi utilizar uma aplicação sintética, para realizar a troca de mensagens entre dois usuários, visto que, a mesmo simula a comunicação realizada entre usuário e escalonador.

Em seguida, observou-se os pacotes na rede onde foram extraídas as informações necessárias. Este procedimento foi realizado com o analisador de redes *tcpdump* combinado a aplicação *Wireshark Network Analyzer*. Com o resultado de ambas aplicações, extraiu-se as informações contidas nas Figuras 7 e 8.

A Figura 7 exibe a utilização da aplicação sem os métodos *cifra()* e *decifra()*. Na parte destacada da Figura 7, visualiza-se a palavra digitada *"teste"* na íntegra, o que prova que mesmo o cliente estando autenticado ao escalonador, o dado pode ser interceptado e o acesso a informação estaria vulnerável em caso de um ataque.

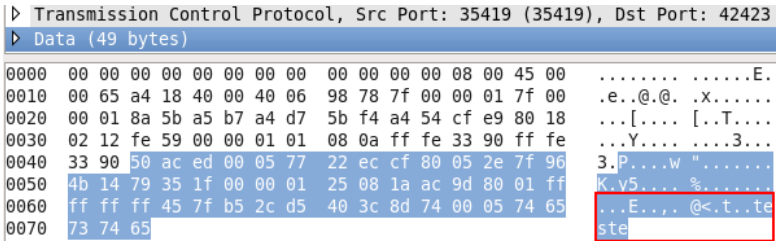

Figura 7. Teste sem o uso do Canal Virtual Seguro da *CASGrid*

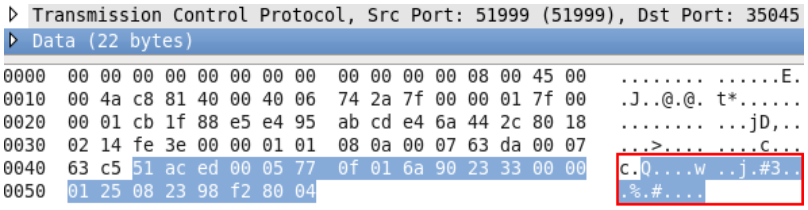

Figura 8. Teste com o uso do Canal Virtual Seguro da *CASGrid*

Ao contrário do que ilustra a Figura 7, a Figura 8 mostra a aplicação sintética utilizando os métodos *cifra()* e *decifra()*. Nela o texto digitado (*teste*) apresentado na parte destacada mostra que a leitura da palavra está incompreensível e fornece ao usuário e ao escalonador a integridade dos dados envolvidos na transação. Portanto, conclui-se que o CVS obteve sucesso na criptografia dos dados transmitidos, implementando a confidencialidade e a integridade na troca de mensagens.

3.4. Arquitetura do SETolF integrada com a *CASGrid*

O SETolF, escalona tarefas em *grids* visando manter o funcionamento normal do sistema, mesmo na presença de falhas. Para tanto, trabalha com um Serviço de Detecção de Defeitos Adaptativo (*AFDService*) [Nunes 2003] para detectar processos suspeitos. Com este serviço acoplado ao escalonador, garante-se que em caso de suspeita de algum processo, o recurso escalonado seja notificado e o sistema continue operando mesmo na presença de falhas.

Sua arquitetura é baseada em clientes, escalonador e *peers*. Onde os clientes enviam tarefas para serem executadas ao escalonador, que por sua vez distribuí-as por NFS aos *peers* que executam as tarefas e retornam ao escalonador a tarefa processada, repassando o resultado final ao cliente.

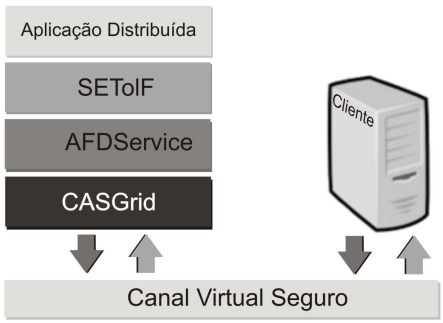

Figura 9. Integração do SETolF com a *CASGrid*

A Figura 9 exibe o SETolF integrado com a *CASGrid* ilustrando que o acesso ao serviço, passa obrigatoriamente pela *CASGrid*, onde inicialmente é realizado o processo descrito na Seção 3.1 e em seguida estabelecido o CVS. Somente após estas rotinas é instanciada a classe *AppMaster* no método construtor da classe *Server* e a classe *AppClient* no método construtor da classe *Cliente* A aplicação enviada pelo usuário, passa pelo *AFDService* chega ao escalonador que distribuí à aplicação aos *peers* para processamento. No SETolF, as aplicações recebidas devem estar no formato *.class* e sua construção é de responsabilidade do usuário.

