Blindagem de uma Grade Computacional utilizando TPM e *Sandbox*

Luiz Fernando Rust da Costa Carmo, Roberto Paes Nemirovsky

Núcleo de Computação Eletrônica – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 23.24 – 20.010-974 – Rio de Janeiro – RJ – Brasil
rust@nce.ufrj.br, robertopaes@posgrad.nce.ufrj.br

Abstract*. The use of Computational grids enables the remote execution of users' applications into geographically dispersed computers, from different owners and domains. Computer interactions in this new context bring novel security challenges and meaningfully increase complexity of efficient solution for sharing resources inside a Grid. Security requirements of a computational grid are definitely different from those commonly found in restricted and controlled environments. This work aims to exploit all the power emerging from the possible use of Trusted Platform Module (TPM), as well as from virtualization techniques (Sandbox), to provide a safe environment for both grid nodes and users' codes. In this way, this paper presents an architectural proposal for Grids based on a join use of TPM and Sandbox, intending to provide a logically sealed environment for the processing of sensitive information.*

Resumo. *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 trabalho busca explorar todo o poder do Trusted Platform Module (TPM), bem como das técnicas de virtualização para fornecer um ambiente seguro tanto para os elementos de computação da grade (nós) quanto para os códigos de usuários a serem executados. Para isso é apresentada um proposta arquitetural para a concepção de uma grade integrando estes dois elementos (TPM e Sandbox), de forma a criar um ambiente logicamente blindado para o tratamento de informações sensíveis.*

1. Introdução

O mundo em que vivemos é cada vez mais dependente das novas tecnologias. A computação, em particular, surgiu para automatizar tarefas que eram executadas de forma trabalhosa. A ciência de uma forma geral, em todas suas áreas de conhecimento, é sem sombra de dúvida uma das beneficiadas pelas tecnologias emergentes oriundas da Ciência da Computação. Seu crescente uso fez surgir uma também crescente necessidade de poder computacional. Novos problemas, muitas vezes ainda não

experimentados pela sociedade, bem como problemas conhecidos, demandam grande tempo e poder de processamento. Os altos custos envolvidos na aquisição de computadores de alto desempenho os tornam praticamente inacessíveis para uma grande quantidade de potenciais usuários. Entretanto, os computadores pessoais de hoje possuem a mesma capacidade de processamento dos supercomputadores de anos 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].

Computação em grade pode ser definida 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 [2]. 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 [3]. 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 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.

Dois grandes problemas da computação em grades são a proteção dos nós que a compõem contra códigos maliciosos submetidos para execução; e a proteção contra nós comprometidos de códigos submetidos para a grade. O primeiro problema mencionado, proteger um nó de códigos maliciosos enviados para ele, pode ser resolvido através de técnicas de virtualização, onde um ambiente logicamente isolado do sistema principal é criado para abrigar o sistema de gerência da grade e a execução dos trabalhos submetidos para a mesma. Com isto, mesmo que o ambiente virtual da grade seja comprometido, não haverá risco de o sistema principal ser afetado. Já o segundo problema é extremamente complexo, pois, em última instância, o sistema operacional principal possui acesso irrestrito a todos os processos executados sobre ele, bem como a todo *hardware* presente.

O avanço e a popularização de co-processadores com funções de criptografia, como, por exemplo, o *Trusted Platform Module* (TPM) [4], favorece a adoção em massa de soluções de segurança mais robustas, tornando-as menos impactante para o usuário, uma vez que o processamento em *hardware* fornece um alto ganho de performance, e, além

disto, de forma ainda mais seguras, pelas próprias características do TPM, que torna logicamente inacessível o acesso às chaves privadas.

Por outro lado, técnicas de virtualização, como *Sandboxing*, isolam a ação de possíveis danos causados ao ambiente operacional, através de técnicas de gerenciamento de memória e controle de acesso a periféricos, adicionando pouca sobrecarga de processamento para o sistema.

O objetivo desde trabalho é investigar, explorar e sugerir formas de evitar que um nó de uma grade computacional seja comprometido, através da utilização do TPM e técnicas de virtualização, e, caso isso ocorra, impedir que o trabalho sendo executado na grade seja afetado ou capturado.

