UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

DISSERTAÇÃO
apresentada à UTFPR
para obtenção do grau de

MESTRE EM CIÊNCIAS

por

RICARDO AUGUSTO DE OLIVEIRA

DESENVOLVIMENTO DE UM REPOSITÓRIO SEGURO INTEGRADO AO GERENCIAMENTO DE REDES ATIVAS BASEADO EM POLÍTICAS

Banca Examinadora:

Presidente e Orientador:

Prof.Dr. LUIZ NACAMURA JR.	**UTFPR**

Examinadores:

Prof. Dr. JONI DA SILVA FRAGA	**UFSC**
Prof. Dr. LAU CHEUK LUNG	**PUC-PR**
Prof(a). Dr(a). ANELISE MUNARETTO FONSECA	**UTFPR**

Curitiba, Junho 2006.

RICARDO AUGUSTO DE OLIVEIRA

DESENVOLVIMENTO DE UM REPOSITÓRIO SEGURO INTEGRADO AO GERENCIAMENTO DE REDES ATIVAS BASEADO EM POLÍTICAS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial do Centro Federal de Educação Tecnológica do Paraná, como requisito parcial para a obtenção do grau de "Mestre em Ciências" – Área de Concentração: Telemática.

Orientador: Prof. Dr. Luiz Nacamura Júnior

Curitiba
2006

O48d Oliveira, Ricardo Augusto de
Desenvolvimento de um repositório seguro integrado ao gerenciamento de redes Ativas baseado em políticas / Ricardo Augusto de Oliveira. Curitiba. UTFPR, 2006
XII, 99 f. : il. ; 30 cm

Orientador: Prof. Dr. Luiz Nacamura Júnior
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2006
Bibliografia: f. 87-89

Redes de computadores – Gerenciamento. 2. Criptografia. 3. Telecomunicações. I. Nacamura Junior, Luiz, orient. II. Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: 004.62

Agradecimentos

Em primeiro lugar agradeço a Deus por tudo que tem feito por mim, por colocar em meu caminho as pessoas que me ajudaram a realizar este trabalho e por ter iluminado minha mente para encontrar as soluções dos diversos problemas que surgiram no decorrer desta caminhada.

A minha esposa Juliana e ao meu filho Lucas, minhas grandes paixões, agradeço pelo enorme incentivo, cumplicidade, amor e carinho que sempre me deram. Agradeço pela paciência e compreensão nos momentos de ausência, em que deixei de ser pai e esposo para poder me dedicar a este trabalho.

Minha eterna gratidão a meu pai Lourival e minha mãe Irma pelo dom da vida e pelo apoio incondicional. Agradeço o apoio dos meus irmãos Fábio e Juliana, cunhados Emerson, Rossiane, José Nogueira e Jussara, tio Plínio e da minha querida sogra Maria Auxiliadora. Agradeço especialmente a minha irmã Silvia, que me ajudou a realizar o sonho da graduação e que por isso é grande responsável pelos sucessos que tenho obtido em minha vida.

Agradeço ao meu orientador Prof. Dr. Luiz Nacamura Junior, grande responsável pela conclusão deste trabalho, pela oportunidade dada de mostrar meu potencial, pela confiança, paciência e incentivo que me deu durante todo o caminho.

Agradeço o apoio dos amigos Hernani Sozzi Jr e sua esposa Geci, Jéferson Massinhan, Juliano Toaldo, Arthur Ravache, Sandra Marcondes, Rubens Kichner, Roberto Teixeira, Andrey Fisher, Rodrigo Soto, Ricardo Lopes, Eduardo Schnell, Cleide Jane, Alexander Burbelo, Kiko, Daniel Koeller, Edson Barreto, em especial ao amigo Anderson Shirata pelo grande incentivo e ao amigo Marcus Vinicius pelo apoio técnico.

Agradeço aos amigos do LASD, Fernando, Marcos, Henrique, Yoquir, Hermes, Nico e em especial a Adriano Ferrasa pelo grande apoio. Agradeço à UTFPR pela oportunidade e infraestrutura.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas e Siglas

AA	Aplicação Ativa
AE	Ambiente de Execução
AES	Advanced Encryption Standard
API	Application Program Interface
COPS	Common Open Policy Service
DARPA	Defense Advanced Research Project Agency
DEPMA	Dynamically Extensible Policy-Based Management Architecture
DES	Data Encryption Standard
DiffServ	Differentiated Services
FGP	Ferramenta para Gerenciamento de Políticas
GC	Gerador de Chaves
GRSDD	Gerador do Repositório Seguro de Dados Direcionados
GRSL	Gerador do Repositório Somente Leitura
GTSS	Gerenciador de Tarefas de Segurança do Servidor
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IOA	Interface para Objetos Ativos
IP	Internet Protocol
LDAP	Lightweigh Directory Access Protocol
LPDP	Local Policy Decision Point
MSC	Módulo de Segurança Cliente
MSS	Módulo de Segurança Servidor
NIST	National Institute of Standards and Technology
PAFP	Protocolo Ativo para Fornecimento de Políticas
PASFP	Protocolo Ativo Seguro para Fornecimento de Políticas
PBM	Policy Based Management
PDP	Policy Decision Point

PEP	Policy Enforcement Point
QoS	Quality of Service
RAI	Repositório Ativo de Informações
RAPWG	Resource Allocation Protocol Working Group
RC	Repositório de Chaves
RNC	Repositório de Nós Confiáveis
RP	Repositório de Políticas
RSDD	Repositório Seguro de Dados Direcionados
RSL	Repositório Somente Leitura
RSPRA	Repositório Seguro de Políticas para Redes Ativas
SPDP-A	Serviço PDP Ativo
SPEP-A	Serviço PEP Ativo
SSL	Secure Sockets Layer
VRSDD	Verificador do Repositório de Dados Direcionados
VRSL	Verificador do Repositório Somente Leitura
VKS	Verification-Key Storing Server
3DES	Triple Data Encryption Standard

Resumo

A tecnologia de redes ativas aplicada ao gerenciamento de rede baseado em política provê uma excelente ferramenta de gerenciamento, pois toda flexibilidade de redes ativas permite modificações dinâmicas no comportamento da rede. Porém, a flexibilidade de redes ativas também introduz problemas de segurança. Motivado por estas questões de segurança, o presente trabalho apresenta uma arquitetura que provê segurança ao gerenciamento de redes ativas baseado em políticas, através da utilização de repositórios seguros utilizados em agentes móveis. Um protótipo da arquitetura foi implementado e um estudo sobre o impacto de processamento extra foi realizado.

Abstract

The technology of active nets applied to the management of net based on politics to provide an excellent tool with management, therefore all flexibility of active nets allows dynamic modifications in the behavior of the net. However, the flexibility of active nets also introduces security problems. Motivated for these questions of security guard, the present work presents an architecture that to provide security to the management with active nets based in politics, through the use of used safe repositories in mobile agents. An archetype of the architecture was implemented and a study on the impact of extra processing it was carried through.

1 Introdução

O gerenciamento de rede baseado em políticas *(PBM – Policy Based Management),* proposta pela IETF(*Internet Engineering Task Force*), tem o objetivo de prover automação das configurações de rede [AHN 03]. O PBM deve ser apto a administrar um grande número de usuários, aplicações e recursos para estipular suas políticas e então configurar cada elemento de rede individualmente [ZEBIANE 03]. Tal gerenciamento busca cumprir com o planejamento operacional da rede, fazendo com que usuários e aplicações preferenciais tenham acesso aos recursos da rede. O PBM tem grande importância em soluções de problemas de rede causada pela evolução das mesmas, nestes casos aumentar os recursos de rede nem sempre é o suficiente. Os objetivos da arquitetura inclui a criação de um sistema padrão que trata a aplicação e administração de políticas [ZEBIANE 03]. Restrições intrínsecas ao modelo PBM vêm a torná-lo um tanto quanto estático. Tal fato frente a uma rede onde decisões estratégicas tendem a mudar com o surgimento de novos requisitos, vem a impor uma abordagem diferenciada para concepção deste modelo, onde as entidades envolvidas, possam ser mais autônomas e suas interações ocorram de uma forma flexível [FERRASA 04].

O modelo DEPMA (Dynamically Extensible Policy-Based Management Architecture) proposto por [FERRASA 04] explora a concepção do modelo de gerenciamento baseado em políticas (PBM) utilizando-se da tecnologia de redes ativas a fim de obter vantagens em termos de flexibilidade e dinamismo no que se refere ao intercâmbio e atualização de políticas. O modelo modifica os originais elementos da arquitetura PBM (PEP e PDP) e os transforma em aplicações ativas (SPEP-A e SPDP-A) descentralizando as decisões de políticas. O RAI (Repositório Ativo de Informações), componente chave da arquitetura DEPMA, vem a fornecer meios para execução de funções de monitoramento de recursos e serviços, além disso, dentro do contexto do gerenciamento baseado em políticas, pode atuar ainda como uma entidade responsável pela organização, recuperação, instalação e exclusão de políticas e pelo suporte a inclusão de novas políticas de forma dinâmica. A arquitetura disponibiliza cápsulas ativas que fazem o transporte e inclusão de novas política *"on the fly"* na rede.