A integração entre autenticação na CASGrid e o SETolF é realizada após ocorrer todo o processo de geração do certificado do cliente e a validação da chave. Somente após esse processo, as tarefas do cliente poderão ser executadas no ambiente distribuído.

4. Conclusões e trabalhos futuros

O uso de certificados digitais e chaves assimétricas proporcionam um eficiente mecanismo de segurança para prover autenticação. A associação da chave pública ao certificado reduz consideravelmente a possibilidade de ataque, pois mesmo a comunicação entre usuário e AC não ser criptografada, a comunicação entre escalonador e usuário somente ocorrerá após a verificação do escalonador com a AC.

Outra funcionalidade que a *CASGrid* oferece é a criptografia dos dados enviados para o Escalonador através do CVS. Embora a possibilidade de interceptação dos dados na troca de mensagens exista, a combinação da criptografia simétrica e assimétrica na implementação deste canal, mostra que a leitura destes dados torna-se extremamente difícil. Com esta funcionalidade, agrega-se a propriedade da confidencialidade ao canal seguro.

Embora os experimentos realizados destinaram-se ao SETolF, ressalta-se que a *CASGrid* pode ser aplicada em outros sistemas computacionais. Assim, autenticação, confidencialidade e integridade na comunicação são funcionalidades que estarão disponíveis aos usuários que implementarem a solução proposta.

Referências

Adams, C. and Lloyd, S. (2004). *Understanding PKI - Concepts, Standards, and Deployment Considerations*. Addison Wesley, Indianapolis, IN, USA, 2 edition.

Cole, E. (2001). *Hackers Beware: The Ultimate Guide to Network Security*. New Riders Publishing, Berkeley, CA, 1 edition.

Detsch, A., Gaspary, L. P., Barcellos, M. P., and Sanches, R. N. (2006). Uma abordagem para incorporação flexível de aspectos de segurança em aplicações peer-to-peer. *SBRC2006 - 24 Simpósio Brasileiro de Redes de Computadores*.

Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). A security architecture for computational grids. *Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92.

Housley, R. and Polk, T. (2001). *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. John Wiley and Sons, New York, NY, USA.

Licht, F. L. (2006). Fornecimento automatizado de certificados de curta duração para dispositivos móveis em grades computacionais. Master's thesis, Instituto Militar de Engenharia, Rio de Janeiro.

Mozzaquatro, B., Turchetti, R., Alves, V., ao, J. E., and Canal, A. P. (2008). Proposta de um ambiente de programação com técnicas de escalonamento distribuído - setolf. *Anais do XII Simpósio de Ensino Pesquisa e Extensão - SEPE*.

Nunes, R. (2003). *Adaptação Dinâmica do Timeout de Detectores de Defeitos Através do Uso de Séries Temporais*. PhD thesis, Universidade Federal do Rio Grande do Sul, Instituto de Informática, Curso de Pós-Graduação em Computação, Porto Alegre.

of Standards, N. I. and Tecnology (2001). Specification for the advanced encryption standard (aes). *Federal Information Processing Standards (FIPS)*, (197).

of the Bouncy Castle, T. L. (2009). Bouncy castle homepage. http://www.bouncycastle.org/. acessado em 17 de setembro de 2009.

Pedroso, E. T. (2006). *SeguranÃ§a em Grades Computacionais*. PhD thesis, Universidade Estadual de Campinas.

Stallings, W. (2008). *Criptografia e segurança de redes*. Pearson Prentice Hall, Rio de Janeiro, 4 edition.

Tannenbaum, A. S. (2003). *Redes de Computadores*. Campus/Elsevier, Rio de Janeiro, 10 edition.