2. Trabalhos Relacionados

Quando se fala de segurança em grades computacionais há de se observar muitos aspectos distintos. A própria natureza distribuída de tais sistemas cria diversas possibilidades de ataque e pontos de exploração. Uma grade computacional pode ser composta por algumas unidades de máquinas trabalhando cooperativamente, ou milhares destas. Da mesma forma, estas máquinas podem estar fisicamente localizadas dentro de uma sala altamente protegida ou espalhadas por todo o mundo, em forma de máquinas de usuários expostas a vírus, *worms*, cavalos-de-tróia, *backdoors* e demais ameaças.

Com o objetivo de classificar os pontos de risco e analisá-los detalhadamente, [5] faz uma separação em quatro categorias: *naming and authentication; secure communication; trust, policy, and authorization;* e *enforcement of access control*. Além disto, ainda faz uma análise completa de cada categoria destas visando apresentar o estado da arte atual em termos de segurança em grades computacionais.

Um projeto para integrar as funcionalidades do *Trusted Platform Module* (TPM) à *Grid Security Infrastructure* (GSI), denominado Daonity [6], utiliza primitivas de criptografia executadas em *hardware* para proteger o ambiente da grade computacional.

Analisando os problemas de confidencialidade e integridade existentes na computação em grade, [7] identifica e discute diversas inovações que a tecnologia de *Trusted* Computing oferece para aumentar a segurança em grades.

[8] descreve uma proposta de virtualização do TPM, ou seja, disponibilizar um TPM virtual para máquinas virtuais mantendo todos os requisitos de *Trust* requeridos pelo TCG, ou seja, *Measurement*, *Attestation* e *Sealing*.

Outra utilização interessante do TPM, desta vez para atestar a integridade de *web services*, é descrita em [9]. Neste estudo, é apresentado um protocolo de comunicação que possibilita a um cliente solicitar uma verificação de toda a cadeia de software suportando o *web service* disponibilizado. Esta cadeia consiste da BIOS, carregador do S.O., kernel do S.O., módulos do S.O. dinamicamente carregados e o web service e seus arquivos em si.

A utilização de *Sandbox*, uma classe de virtualização, com objetivo de criar um ambiente isolado e dedicado para computação em grade é descrita em [10]. Neste, as

diversas classes de *Sandbox* são descritas e analisadas e, além disto, são apresentadas as soluções hoje existentes.

Diferentemente das propostas acima, este trabalho visa combinar estas duas tecnologias emergentes e cada vez mais difundidas, virtualização e TPM, para isolar mutuamente os ambientes de computação da grade e do nó. Através da virtualização confina-se o código enviado para a grade a um ambiente controlado. Este ambiente por sua vez, utilizando-se das propriedades do TPM, torna-se blindado contra ataques externos a ele, oriundos do comprometimento do nó, possibilitando que dados sensíveis sejam efetivamente protegidos, independente do que ocorra com o nó.

3. Conceitos básicos

3.1. *Trusted Platform Module*

A mesma facilidade que permite um sistema evoluir o seu *software* para torná-lo mais eficiente e seguro, permite que um atacante remoto comprometa o mesmo a fim de alterar a sua funcionalidade, e esse passe a trabalhar a favor do atacante, quer seja pelo roubo de informações, quer seja pela falsificação das mesmas. Com base na dificuldade de se garantir a confiabilidade e integridade de um sistema (um *software*) unicamente por meio de um sistema de segurança (outro *software*), soluções por meio de *hardware* foram propostas para diminuir ou até mesmo anular a eficácia de um ataque remoto, trazendo maior segurança aos sistemas existentes.

Entre as diversas iniciativas existentes, destaca-se a do *Trusted Computing Group* (TCG), uma organização sem fins lucrativos criada em 2003. Inicialmente teve como membros participantes AMD, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft e Sun Microsystems. Sua maior contribuição foi o desenvolvimento do *Trusted Platform Module* [4], um circuito semicondutor que é capaz de armazenar informação de forma segura, inviolável por ataques via software. O TPM permite o cálculo e armazenamento de *hashes* em registradores internos, conhecidos como *Platform Configuration Registers* (PCR), facilidades para a geração segura de chaves criptográficas e números aleatórios, atestação remota e blindagem de memória.

A blindagem de memória evita que um código externo ao programa em execução possa ler ou modificar a região de memória reservada para o mesmo; realizada por um conjunto de primitivas TPM que garante acesso exclusivo a áreas protegidas (de memória, registrador, etc.), onde é possível trabalhar com dados críticos.

A atestação remota da plataforma é feita pela assinatura digital de estruturas internas do TPM através do uso de AIKs (A*ttestation Identity Key*), i.e., de características da plataforma que são armazenadas nos registros internos do TPM; para ser mais preciso, a atestação gerada não é de identidade e sim semântica, provendo evidências que um determinado software foi carregado na plataforma em uma determinada ordem desde o boot até o presente.