Todas as vantagens adquiridas com as redes ativas, trazem sérios problemas de segurança, vários trabalhos abordam este problema. Segundo [LIU 03], o incremento de serviços em uma rede ativa faz com que o papel de segurança desempenhado pelo sistema

operacional se torne crucial. Na visão de [ALEXANDER 01], controlar acessos a recursos de uma rede ativa requer controlar as ações de módulos carregados. [KIM 03] tratou de garantir a transferência de pacote ativo utilizando assinatura digital utilizando um servidor de armazenamento e geração de chaves. Mesmo utilizando redes ativas, a arquitetura DEPMA foi criada sem nenhum tipo de preocupação com segurança. Suas cápsulas ficam vulneráveis a adulteração por nós maliciosos, o que pode comprometer todo o comportamento da rede. Devido a suas características redes ativas e agentes móveis possuem grande semelhança, sendo que o conceito de redes ativas é mais genérico do que o paradigma de agentes móveis, o que poderia nos dar a visão de que um agente móvel poderia ser considerado como um tipo específico de um nó ativo que compõem uma rede ativa [PSOUNIS 99].

Devido a tal semelhança, pode-se observar algumas abordagens de segurança interessantes para agentes móveis, que podem ser adequadas a redes ativas. [KARNIK 98] trata o problema de segurança de agentes móveis utilizando o conceito de contêiner somente-leitura que é inserido no agente e serve para detectar modificações nos dados imutáveis deste agente. [WANGHAM 04] utilizou como base o trabalho de [KARNIK 98] e adaptou os contêineres criando os repositórios seguros para agentes móveis. Este esquema possui duas técnicas: uma para proteção do código e dos dados imutáveis do agente e outra para os resultados parciais recolhidos durante as viagens do agente. Outra técnica, que visa não só a integridade mas também a confiabilidade de alguns dados para que não sejam revelados a plataformas não autorizadas, compõe o esquema proposto (Repositório de Dados Direcionados). A solução do repositório seguro para agentes móveis poderia ser adaptada para garantir aspectos de segurança em uma rede ativa, mas especificamente, para tratar os problemas de segurança da arquitetura DEPMA, desta forma tornando a arquitetura robusta e menos vulnerável.

1.1 Objetivo

Este trabalho propõe uma arquitetura segura para garantir as vantagens da arquitetura DEPMA proposta em [FERRASA 04]. A nova arquitetura será derivada da arquitetura DEPMA, sendo que para isso, modificações em sua estrutura interna serão necessárias. Os problemas de segurança do DEPMA serão tratados utilizando-se uma adaptação, para redes ativas, do esquema proposto em [WANGHAM 04]. Será criado um ambiente de simulação para validar a nova arquitetura. Levando-se em consideração que a adição de segurança em um modelo é

realizado em detrimento do desempenho, será realizado um estudo sobre o custo de processamento adicionado no processo, em relação a arquitetura original.

1.2 Trabalhos Relacionados

Os alicerces do trabalho proposto encontram-se em dois outros trabalhos: o primeiro é [FERRASA 04] que, para dar maior flexibilidade ao gerenciamento baseado em políticas, aplicou o conceito de redes ativas no modelo PBM. Desta forma, a arquitetura proposta traz todas as vantagens de redes ativas para a disseminação e inclusão de política em uma rede *"on-the-fly"* . O fato de questões de segurança não serem tratados em [FERRASA 04], motivou a realização do trabalho proposto. O segundo trabalho é [WANGHAM 04], que adaptando o conceito de contêineres somente leitura, apresentou uma proposta de repositórios seguros para garantir a integridade de agentes móveis. Além desses dois trabalhos é de fundamental importância citar o trabalho de [KARNIK 98] que propôs o esquema de contêiner somente leitura e vetores direcionados, que foi adaptado por [WANGHAM 04].

O estudo de redes ativas, seus problemas de segurança e soluções foi fundamental para a conclusão deste trabalho. Nesta linha é importante citar: [LIU 03] que propõe um modelo que provê segurança a nível de sistema operacional do nó. [YANAN 02] trata a segurança ao nível de cápsula. Seu trabalho consiste na idéia de que cada cápsula tem de passar por verificações de segurança antes de se tornar apta a executar. Para isso é necessária a construção de uma nova cápsula, com mais campos no cabeçalho que contém informações relacionadas à segurança. [KIM 03] armazena chaves de verificação de nós autorizados em um VKS (*Verification-Key Storing Server*). Quando um nó autorizado requisita uma chave para verificar uma assinatura, este servidor envia a chave após checar se o nó é um nó autorizado.

A solução apresentada neste trabalho adapta o esquema proposto por [WANGHAM 04] para ser aplicado na arquitetura proposta por [FERRASA 04], ou seja, faz a adaptação do esquema de repositórios seguros utilizados em agentes móveis para resolver problemas de segurança na troca de políticas de redes ativas.

1.3 Organização do Trabalho

O primeiro capítulo deste trabalho abordou as motivações deste trabalho, o objetivo da arquitetura proposta e os trabalhos relacionados.

O segundo capítulo aborda o conceito de redes ativas e gerenciamento baseado em políticas. Apresenta a arquitetura DEPMA que une esses dois conceitos, ou seja, aplica o conceito de redes ativas no gerenciamento baseado em políticas e mostra os problemas de segurança que não foram tratados.

O terceiro capítulo aborda mecanismos de segurança como criptografias simétricas e assimétricas. Exibe soluções de segurança implementadas para agentes móveis utilizando os contêineres somente leitura e o esquema de repositório seguro para agentes móveis.

O quarto capítulo apresenta a união dos conceitos abordados nos dois capítulos anteriores e exibe a arquitetura proposta para atingir o objetivo deste trabalho.

O quinto capítulo apresenta os detalhes de implementação da nova arquitetura, o ambiente de testes e os resultados colhidos.

O sexto capítulo apresenta as conclusões deste trabalho. São abordadas questões relativas aos resultados alcançados, aos problemas solucionados e as limitações do trabalho que podem motivar trabalhos futuros.

2 Redes Ativas e o Gerenciamento Baseado em Políticas

2.1 Introdução

Segundo [AHN 03], políticas consistem de uma ou mais regras que descrevem a ação que ocorrerá quando uma condição específica é verdadeira. Políticas são utilizadas para controlar o comportamento de uma rede através de tomada de decisões e ações. [SLOMAN 98] define políticas como sendo regras que governam o comportamento de um sistema.

Políticas servem para permitir a implementação, monitoração e modificação dos diversos perfis associados aos usuários ou grupos de usuários que utilizam serviços da rede. Servem para disciplinar e otimizar a utilização dos recursos de rede e para garantir qualidade de serviço, em situações onde o tráfego que atravessa a rede seja superior a largura de banda disponível [MENEZES 05].

A comunidade de pesquisa DARPA (*Defense Advanced Research Project Agency*), em discussões sobre as direções futuras dos sistemas de rede, levantou inúmeros problemas nas redes convencionais. A dificuldade de integração de novas tecnologias e padrões dentro da infra-estrutura de rede, baixa performance devido às operações redundantes das várias camadas de protocolos e dificuldade de introduzir novos serviços no modelo estrutural existente foram algumas dos problemas encontrados. Para resolver estes problemas foi criado o conceito de redes ativas [TENNENHOUSE 97].

O paradigma de Redes Ativas oferece uma infra-estrutura de comunicação flexível, que permite modificações dinâmicas no comportamento da rede. As características apresentadas por estes modelos em favor de um maior controle, flexibilidade e dinamismo da rede, fornece excelentes ingredientes para a aplicação do gerenciamento baseado em políticas em redes ativas.

Neste capítulo será apresentado o modelo de gerenciamento de redes baseado em política proposto pela IETF, o conceito de redes ativas e a junção dos dois conceitos, dando origem a arquitetura DEPMA. Os problemas de segurança da arquitetura DEPMA são os objetos de estudo deste trabalho.

2.2 Gerenciamento de Redes Baseado em Políticas (Modelo IETF)

O gerenciamento de rede baseado em políticas (PBM) tem o objetivo de prover automação das configurações de rede [AHN 03]. O PBM deve ser apto a administrar um grande número de usuários, aplicações e recursos para estipular suas políticas e então configurar cada elemento de rede individualmente [ZEBIANE 03]. Tal gerenciamento busca cumprir com o planejamento operacional da rede, fazendo com que usuários e aplicações preferenciais tenham acesso aos recursos da rede, além disso tem grande importância em soluções de problemas de rede causados pela evolução das mesmas, nestes casos aumentar os recursos de rede nem sempre é o suficiente.