O armazenamento seguro permite que uma informação, ao ser armazenada em disco, seja criptografada usando criptografia AES e sua chave (criptografada) seja armazenada em disco.

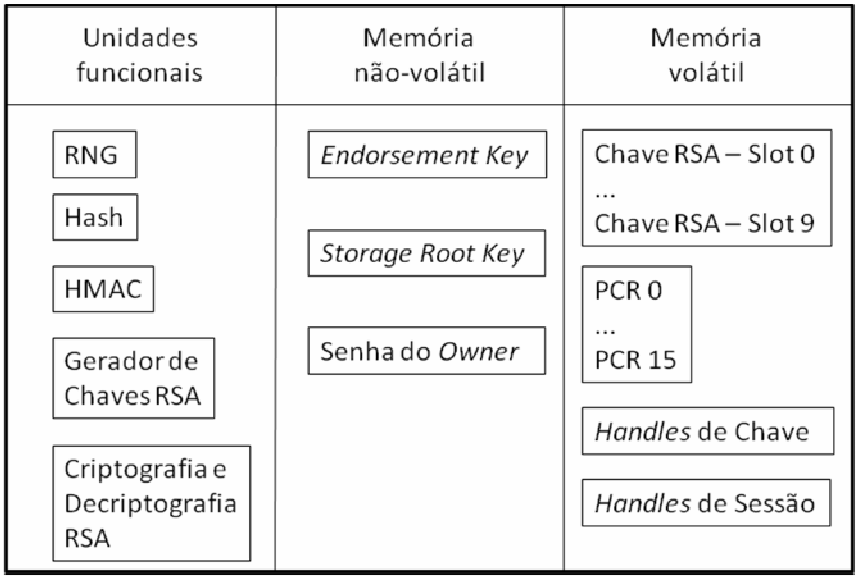

Figura 1: Módulos funcionais do TPM

O conjunto de registradores PCR é de suma importância na arquitetura do TPM. Estes registradores são inicializados junto com a inicialização do sistema e só podem ser modificados através de uma reinicialização do sistema ou através da operação de extensão. Esta operação funciona seguindo a seguinte função:

$$\text{Extend}(PCR_N, \text{valor}) = \text{SHA1}(PCR_N \parallel \text{valor})$$

Estas extensões são usadas durante o processo de boot, inicialmente pela BIOS, referênciada como *Core Root of Trust for Measurement* (CRTM), e em seguida por todos os módulos envolvidos na carga do sistema, e um valor final de PCR representa a acumulação de uma seqüência única de mensurações. Seu valor final é utilizado para decidir quando um sistema pode ser considerado confiável.

Modos de Comunicação previstos:

- *Binding*: operação tradicional de criptografia de mensagem pelo uso da chave pública do destino (apenas o destino pode conhecer o conteúdo da mensagem);
- *Signing*: associa um espelho da integridade de uma mensagem com a chave usada para gerar sua assinatura (o TPM rotula alguma de suas chaves como “chaves de sinalização” cuja função é criptografar o *hash*);
- *Sealing*: é a operação de *binding* condicionada à checagem de métricas da plataforma (especificadas pelo autor da mensagem), de forma que algumas métricas devem ser satisfeitas para que a mensagem possa ser decodificada; a operação de blindagem associa a chave de seção a um conjunto de valores do PCR; uma mensagem blindada é criada selecionando um intervalo de PCRs e criptografando assimetricamente seus valores junto com a chave simétrica usada para codificar a mensagem; de posse da chave privada (criptografia assimétrica), o destinatário apenas recupera a chave simétrica quando o sistema estiver de acordo com os valores PCR solicitados (ou seja, quando a plataforma estiver funcionando em um estado bem específico);

- *Sealed-signing*: inclui verificação da preservação da integridade na operação de *sealing*.

3.2. Virtualização (*Sandbox*)

Computadores modernos são suficientemente poderosos para uso de virtualização de modo que, nos dias de hoje, tornou-se possível a execução de vários sistemas completos (programas básicos e programas aplicativos) sob um único *hardware* físico. A esta representação de vários sistemas completos sendo executados sob um mesmo *hardware* físico dá-se o nome de virtualização de software. Isto foi alcançado através da abstração de um *hardware* convencional atribuindo a maior parte, e/ou em alguns casos, toda a sua funcionalidade ao software. A este conceito de abstração deu-se o nome de "máquina virtual" que é um software representando as funcionalidades de um *hardware* [11].

O uso da virtualização representa a ilusão de várias máquinas virtuais (VMs) independentes, cada uma rodando uma instância de um sistema operacional virtualizado [12].

O uso da tecnologia de virtualização se difundiu bastante devido ao fato da mesma propor soluções para problemas reais existentes na computação moderna. Mas a idéia de virtualização não é tão recente assim. Inclusive, a virtualização na camada do sistema operacional data de um bom tempo atrás, onde a primeira máquina virtual foi o VM/CMS da IBM que se refere a família System/370, System/390, zSeries, System z9 IBM *mainframes* e sistemas compatíveis. Criado no final da década de 70 e utilizado até hoje pela IBM, os sistemas de virtualização estão cada vez mais presentes na realidade tecnológica mundial [13].

Atualmente, os sistemas virtualizados estão conquistando seu espaço devido ao fato de resolverem alguns pontos que hoje são críticos em diversas empresas tais como: incompatibilidade entre *hardware* e *software* no que diz respeito a suas modificações no decorrer do tempo, ou seja, o *hardware* das empresas estão sendo atualizados a uma velocidade maior que os programas legados que devem ser executados e que são responsáveis por controlar a atividade fim das empresas; subutilização dos recursos de *hardware* pelos programas pelo mesmo motivo citado anteriormente, ou seja, os programas legados não conseguem explorar em sua totalidade a capacidade dos *hardware* atuais; dentre outros.

O sucesso da tecnologia de virtualização se baseou em alguns princípios. Primeiramente, a camada de virtualização deve isolar uma máquina virtual da outra de modo que não exista nenhuma interferência entre ambas. Não é aceitável que o funcionamento de uma máquina virtual afete a performance de outra máquina virtual [14]. Segundo, é necessário suportar uma variedade diferente de sistemas operacionais para acomodar os diferentes aplicativos populares existentes [14]. Terceiro, o *overhead* introduzido pela camada de virtualização deve ser pequeno [14].

Por outro lado, a virtualização ocupa um papel muito importante na área de segurança da informação, uma vez que, devido ao seu próprio modelo de funcionamento, um ambiente virtual funciona de forma independente e isolada do sistema operacional principal. Neste modelo, todo acesso ao *hardware* é interceptado pelo gerenciador do

ambiente virtual (*hypervisor*), sendo possível, assim, um controle total do sistema virtualizado e um isolamento efetivo do mesmo.

Unindo estas duas tecnologias apresentadas, [4] propõe uma arquitetura para disponibilizar para um ambiente virtual todas as funcionalidades do TPM, sem perder a cadeia de *Trust*. Em sua proposta, cada instância de uma máquina virtual possui em seu Sistema Operacional um *device driver* que se comunica com o gerenciador do TPM Virtual (vTPM). Este gerenciador por sua vez, controla todas as requisições vindas dos ambientes virtuais, e se comunica diretamente com o TPM.

A grande contribuição de [4] é a possibilidade de realização de *measurement* do ambiente virtual, utilizando posições de PCR superiores do TPM físico para armazenar métricas de integridade da *Sandbox* e do que nela é executado. Isto possibilita assegurar que toda a cadeia de *boot* deste ambiente é genuína, ausente de códigos maliciosos. Cada ambiente virtual possui sua própria chave EK (*Endorsement Key*), bem com sua própria chave SRK (*Storage Root Key*). Toda e qualquer informação persistente dos TPM virtuais é mantida em memória persistente, criptografada e associada ao conjunto de PCR da máquina virtual.

A conseqüência direta da realização de *measurement* no ambiente virtual é a utilização do PCR para registro das métricas geradas. Com isto, informações sensíveis processadas dentro da *Sandbox* podem ser protegidas e atreladas (*sealing*) a este subconjunto do PCR utilizado. Assim, mesmo que um código executando do lado de fora da *Sandbox* consiga acesso ao TPM, ele não poderá realizar a operação de *unseal* para decriptografar as informações sensíveis, pois o conjunto PCR real será distinto do conjunto PCR do ambiente virtual.

4. Arquitetura

A arquitetura proposta neste trabalho visa fornecer um ambiente confiável para execução de códigos em uma grade computacional, protegendo os dados sensíveis contidos na grade em caso de comprometimento do nó onde o código é executado, e também a situação oposta, protegendo o nó de possíveis códigos maliciosos enviados para a grade.