Os objetivos da arquitetura PBM inclui a criação de um sistema padrão que trata a aplicação e administração de políticas. Especificamente, o sistema deverá desenvolver varias funções incluindo [ZEBIANE 03]:

1. **Administrador de Políticas:** criar e distribuir sistemas hierárquico para coordenar e gerenciar políticas entre dispositivos gerenciadores, diretórios e vários dispositivos de aplicação de políticas, usando a arquitetura padrão proposta.
2. **Interface para um Banco de Dados (diretório):** provê acesso de alto nível, informações de políticas de usuário e nível de aplicação via dados armazenados em um repositório público.
3. **QoS Gateway:** Provendo aplicação e gerenciamento de políticas fim-a-fim via sinal padrão, provisionando protocolos incluindo *DiffServ*.
4. **Corretor de Tamanho de Banda:** provendo corretagem de serviços de tamanho de banda, permitindo um sistema automático para múltiplos domínios para negociar nível de garantia de serviço.
5. **Monitoramento e Estimativas Centralizadas:** provê contabilidade centralizada baseada em políticas e serviços de monitoramento remoto.

A IETF criou um grupo de estudos para a pesquisa de gerenciamento baseado em políticas, tal grupo possui o nome de RAPWG (*Resource Allocation Protocol Working Group*) [RAP 05]. Segundo [YAVATKAR 00], os dois principais componentes da arquitetura da IETF

são o PEP (*Policy Enforcement Point*) e o PDP (*Policy Decision Point*). A figura 2.1 apresenta uma configuração básica envolvendo estes dois elementos:

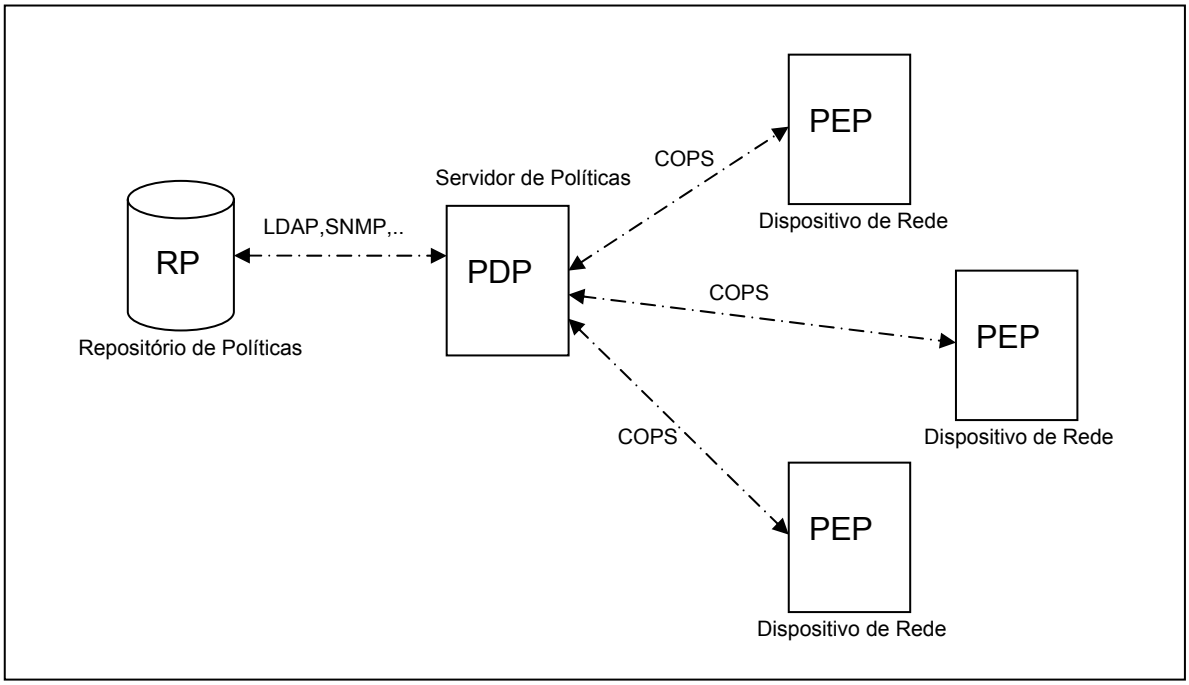


Figura 2-1: *Modelo básico do modelo proposto pelo RAPWG*

O PEP é o elemento situado nos dispositivos de rede (nós) ele tem a função de aplicar as decisões de políticas. O PDP é o elemento que toma todas as decisões de políticas, geralmente se situa em um servidor de políticas tomando as decisões que são enviadas e aplicadas pelo PEP. O PDP utilizar mecanismos adicionais e protocolos para executar funcionalidades adicionais como autenticação de usuários, armazenamento de informações de políticas, etc...

O repositório de políticas (RP) é uma espécie de armazém onde as políticas são estocadas. O repositório de políticas pode ser criado no formato de um serviço de diretório. A IETF sugere o uso do LDAP (*Lightweigh Directory Access Protocol*), apresentado em [HOWES 97], pelo PDP para fazer acesso ao repositório de políticas. Outros protocolos pode ser utilizados pelo PDP para fazer acesso ao RP, como o HTTP, SNMP e SSH, etc...

[YAVATKAR 00] descreve a interação entre os elementos PEP e PDP da seguinte forma:

1. O PEP receberá uma notificação ou uma mensagem que necessita de uma decisão de política.
2. O PEP então formula uma requisição para uma política de decisão e envia para o PDP. Esta requisição pode conter um ou mais elementos de políticas (encapsulados em um ou mais objetos de políticas).
3. O PDP retorna a política de decisão para o PEP que aplica a política, aceitando ou rejeitando a requisição.

O PDP pode ainda retornar informações adicionais ao PEP as quais incluem um ou mais elementos de políticas. A configuração apresentada na figura 2.1 pode não ser suficiente para aplicar políticas locais. Para contornar esta limitação, a figura 2.2 apresenta uma variação da configuração apresentada anteriormente. Nesta nova configuração, o PDP pode estar localizado no mesmo dispositivo de rede que o PEP.

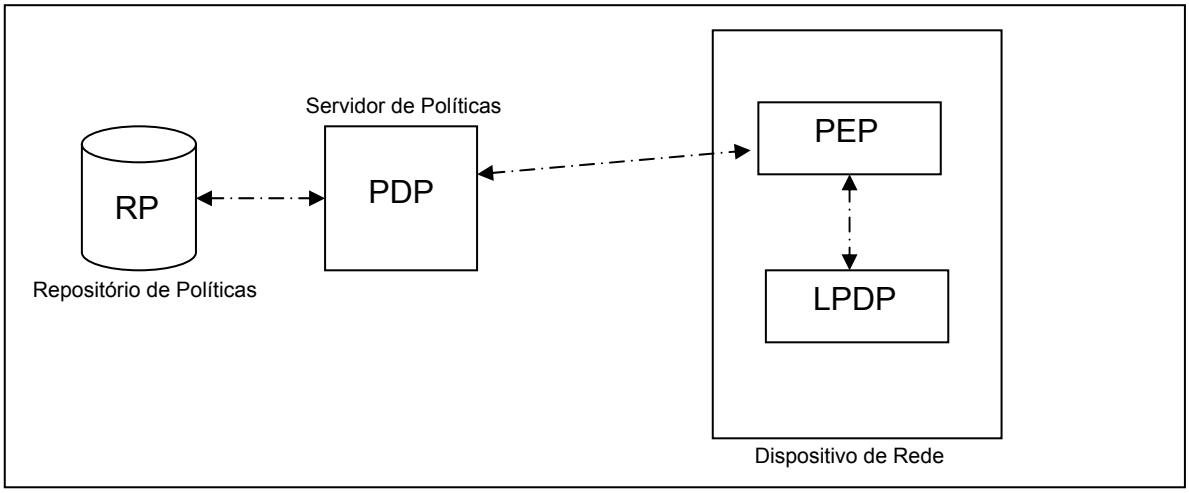


Figura 2-2: *Modelo que apresenta o PDP e o PEP localizados no mesmo dispositivo de rede*

O modelo apresentado, mostra o PEP e o PDP em um mesmo dispositivo de rede, este PDP recebe o nome de LPDP *(Local Policy Decision Point)*. A interação do modelo se ocorre da seguinte forma:

1. O PEP receberá uma notificação ou uma mensagem que necessita de uma decisão de política.
2. O PEP utiliza o LPDP para obter as decisões de política local (decisões parciais).

3. A decisão de política local e a requisição de política original são enviados ao PDP localizado no servidor de políticas.
4. O PDP tomará a decisão final e poderá sobrepor a decisão tomada pelo LPDP.
5. O PEP aplicará a decisão tomada pelo PDP.