4.1. Requisitos

Para atender todos os requisitos de segurança envolvidos na comunicação do Gerente da grade com um determinado nó, bem como a execução do código submetido para este com isolamento mútuo entre o sistema operacional do nó e o ambiente computacional da *Sandbox*, a arquitetura da grade deve contemplar os seguintes pontos:

- Proteção da comunicação entre o Gerente da grade e o nó;
- Proteção do sistema operacional do nó contra códigos maliciosos executados na grade;
- Proteção das informações manipuladas no nó contra códigos maliciosos alojados no sistema operacional principal.

4.2. Módulos funcionais

O modelo de arquitetura pode ser decomposta em módulos com funções específicas para garantir os requisitos desejados, como mostrado a seguir:

- Gerente da grade: responsável pelo recebimento de códigos de usuários para execução, escolha do recurso (nó) onde será feita a computação, envio do código para execução, recebimento do resultado da computação, repasse do resultado para o usuário;
- Gerente do nó: recebe requisições do Gerente da grade, e cria Sandbox a ser utilizada na computação;
- Cliente da grade: comunica-se diretamente com o Gerente da grade, recebe o código, realiza a computação e envia o resultado.

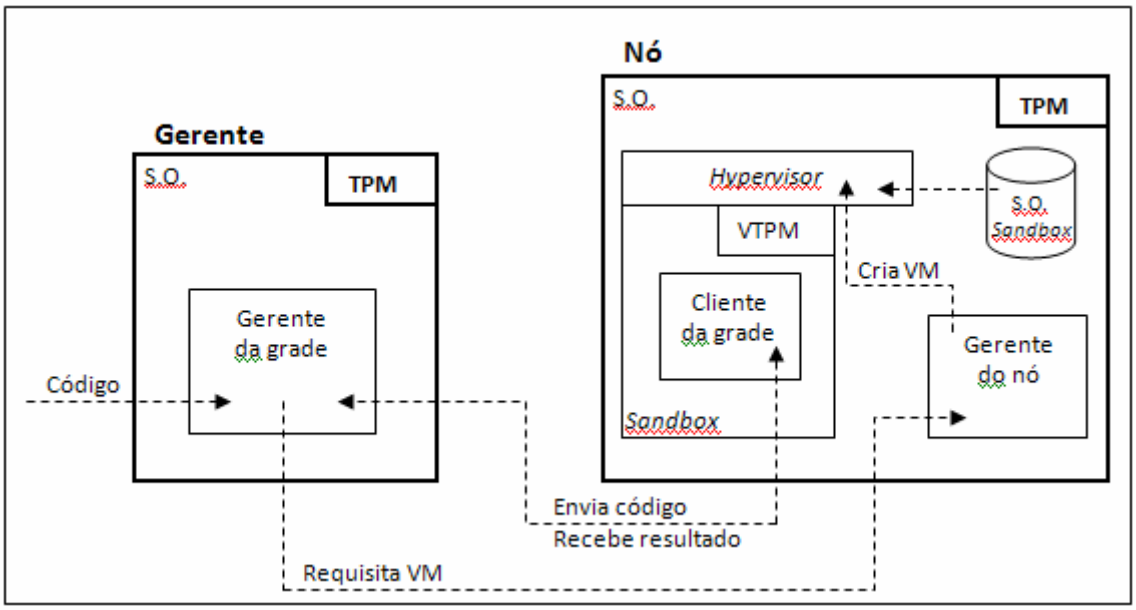

Figura 2: Arquitetura geral do sistema

4.3. Funcionamento

Ao receber uma requisição de um cliente para execução de código (*job*) na grade, o Gerente da grade elege um nó para tal, de acordo com seus critérios. Neste momento, o Gerente da grade conecta-se ao nó selecionado, onde ocorre a primeira verificação de segurança: tanto o Gerente da grade quanto o Gerente do nó verificam mutuamente a autenticidade da outra ponta, através da análise dos certificados digitais de ambos, ou seja, verifica-se que não há um atacante fazendo-se passar por um elemento da grade.

Após esta etapa, o gerente requisita uma *Sandbox* para a execução do código enviado pelo cliente. Este ambiente é criado a partir de um arquivo, representando a imagem de um *file system*, assinado digitalmente pelo Gerente da grade, logo, caso o mesmo seja modificado, o Gerente do nó irá detectar a alteração.

Para que haja uma verificação efetiva da integridade inicial da *Sandbox*, é necessário que um ambiente novo seja sempre criado e inicializado para cada *job* a ser executado.

Somente desta forma o TPM poderá realizar a atestação de toda a cadeia de *boot* do ambiente virtual, verificando a integridade do BIOS, *Boot loader*, *kernel*, módulos, etc. Assim, tem-se a certeza que todos os componentes do ambiente virtual foram carregados e funcionam conforme esperado pelo Gerente da grade. A este processo de verificação do TPM dá-se o nome de *Attestation*.

Caso haja uma tentativa de se criar uma *Sandbox* a partir de um arquivo que não o determinado pelo Gerente da grade, a *Sandbox*, através do mecanismo de TPM Virtual apresentado em [8], irá detectá-la e o processo será interrompido. Mesmo que se crie um ambiente virtual paralelo, de forma que não haja bloqueio do processo de *boot* em função de alterações na *Sandbox*, no momento em que o Cliente da grade tentar se comunicar com o Gerente da grade, este irá detectar que a cadeia de atestação da *Sandbox* não corresponde ao esperado, a um estado íntegro, pelo fato desta comunicação ser validada remotamente pelo TPM do Gerente da grade. A este processo de verificação remota dá-se o nome de *Remote Attestation*.

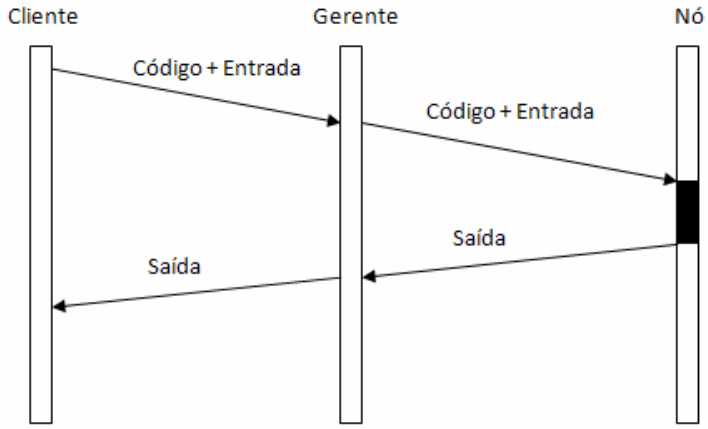

Figura 3: Diagrama temporal do funcionamento

Um importante ponto na arquitetura proposta é a separação em dois módulos, chamados Gerente do nó e Cliente da grade, dentro de um determinado nó. Esta separação se justifica pelo fato do Gerente do nó não precisar se envolver efetivamente na computação da grade, sendo assim, seu comprometimento não afeta o processamento do *job*. O Cliente da grade, por sua vez, tem seu código protegido pelo TPM Virtual, bem como sua comunicação com o Gerente da grade.

Uma vez que ambas as partes, Gerente da grade e nó, se autenticam entre si, através da verificação dos certificados digitais e da atestação remota do TPM, e garantem que todo o sistema está íntegro, ocorre a troca de mensagens entre os sistemas, e o código a ser executado é enviado ao nó.

O código a ser executado deve levar em conta a existência de um TPM Virtual e se utilizar das funcionalidades, protegendo todo e qualquer informação sensível. Esta proteção deve se utilizar da funcionalidade *Sealing* do TPM, ou seja, a chave criptográfica deve ser atrelada ao conjunto de PCR do ambiente virtual. Desta forma, somente um código sendo executado dentro da *Sandbox* poderá realizar a decriptografia dos dados. Com isto, o ambiente virtual se torna blindado, pois nenhum código executado fora da *Sandbox* conseguirá acesso a dados sensíveis protegidos.

4.4. Premissas adotadas

Na arquitetura proposta têm-se as seguintes premissas:

- O Gerente da grade encontra-se em um estado confiável, livre de qualquer código malicioso;
- O Gerente da grade possui TPM;
- O sistema operacional a ser executado na *Sandbox* também está livre de qualquer código malicioso caracterizado como sendo legítimo, ou seja, incluído na cadeia de atestação do TPM;
- O nó possui TPM;
- O *hypervisor*, responsável pelo gerenciamento da *Sandbox*, possui suporte ao TPM Virtual conforme apresentado em [4];
- O código a ser executado se utiliza de primitivas de criptografia para proteção de dados sensíveis.