O PDP pode, a qualquer momento, enviar uma notificação assíncrona para o PEP modificando um decisão tomada anteriormente ou para gerar uma mensagem de erro.

O modelo IETF estabelece o COPS *(Common Open Policy Service)* [BOYLE 00] como protocolo para transferência de políticas de decisão entre o PDP e PEP. Porém o modelo é aberto para a utilização de outros tipos de protocolos como HTTP, FTP ou SNMP.

O COPS é um protocolo proposto para troca de informações de políticas de rede entre os pontos de decisões de políticas (PDP) e os pontos de aplicação de políticas (PEP) como parte do QoS *(Quality of Service)* completo. Este protocolo foi desenvolvido em total oposição ao protocolo SNMP, o qual mostra-se não adequado à arquitetura PBM. O COPS possui três níveis distintos: o protocolo básico, as diretivas de uso para o tipo de cliente e a representação do comportamento dos dados. O modelo usado pelo protocolo COPS supõe que cada domínio administrativo tenha ao menos um servidor de políticas [ZEBIANE 03].

2.3 Redes Ativas

Entre 1994 e 1995 a comunidade de pesquisa DARPA (*Defense Advanced Research Project Agency*), em discussões sobre as direções futuras dos sistemas de rede, levantou inúmeros problemas nas redes convencionais. A dificuldade de integração de novas tecnologias e padrões dentro da infra-estrutura de rede, baixa performance devido às operações redundantes das várias camadas de protocolos e a dificuldade de introduzir novos serviços no modelo estrutural existente, foram algumas dos problemas encontrados. Para resolver estes problemas foi criado o conceito de redes ativas [TENNENHOUSE 97].

Segundo [LIU 03], Redes Ativas vem com o objetivo de prover um *framework* que habilita a customização do processo de comunicação das aplicações de rede. [SMITH 04] diz que o objetivo de redes ativas é criar tecnologia de rede que, ao contrário das redes atuais, sejam de fácil desenvolvimento e que permitam a customização de aplicações específicas.

Redes ativas tornam possível o desenvolvimento e inclusão de novos serviços dentro da rede *"on-the-fly"*. Este objetivo é alcançado através do processamento de pacotes ativos que carregam código (programas) que são executados nos nós ativos da rede. Segundo [GUO], em uma rede ativa, informações injetadas dentro da rede podem ser modificadas ou redirecionadas durante seu transporte. A tecnologia de redes ativas pode ser usada para estender rapidamente os serviços dentro da rede.

Segundo [TENNENHOUSE 97], redes ativas são ativas no sentido de que os nós podem executar e modificar o conteúdo de um pacote. O processamento pode ser customizado por usuário ou por aplicação. Este código é transportado pela rede dentro de pacotes chamados de cápsulas. Estas cápsulas transportam, dentro de si, códigos (programas) que são executados dentro de nós que possuem suporte para executar tais códigos, estes nós são chamados de nós ativos.

[FERRASA 04] diz que de uma forma geral, uma rede ativa pode ser concebida como um conjunto de nós, onde entre estes, existem os nós chamados ativos, os quais podem realizar processamentos adequados ao fluxo de dados que passam através destes. Tais processamentos não se restringem somente em prol da entrega de dados, e podem conter funções específicas fornecidas pelo usuário. Em redes ativas os roteadores e/ou *switches* executam computação customizado nas mensagens que passam através deles. Estes roteadores ativos podem também operar com roteadores legados, com datagramas da maneira tradicional [TENNENHOUSE 97].

Arquiteturalmente, redes ativas possuem abordagens distintas no que diz respeito a forma de execução dos códigos ativos, nos próximos itens serão apresentadas duas abordagens, a discreta e a integrada.

2.3.1 Abordagem Discreta

O processamento de mensagem pode ser arquiteturalmente separado da inclusão de programas dentro da rede, com mecanismos separados para cada função. Usuários injetam suas rotinas de processamento dentro dos roteadores e depois enviam seus pacotes com dados, através dos nós programáveis, para serem processados. Quando o pacote chega em um nó, seu cabeçalho é examinado e o programa apropriado é despachado para executá-lo. No contexto discreto, programas que são inseridos em um nó ativo não são enviados de forma integrada com os dados que sofrerão algum tipo de processamento [TENNENHOUSE 97],[FERRASA 04].

Mecanismos separados de carga e execução se tornam interessantes quando o programa carregado deve ser cuidadosamente controlado ou quando os programas são grandes. Permitir à administradores de rede carregar dinamicamente seus códigos dentro de roteadores é bastante útil para o propósito de extensão das funcionalidades do roteador [TENNENHOUSE 97].

2.3.2 Abordagem Integrada

A visão mais extrema de redes ativas é a que considera que toda mensagem é um programa. Estas mensagens (cápsulas) que passam através dos nós, contêm um fragmento de código que pode ou não conter dados embutidos. Quando uma cápsula chega em um nó ativo, seu código é interpretado. Cada nó ativo possui um mecanismo para ler o código encapsulado, um ambiente que permite a execução do código e em repositório para armazenamento de cápsulas [WETHERALL 96].

2.4 Modelo Arquitetural de Redes Ativas

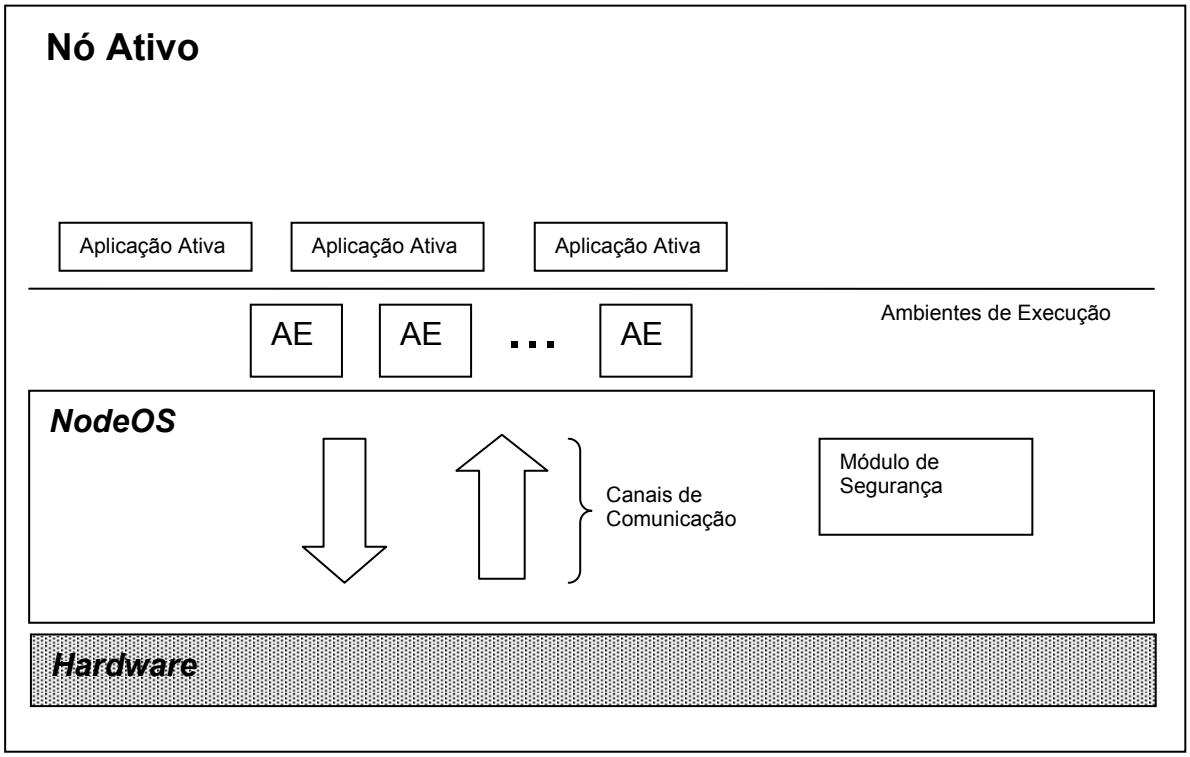


Figura 2-3: *Modelo Arquitetural de Redes Ativas*

O NodeOS é similar a um *kernel* tradicional de sistema operacional, e tem a função de gerenciar e alocar recursos do nó. Funciona como uma camada entre os ambientes de execução (AE's) e os recursos do nó, fornecendo as funções básicas a partir das quais os AE's constroem abstrações que formam a API de rede. Ele gerencia os recursos do nó ativo e escalona a demanda por esses recursos incluindo a transmissão, o processamento e o armazenamento de dados. O NodeOS isola os AE's dos detalhes de gerência de recursos, e também da existência de outros AE's.