5. Avaliação da proposta arquitetural

Em um ambiente de computação em grade, o sistema de computação fica exposto a possíveis ataques em diversos pontos do processo, principalmente por sua característica inerentemente distribuída. Este trabalho não busca resolver todos os problemas de segurança associados à computação em grade, e tem como foco principal evitar a interferência de códigos maliciosos injetados no processo, ou que executem no mesmo ambiente físico que ele.

A introdução do TPM na arquitetura da grade traz avanços significativos em termos de segurança. Somente baseando-se em suas características, a proteção das chaves privadas contra ataques lógicos e físicos vem eliminar um grande ponto de atenção da tecnologia de Infra-estrutura de Chaves Públicas, a segurança da chave privada. Sua propriedade de atrelar operações de criptografia a um conjunto de PCR (*Sealing*), que serve para atestar a integridade inicial do sistema, garante que a informação protegida pelo TPM só poderá ser acessada em ambientes confiáveis. Aliado a isto, outra importante funcionalidade do TPM, a atestação remota, permite que a comunicação entre o Gerente da grade e o Cliente da grade só ocorra se ambos estiverem íntegros.

O uso de Sandbox em conjunto com TPM Virtual, ou seja, proporcionar a cada ambiente virtual todo o poder do TPM de forma exclusiva, com todas as garantias de *Trust* requeridas pelo TCG, propicia à grade um ambiente altamente confiável e controlado. Desta forma, independente do estado em que se encontra um determinado nó, pode-se ter certeza que a *Sandbox* a ser utilizada não se encontra comprometida. Além disto, a operação de *sealing* do TPM, quando utilizada dentro da *Sandbox*, vincula a criptografia de dados sensíveis da computação ao conjunto de PCR do ambiente virtual, tornando impossível a um código externo realizar a descriptografia destes dados, criando então um ambiente logicamente blindado para execução segura de códigos.

Por outro lado, o uso da virtualização protege o sistema operacional principal do nó. Isto ocorre devido ao controle da execução do código inserido no ambiente virtual, através de técnicas de gerência de memória. De forma semelhante, todo acesso ao *hardware*

também é controlado pelo gerenciador do ambiente virtual. Além de proteger o sistema operacional principal, outra característica importante é o isolamento entre duas *Sandbox* executando no mesmo sistema.

A tabela 1 resume todos os elementos oferecidos pelas tecnologias de TPM e *Sandboxing* utilizadas neste trabalho, bem como o aumento do nível de segurança fornecido pela sua utilização.

Tabela 1: Aumento de segurança proporcionado pela arquitetura

Ameaça	Funcionalidade utilizada	Impacto na segurança
Código malicioso enviado para a grade	Utilização de *Sandbox*	Confina a ação de qualquer código malicioso enviado para a grade à *Sandbox*, protegendo assim o nó.
Interferência entre as várias execuções ocorrendo na grade	Utilização de *Sandbox*	Isola logicamente as várias computações da grade sendo executadas em um determinado nó.
Atacante se passando por Gerente	Certificação digital no Gerente e no Nó	Permite autenticação de ambas as partes
Atacante se passando por Nó	Certificação digital no Gerente e no Nó	Permite autenticação de ambas as partes
Sandbox com sistema operacional comprometido (Nó comprometido antes do início da computação)	Atestação local da *Sandbox* usando TPM	Permite comprovar que nenhum componente da *Sandbox* foi alterado
Sandbox com sistema operacional comprometido (Nó comprometido antes do início da computação)	Atestação remota da *Sandbox* usando TPM	Permite ao Gerente comprovar a integridade da *Sandbox* remotamente Comunicação só é permitida caso a integridade seja atestada
Captura de informações sensíveis sendo processadas na *Sandbox* (Nó comprometido após o início da computação)	Operação de *sealing* do TPM – Vinculação de operações de criptografia a um determinado conjunto de PCR	- Permite que somente códigos executados dentro da *Sandbox* tenham acesso a informações sensíveis da computação - Protege as informações da grade contra códigos maliciosos executando fora da *Sandbox*
Captura de informações sensíveis trafegando na rede	Blindagem da comunicação utilizando TPM	Utiliza de chaves de criptografia geradas e protegidas pelo TPM

6. Conclusões e Trabalhos Futuros

O advento dos sistemas computacionais em grade traz embutido novos requisitos de segurança e aumenta a complexidade das possíveis soluções. O trabalho apresentado neste artigo descreveu uma proposta arquitetural para a concepção de uma grade integrando o poder do *Trusted Platform Module* (TPM), com a potencialdade das técnicas de virtualização. O Objetivo é fornecer um ambiente seguro tanto para os elementos de computação da grade (nós) quanto para os códigos de usuários a serem executados.