Além disso, o NodeOS implementa canais de comunicação, sobre os quais os AE's enviam ou recebem pacotes. Estes canais consistem de *links* de transmissão físicos e o processamento do protocolo associados com as camadas de mais alto nível (como o IP). Quando um nó ativo recebe um pacote vindo de um *link* físico, ele classifica esse pacote baseando-se no conteúdo do mesmo, cada pacote ou é associado a um canal existente ou é descartado. O mapeamento dos pacotes que chegam aos canais é controlado por meio de um padrão especificado pelo AE desde que ele cria o canal. Um AE requisita a criação de um canal para aqueles pacotes que seguem um certo padrão de cabeçalho, como uma combinação do protocolo IP com os números de porta TCP. Toda requisição de alocação de recurso feita por um AE é analisada pelo módulo de segurança localizado no NodeOS.

Qualidade de serviço é fornecida pelo NodeOS através de mecanismos de escalonamento que controlam o acesso aos recursos de computação e de transmissão do nó. Esses mecanismos isolam o tráfego de usuários dos efeitos causados pelo tráfego gerado por outros usuários, de modo que cada um deles parece ter a sua própria máquina ou *link* virtual. Quando os canais são criados, o AE requisitante especifica, para o escalonador, qual é o tratamento desejado [CALVERT 99].

Segundo [CALVERT 99], um ambiente de execução define uma máquina virtual e uma interface que pode ser acessada para enviar as instruções codificadas apropriadas para o AE em pacotes. A função desta máquina virtual não é definida pela arquitetura, o NodeOS fornece um grupo de funções que podem ser usadas pelos AE's para implementa uma maquina virtual. Alguns AE's implementam uma máquina virtual universal, enquanto outros oferecem interfaces mais restritas.

Uma aplicação ativa (AA) é um programa que, quando executado por uma máquina virtual, implementa um serviço fim-a-fim. Uma AA implementa serviços customizados para aplicações de usuários finais, usando a interface fornecida pela AE. O AE determina como o código da AA é carregado para dentro do nó, o código pode ser carregado com a AA (*in-band*) ou carregado sobre demanda (*out-of-band*). A carga *out-of-band* poderá ocorrer durante fases distintas ou por demanda [CALVERT 99]. Esta carga sob-demanda pode ocorrer automaticamente quando um método que não está presente no nó é invocado ou por invocação explicita. No caso da chamada de um método não localizado no nó, o código ativo é solicitado para o nó anterior ou para um servidor de códigos ativos (depende da implementação).

2.5 Gerenciamento de Redes Ativas Baseado em Políticas – Arquitetura DEPMA

A flexibilidade do Modelo de Gerenciamento Baseado em Políticas o torna apropriado para ser integrado a diferentes tecnologias. Uma destas integrações, proposta em [FERRASA 04], é realizada com redes ativas, o modelo resultante desta união denomina-se DEPMA. O modelo DEPMA (*Dynamically Extensible Policy-Based Management Architecture*) explora a concepção do modelo de gerenciamento baseado em políticas utilizando-se da tecnologia de redes ativas a fim de obter vantagens em termos de flexibilidade e dinamismo no que se refere ao intercâmbio e atualização de políticas. O DEPMA é baseado no modelo de gerenciamento baseado em políticas da IETF.

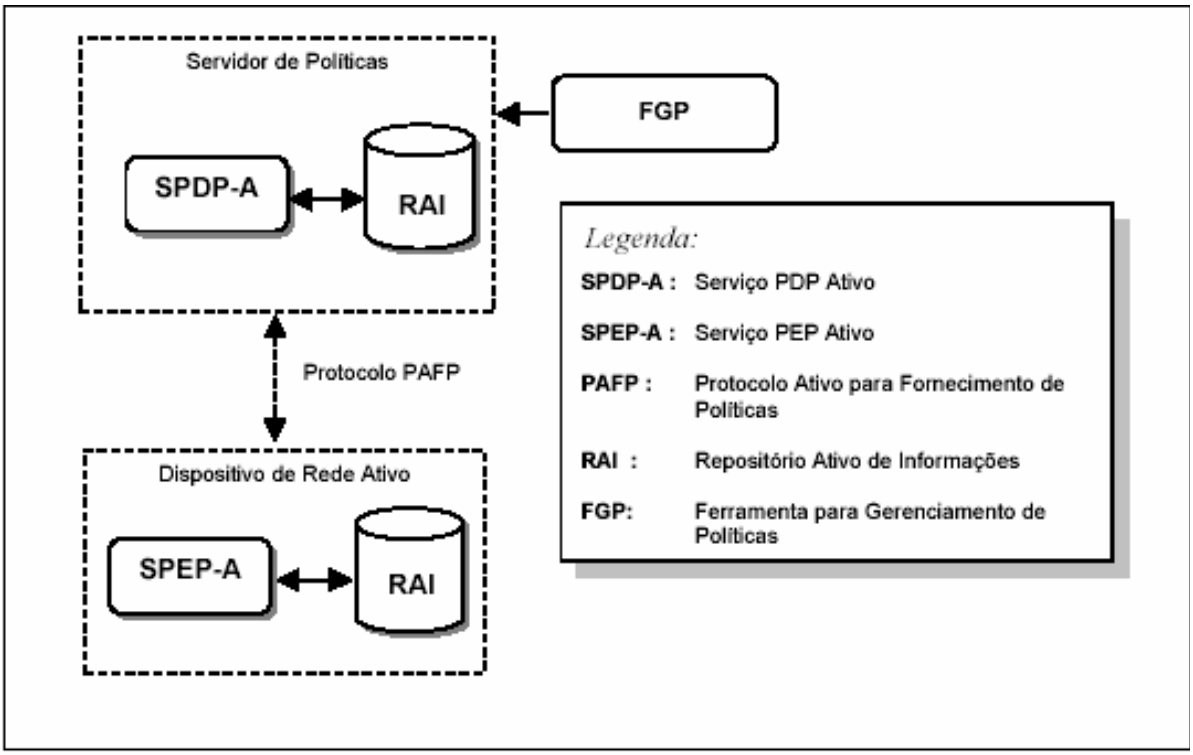


Figura 2-4: *Arquitetura DEPMA*

A arquitetura DEPMA é compostos dos seguintes elementos:

- SPEP-A (Serviço PEP Ativo): responsável pela execução das decisões tomadas e juntamente com o RAÍ definem o comportamento da arquitetura. Ele possui uma interface com o repositório de dados (IRAI), uma interface de comunicação com o protocolo PAFP (IPAFP) e um mecanismo para manipulação de políticas (MMP), o qual é responsável pela interpretação das informações relativas a políticas e posterior iniciação do processo de instalação ou exclusão de políticas.
- SPDP-A: é responsável pela geração de eventos relacionados a políticas e possui interface de comunicação com o protocolo PAFP (IPAFP), uma interface de comunicação com o FGP (IFGP) e um mecanismo para decisões baseadas em políticas (MDP), que é responsável pela interpretação e tomada de decisões baseadas em políticas.
- FGP - Ferramenta para Gerenciamento de Políticas: este elemento deve possuir uma interface de comunicação com o SPDP-A e uma interface responsável pela

comunicação homem-máquina. Ele é responsável pela manutenção do repositório de políticas, armazenando e alterando políticas existentes no repositório.

- PAFP – Protocolo Ativo para Fornecimento de Políticas: oferece meios para o intercâmbio de políticas entre o SPDP-A e o SPEP-A de maneira flexível. É formado por um conjunto de cápsulas ativas, as quais realizam todas as interações necessárias para a execução do processo de intercâmbio de informações referente a políticas.
- RAÍ – Repositório Ativo de Informações: fornece meios para execução de funções de monitoramento de recursos e serviços, e pode atuar como uma entidade responsável pela organização, recuperação, instalação e exclusão de políticas e pelo suporte a inclusão de novas políticas de forma dinâmica. Basicamente é o repositório de políticas do modelo.

A troca de políticas entre o SPDP-A e o SPEP-A é realizada através do protocolo PAFP (Protocolo Ativo para Fornecimento de Políticas). Este protocolo é formado por um conjunto de cápsulas ativas as quais realizam todas as interações necessárias para a execução do processo de intercâmbio de informações referentes a políticas. O PAFP é dividido em funcionalidades dinâmicas e comuns. As funcionalidades comuns dizem respeito à emissão de pedidos, atualizações e exclusão de políticas em clientes de políticas. As funcionalidades dinâmicas realizam o papel de prover principalmente os meios necessários para a extensão e a inclusão dinâmica de novas políticas em clientes de políticas.

O RAI implementa uma interface para objetos ativos (IOA). Um objeto que implementa a IOA, pode ser enviado diretamente para um nó gerenciado, sem a necessidade de interromper qualquer processo sendo executado no nó destino. Cada nó ativo possui uma API para o RAI a qual disponibiliza diversas funções para gerenciamento. Os nós executam os objetos que implementam a interface IAO, tais objetos são chamados de Objetos Ativos – OA. O envio de novas funcionalidades é realizado através de cápsulas ativas. O esquema básico de interação é realizado da seguinte forma:

1. Uma aplicação ativa encapsula informações ou um OA em uma cápsula ativa e envia ao nó de destino
2. Chegando ao nó destino, a cápsula ativa acessa a API do RAÍ e pode realizar chamadas aos métodos exportados.

3. Quando um novo OA é enviado a um nó ele é instanciado, e a partir daí pode desempenhar as funções a que foi designado.

O modelo RAÍ implementa um gerenciamento distribuído (descentralizado) onde cada nó possui o seu RAÍ para tomada de decisões.

2.5.1 Interação da Arquitetura

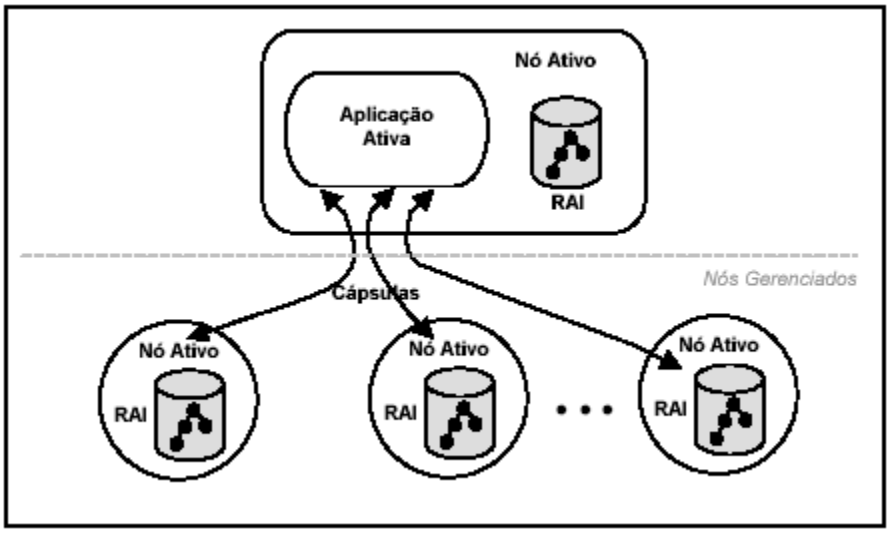


Figura 2-5: *Esquema básico de interação*

A interação na arquitetura acontece entre as entidades SPEP-A, SPDP-A, RAÍ e FGP. A interação entre SPEP-A e SPDP-A pode ser dividida em interação iniciada pelo dispositivo de rede e interação iniciada pelo servidor de políticas.

O processo de interação iniciado pelo dispositivo de rede é realizado da seguinte forma:

- O SPDP-A (aplicação ativa), realizada uma chamada ao método *register* do RAI situado no mesmo nó (registra-se no RAÍ local).
- SPDP-A obtém do RAÍ uma referência para as classes que representam funcionalidades a serem utilizadas pela arquitetura.
- Através das referências obtidas, o SPDP-A pode inserir AO's, os quais representam políticas.

- Após inicializado o servidor, o SPEP-A realiza uma chamada ao RAÍ situado no mesmo nó (registra-se no RAÍ local).
- SPEP-A obtém do RAÍ uma referência para as classes que representam funcionalidades utilizadas pelo dispositivo de rede.
- SPEP-A envia ao SPDP-A uma mensagem (através do protocolo PAFP) indicando que está ativo.
- O SPDP-A envia uma mensagem indicando que aceita ou rejeita a conexão (utilizando protocolo PAFP).
- SPEP-A envia para o SPDP-A pedido de configuração de políticas.
- SPDP-A envia ao SPEP-A uma mensagem que contém todas as informações relativas a configuração requisitada pelo SPEP-A (cápsulas ativas).
- Ao receber as informações (objetos ativos - OA) o SPEP-A insere tais configurações no RAÍ local. Tais OA's se tornam ativos e ocorre a instalação destas políticas.

A interação iniciada pelo servidor de políticas SPDP-A é realizada da seguinte forma:

- SPDP-A acesso o RAÍ local e obtém uma políticas a ser analisada.
- Tomada a decisão, o SPDP-A envia uma mensagem ao SPEP-A com informações referentes as decisões (cápsulas ativas).
- Com o recebimento das informações, o SPEP-A insere no RAÍ local os objetos representando as políticas enviadas. Estes objetos tornam-se ativos e ocorre a instalação/remoção destas políticas.

O servidor de políticas pode ainda delegar a tomada de decisões para os RAI's dos nós, descentralizando o processamento. As interação que mostram a delegação de tomada de decisões são:

- Quando usuário inclui/altera uma funcionalidade da rede através do FGP, este colhe as informações e envia para o SPDP-A.
- O SPDP-A salva as informações no seu RAÍ local.
- O SPDP-A instancia a nova funcionalidade.

- Caso o SPDP-A não tenha se cadastrado ao RAÍ destino, ele o faz para poder acessá-lo.
- Após fazer o cadastro, o SPDP-A envia uma cápsula ativa ao nó destino a fim de cadastrar as novas funcionalidades.
- Tendo salvo as informações, o RAÍ notifica o SPEP-A da chegada da nova funcionalidade.
- Após receber a notificação do RAÍ, o SPEP-A torna ativa a nova funcionalidade.

2.6 Considerações Gerais

Segundo a sua própria definição, redes ativas permitem que seus nós possam executar e modificar o conteúdo de um pacote (cápsula ativa). Neste sentido, o nó ativo possui controle total sobre a cápsula durante a sua execução. Esta flexibilidade, característica de uma rede ativa, expande a possibilidade de causar algum dano. Ataques de recusa de serviço são possíveis contra vários recursos da rede: como ciclos de CPU, armazenamento e tamanho de banda, os quais são usados para carregar programas. Controlar acessos a recursos de uma rede ativa requer controlar as ações de módulos (cápsulas) carregados.

Os problemas de segurança podem ser classificados da seguinte maneira:

- Acesso não Autorizado: um pacote ativo pode acessar e destruir informações importantes de algum nó da rede.
- Captura de Informações: um pacote poderia roubar informação de um nó ativo e vice-versa.
- Sabotagem: um pacote ativo poderia programar a rede de maneira a prejudicar seu desempenho.
- Negação de Serviço: um pacote poderia utilizar mais recursos do nó fazendo com que este passasse a negar serviços por falta de recursos. Outra maneira de provocar a negação de serviço seria desviar todo o tráfego para um único nó, fazendo com que este ficasse sobrecarregado e negasse os serviços solicitados.
- Modificação de Informações: um nó ativo mal intencionado poderia modificar o conteúdo de um pacote ativo e enviá-lo para seu destino.

A vulnerabilidade de redes ativas se dá na transmissão dos dados e na execução da cápsula, desta forma o modelo de gerenciamento baseado em políticas para redes ativas possui problemas de segurança quando do envio de políticas para os nós de aplicação destas políticas, este problemas podem acontecer tanto no modo *Outsourcing* (onde o PEP solicita as políticas ao PDP) quanto no modo *Provisioning* (onde o PDP toma as decisões e então informa ao PEP da configuração a ser utilizada). Desta forma podemos listar alguns problemas de segurança na arquitetura DEPMA, as ameaças listadas estão separados em duas categorias, execução e transmissão:

Acesso não Autorizado: Uma cápsula ativa gerada por um nó mal intencionada poderia remover, do repositório dos nós ativos (clientes), todos os objetos de política. Isso é possível se um nó mal intencionado se passar pelo servidor de políticas e enviar uma requisição de remoção de determinadas políticas aos seus nós (ocorre durante a execução no modo *Provisioning*).

Adulteração de Informação: O conteúdo da cápsula pode ser modificado por um nó mal intencionado, modificando políticas enviadas pelo servidor aos nós ativos. Isso é possível pois enquanto um ambiente de execução (nó ativo) está executando o conteúdo de uma cápsula, o nó tem total controle sobre as informações de tal cápsula (ocorre durante a transmissão nos modos *Provisioning* e *Outsourcing*).

Roubo de Informação: Dados podem ser lidos (obtidos) dos RAI's por um nó mal intencionado que se passa por um nó ativo confiável da rede. Um nó mal intencionado poderia se autenticar no RAÍ do servidor de políticas e enviar um pedido de configuração de políticas para o servidor, e desta forma obter todas as informações de políticas que o servidor passa aos seus nós (ocorre principalmente durante a transmissão no modo *Outsourcing*).

Mascaração: Um PEP mal intencionado, se passando por um PEP confiável, poderia solicitar ao PDP a exclusão de uma política que não utiliza mais (ocorre durante a execução no modo *Outsourcing*).