A combinação de TPM com técnicas de virtualização foi materializada através da proposta de utilização de técnicas de virtualização do TPM, disponibilizando todas as suas funcionalidades do TPM em um ambiente virtual, sem perder a cadeia de *Trust*.

Uma contribuição importante da arquitetura proposta é a necessidade de que um ambiente novo seja sempre criado e inicializado para cada *job* a ser executado, de forma que haja uma verificação efetiva da integridade inicial da *Sandbox*. Somente desta forma o TPM é capaz de realizar a atestação de toda a cadeia de *boot* do ambiente virtual, verificando a integridade do BIOS, *Boot loader*, kernel, módulos, etc. Além disto, a operação de *sealing* do TPM, quando utilizada dentro da *Sandbox*, vincula a criptografia de dados sensíveis da computação ao conjunto de PCR do ambiente virtual, criando então um ambiente logicamente blindado para execução segura de códigos.

Por outro lado, o uso da virtualização protege o sistema operacional principal do nó devido ao controle da execução do código inserido no ambiente virtual e ao controle de acesso ao *hardware* em geral. Outra característica importante é o isolamento entre dos *jobs* em execução no mesmo sistema.

Visando a validar qualitativamente a proposta descrita neste artigo, é apresentada uma análise exaustiva das vulnerabilidades existentes; sendo que para cada vulnerabiliade é apresentada a funcionalidade necessária, e como esta é usada para sua eliminação.

Os trabalhos em andamento incluem uma avaliação quantitativa do desempenho desta proposta através de avaliações empíricas e simulações.

7. Referências

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

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

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

[4] Trsuted Computing Group – TCG, (2006). “Trusted Platform Module specification”, TPM Work Group, dísponível em https://www.trustedcomputinggroup.org/groups/tpm/.

[5] Security for Grids. M. Humphrey, M. Thompson, and K.R. Jackson. Proceedings of the IEEE (Special Issue on Grid Computing), vol 93, No. 3, March 2005. pp. 644 -- 652.

[6] Daonity: grid security with behaviour conformity from trusted computing. Wenbo Mão, Fei Yan, Chunrun Chen. Proceedings of the first ACM workshop on Scalable trusted computing, 2006. pp. 43 – 46.

[7] Innovations for Grid Security from Trusted Computing, Wenbo Mao, Hai Jin e Andrew Martin

[8] vTPM: Virtualizing the Trusted Platform Module, S. Berger, R. Cáceres, K. Goldman, R. Perez, R. Sailer and L. van Doorn, Proc. of 15th USENIX Security Symposium, July 2006.

[9] Satem: Trusted Service Code Execution across Transactions. G. Xu, C. Borcea and L. Iftode. In the Proceedings of the 25th Symposium on Reliable Distributed Systems (SRDS), October 2006.

[10] I-Cluster: The Execution Sandbox, Bruno Richard.

[11] BARHAM, P.; DRAGOVIC, B.; FRASER, K.; HAND, S.; HARRIS, T.; HO, A.; NEUGEUBAUER, R.; PRATT, I.; WARFIELD, A. Xen and the Art of Virtualization. 2003. University of Cambridge Computer Laboratory.

[12] SMITH, J. E.; NAIR, R. The Architecture of Virtual Machines. p. 32–38, maio 2005.

[13] wikipedia -VM_CMS, 2007

[14] GARFINKEL, T.; PFAFF, B.; CHOW, J.; ROSENBLUM, M.; BONEH, D. Terra: A Virtual Machine-Based Platform for Trusted Computing. 2003.

[15] Segurança em Grades Computacionais. José Braga Pinheiro Jr. e Fabio Kon. Minicurso apresentado no 20o Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. pp. 65-111. SBC. Florianópolis, 2005.

[16] http:// www.xensource.com/

[17] Suh, Gookwon E. (2005), “AEGIS: A Single-Chip Secure Processor”, Tese de doutorado submetida ao Departamento de Engenharia Elétrica do MIT.

[18] Lie, D., Thekkath, C., Horowitz, M., (2003). “Implementing an Untrusted Operating System on Trusted Hardware”, In the SOSP’03, New York, USA.

[19] Menon, A., Renato Santos, J., Turner, Y. (2005). “Diagnosing Performance Overheads in the Xen Virtual Machine Environment”, In the First ACM/USENIX Conference on Virtual Execution Environments (VEE'05), Chicago, USA.