A arquitetura DEPMA foi especificada sem a preocupação com segurança. A proposta deste trabalho trata basicamente dos problemas de segurança descritos acima, que envolvem o gerenciamento de redes ativas baseado em políticas. No próximo capítulo serão apresentados os aspectos de segurança que deram embasamento para a solução proposta neste trabalho.

3 Mecanismos de Segurança

3.1 Introdução

Segurança é uma constante preocupação quando se trata de redes, esta preocupação se torna ainda maior quando o assunto é redes ativas. Existem vários trabalhos focados em segurança de redes ativas. Segundo [LIU03], as pesquisas em redes ativas se dividem em duas categorias diferentes. A primeira categoria trata de noções ditas tradicionais dentro de segurança, e inclui autenticação, controle de acesso, criptografia entre outros. A segunda categoria é relacionada com segurança do ambiente móvel, proteção dos nós de códigos móveis originados em domínios estranhos e proteção de pacotes ativos ou códigos de nós maliciosos.

O paradigma de rede ativa possui semelhanças com agentes móveis. Segundo [PSOUNIS 99], o conceito de redes ativas é mais genérico do que o paradigma de agentes móveis. Redes ativas supõe o plano de rede como um conjunto de nós ativos que podem executar processamento, e um conjunto de pacotes ativos que podem transportar código que são entendidos realmente como programas. Sendo assim, um agente móvel pode ser tratado como um tipo específico de pacote ativo. A maior diferença entre os dois paradigmas está no fato de que um agente móvel é executado como aplicação enquanto redes ativas tem seu processamento no nível de camada de rede [PSOUNIS 99]. Devido a esta semelhança, podemos entrar no universo de agentes móveis para encontrar soluções de segurança que podem ser aplicáveis à redes ativas.

Neste capítulo serão apresentados métodos de segurança aplicados em agentes móveis e mecanismos de criptografia que serão utilizados na solução proposta por este trabalho.

3.2 Criptografia

Criptografia é o processo de codificar mensagens de tal maneira a esconder o seu conteúdo. A criptografia garante confidencialidade, integridade dos dados e autenticação.

- **Confidencialidade:** Os algoritmos de criptografia (que usam chaves de criptografia) são usados para converter texto sem formatação em texto codificado e o algoritmo de decodificação equivalente é usado para converter o texto codificado em texto sem

formatação novamente. Os algoritmos de criptografia simétricos usam a mesma chave para a codificar e decodificação, enquanto que os algoritmos assimétricos usam um par de chaves pública/privada, sendo que uma codifica e a outra decodifica a mensagem.

- **Integridade de Dados:** Para garantir que os dados sejam protegidos contra modificação acidental ou mal-intencionada. A integridade, geralmente, é fornecida por códigos de autenticação de mensagem ou *hashes*. Um valor de *hash* é um valor numérico de comprimento fixo derivado de uma seqüência de dados. Os valores de *hash* são usados para verificar a integridade dos dados enviados por canais não seguros. O valor do *hash* de dados recebidos é comparado ao valor do *hash* dos dados, conforme eles foram enviados para determinar se foram alterados.
- **Autenticação:** Criptografia é utilizada como mecanismo para autenticar a comunicação entre dois nós. Um nó que decodifica uma mensagem usando uma chave particular, pode assumir que a mensagem é autentica se ela contém o *checksum* correto ou algum outro valor esperado. Ele pode deduzir que o remetente da mensagem possui a chave de codificação correspondente e então deduz a identidade do remetente se a chave é conhecida somente pelos dois nós.

Uma chave é uma seqüência de bits, e quanto maior o número de bits desta seqüência maior será a segurança obtida, ou seja, a segurança é diretamente proporcional ao tamanho da chave. Se a mesma chave for utilizada para codificar e decodificar uma mensagem, a criptografia é chamada simétrica. Se as chaves de codificação e decodificação forem diferentes, a criptografia é chamada de assimétrica.

O problema da criptografia de chaves simétricas é definir como vamos enviar ao destinatário, de forma segura, a chave secreta para a decodificação. Apesar disso, a criptografia simétrica é bastante eficiente em conexões seguras na Internet, onde processos computacionais trocam senhas temporárias para algumas transmissões críticas (SSL - *Secure Sockets Layer* funciona à base de criptografia simétrica).

Os algoritmos de criptografia assimétrica são baseados em funções matemáticas complexas, e necessitam de duas chaves diferentes para codificar e decodificar. Neste caso são utilizadas duas chaves distintas, uma pública e outra privada. Qualquer nó que possui a chave

pública tem a capacidade de codificar uma mensagem, porém somente o nó que detém a chave privada tem a capacidade de decodificá-la.

O algoritmo RSA, proposto por [RIVEST 78], permite que as chaves pública e privada, sejam utilizadas para codificar e decodificar uma mensagem, possibilitando a autenticação do emissor. O algoritmo RSA é baseado no uso do produto de dois números primos grandes (10^{100}), assim a RSA conta com o fato que a definição de fatores primos de números tão grandes é computacionalmente difícil, tornando praticamente impossível de computar.

Como a criptografia assimétrica possui um custo computacionais elevado em relação a criptografia simétrica (por ser baseada em funções matemáticas complexas), o uso de uma criptografia híbrida, que aproveita as vantagens das duas técnicas, é amplamente utilizada [COLOURIS 00].

O uso da criptografia híbrida consiste em utilizar a criptografia assimétrica para codificar as chaves secretas que irão juntamente com uma mensagem para o destinatário. O destinatário recebe a mensagem e decodifica a chave secreta de maneira segura. A partir deste ponto, todas as mensagem codificadas que serão trocadas pelas partes serão codificadas e decodificadas por esta chave secreta (criptografia simétrica).

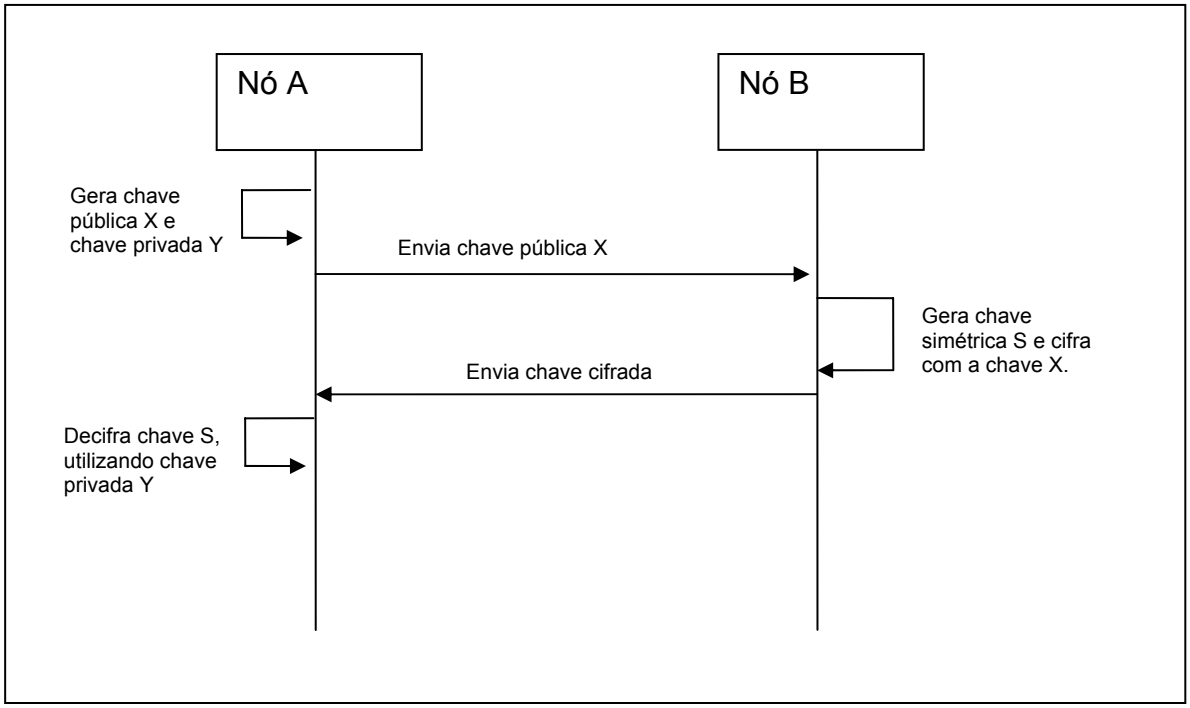


Figura 3-1*: Dinâmica da criptografia híbrida.*

A figura 3.1, mostra um exemplo de utilização da criptografia híbrida onde o Nó A gera a chave pública X e a chave privada Y. A chave pública é enviada para o nó B que gera uma chave simétrica S (que servirá para codificar e decodificar as mensagens que os dois nós trocarão a partir deste momento), e a cifra com a chave X. Desta forma somente o nó A, que possui a chave privada Y, conseguirá decifrar a mensagem e obter a chave S. Este esquema trata o problema de distribuição de chave secreta que o esquema de chave simétrica possui.

3.3 Agentes Móveis e o Contêiner Somente Leitura

Segundo [KARNIK 98], um agente móvel é um programa capaz de migrar livremente pelos nós da rede para executar alguma computação em benefício do usuário. Esta mobilidade de mover o agente em um sistema distribuído permite o desenvolvimento de serviços e aplicações mais flexíveis e dinâmicas quando comparado ao paradigma cliente-servidor.

Um agente móvel pode ser definido como um agente de software que migra de um *host* para outro em uma rede heterogênea se executando de maneira autônoma no seu destino. A

utilização de agentes móveis requer a necessidade de um ambiente computacional nas máquinas onde esses códigos irão ser executados. Ataques de plataformas maliciosas contra os agentes são os problemas de segurança mais difíceis de serem contornados e ainda sem solução adequada [WANGHAM 04].

Em transito, um agente é vulnerável a ataques de vários tipos, ele pode ser interceptado e adulterado por nós maliciosos, assim como os pacotes que trafegam em uma rede ativa. Outro tipo de ataque é realizado pelo servidor do agente. Para o agente ser executado, seu estado é aberto ao servidor que se for malicioso poderá adulterar o agente [KARNIK 98].

Tendo em vista tais ameaças [KARNIK 98] propôs um esquema para solucionar tais problemas de segurança nos agentes móveis. Esta solução consiste na definição de um contêiner somente-leitura que será inserido no agente e que servirá para detectar modificações nos dados imutáveis deste agente.

3.3.1 O Mecanismo do Contêiner Somente-Leitura

Muitas vezes, um agente móvel contém alguns itens somente leitura como parte de seus estados. Por exemplo, as credenciais de um agente poderiam não ser modificada por qualquer outro que não o seu dono, e desta forma são somente-leitura durante sua viagem. Similarmente, um agente de entrega de e-mail pode carregar uma mensagem de e-mail cuja intenção é entregar para algum grupo de recipiente. Esta mensagem deveria ser imutável, de forma que um servidor malicioso no caminho de um agente não pode adulterar a mensagem de um usuário [KARNIK 98].

Foi inserido no agente um objeto denominado contêiner somente-leitura. Este objeto contém um vetor de objetos de tipos arbitrários junto com a assinatura digital do proprietário do agente. Como parte da construção do objeto agente, este vetor pode ser inicializado com os valores somente-leitura apropriados. A assinatura digital é computada primeiro usando uma função *hashing* e então codificada usando a chave privada fornecida pelo construtor **Sign = Ka(h(objs))**

A computação da assinatura, pode ser feita pelo *home site* do agente quando ele é criado. Desde então este é a única localização onde a chave privada **Ka** é encontrada. A chave privada é descartada pelo construtor após a assinatura ser computada, visto que ele não pode ser carregado de forma segura junto com o agente. O método de verificação do contêiner somente-leitura

permite a qualquer servidor, no caminho do agente, checar se o estado somente-leitura do agente foi adulterado. Para fazer tal verificação é necessário ter acesso a chave publica do agente **Kp**. O verificador utiliza a chave publica para decodificar a assinatura, e compara o resultado com um *hash* recalculado do vetor de objetos. Se este valor estiver compatível, o servidor pode assumir que nenhum dos objetos foi adulterado. A condição é checada por [KARNIK 98]: **h(objs)==Kp(sign)**.

Existem dois caminhos pelos quais os servidores maliciosos poderiam tentar quebrar o esquema proposto:

1. O servidor malicioso modifica algum objeto somente-leitura de modo que a assinatura do proprietário do agente fique válida.
2. O servidor malicioso modifica algum objeto somente-leitura e também a assinatura de forma que o contêiner somente-leitura adulterado apareça como valido para outros servidores.

A primeira opção pode ser rejeitada, porque a função *hash* utilizada (SHA) é resistente à colisão, isto é, dado uma pré-imagem e sua imagem abaixo da função, é impraticável calcular outra pré-imagem que produza a mesma imagem. Em outro contexto, é impraticável para um servidor malicioso adulterar qualquer um dos objetos somente-leitura, enquanto fica retendo a mesma assinatura valida. A segunda opção, a suposição base fundamental de qualquer sistema de criptografia é que a chave privada é conhecida somente pelo dono da chave. Desta forma, nenhuma outra entidade pode produzir a assinatura do proprietário do agente sobre um valor *hash* modificado. Desta forma a segunda hipótese é descartada [KARNIK 98].

3.3.2 Vetores de Dados Direcionados

Em determinados casos, os dados de um agente devem ser protegidos para que somente determinados nós possam ter acesso. Nestes casos, o proprietário do agente não quer que as informações sejam lidas e utilizadas por outros nós localizados entre o remetente e o destinatário. Para resolver este problema [KARNIK 98] propôs o esquema da revelação seletiva.

No esquema da revelação seletiva, o agente possui vetores de objetos que são codificados com a chave público do servidor de destino. A identificação do servidor correspondente é colocada em um vetor separado. Estes dois vetores são assinados e um código *hash* é gerado

pelo dono do agente. Ao chegar em um servidor, o agente é examinado e objetos com destino a este servidor são procurados no vetor de objetos. Se encontrado este objeto é encaminhado ao servidor que requisita a decodificação utilizando sua chave privada. O objeto decodificado pode ser então usado pelo agente durante sua computação neste servidor. Assinatura digital é utilizada para verificar se o objeto não foi adulterado [KARNIK 98].

O proprietário do agente codifica cada objeto com a chave pública do servidor antes do agente ser enviado. A única forma de decodificar este agente é utilizando a chave privada correspondente, o que somente o servidor apropriado pode fazer. Se algum objeto for modificado, a assinatura contida no agente se torna invalida. Similarmente se alguém tentar modifica o conteúdo do vetor de servidores também, a assinatura se tornará incorreta e a adulteração será detectada. Novos itens não poderão ser incluídos no vetor por esta mesma razão [KARNIK 98].

3.4 Repositórios Seguros para Proteção de Agentes Móveis

Neste item será apresentada a proposta de [WANGHAM 04] que utilizou o conceito de contêiner somente leitura proposto em [KARNIK 98] para resolver o problema de segurança dos agentes móveis e sua fragilidade perante plataformas maliciosas.

3.2. Repositório Seguro

Visando detectar possíveis alterações, remoções ou inserções de dados em um agente móvel, três técnicas foram empregadas: uma para a proteção do código e dos dados imutáveis do agente (Assinatura do Código do Agente e do Repositório de Dados Somente-Leitura) e a outra para os resultados parciais recolhidos durante as viagens do agente (Repositório Seguro de Resultados Parciais). Outro esquema utilizado é o Repositório de Dados Direcionados que visa a confidencialidade de alguns dados para que não sejam revelados a plataformas não autorizadas [WANGHAM 04].

O Repositório Somente-Leitura (RSL) é utilizado para proteger a integridade do código do agente móvel que é requisitado sob demanda e das credenciais do agente que são informações imutáveis durante o transporte. O RSL assina o código do agente e envia a assinatura juntamente com o código para que qualquer plataforma visitada pelo agente possa

verificar sua integridade. O técnica adotada no RSL é baseada no Contêiner Somente-Leitura, proposto por [KARNIK 98].

O Repositório Seguro de Resultados Parciais (repositórioRP) será carregado pelo agente e será utilizado para proteger os dados sensíveis gerados em uma plataforma, para que possíveis modificações possam ser detectadas. O repositórioRP utiliza um algoritmo para inicialização do repositórioRP que consiste na aplicação de uma função *hash* sob um número aleatório gerado e a identidade da primeira plataforma a ser visitada. Para a inserção de resultados parciais [WANGHAM 04] propõe três protocolos, para serem utilizados pelas plataformas, que podem ser escolhidos pelo proprietário do agente de acordo com a necessidade. Cada plataforma visitada poderá verificar a integridades dos resultados parciais transportadas pelo agente.

O Repositório Seguro de Dados Direcionados (RSDD), possibilita uma revelação seletiva do estado de um agente, na qual o programador do agente pode implementar um vetor de dados direcionados em que cada entrada do repositório tenha plataformas específicas como destinatárias. Esta técnica exige que as plataformas definidas como receptoras estejam pré-determinadas. [WANGHAM 04] implementou uma redundância criptográfica que liga a entrada do RSDD com a instância do agente móvel correspondente para resolver uma vulnerabilidade contida no modelo proposto inicialmente por [KARNIK 98]. Esta redundância criptográfica utiliza a chave pública do receptor para cifrar os dados concatenados com o identificador único da instância do agente. A plataforma de origem deve assinar este valor para garantir a sua integridade. Quando o agente é recebido pelo destinatário, antes de executar o agente, a plataforma deve verificar a origem e integridade dos dados. A estrutura proposta para um agente móvel é apresentada na figura 3.2.

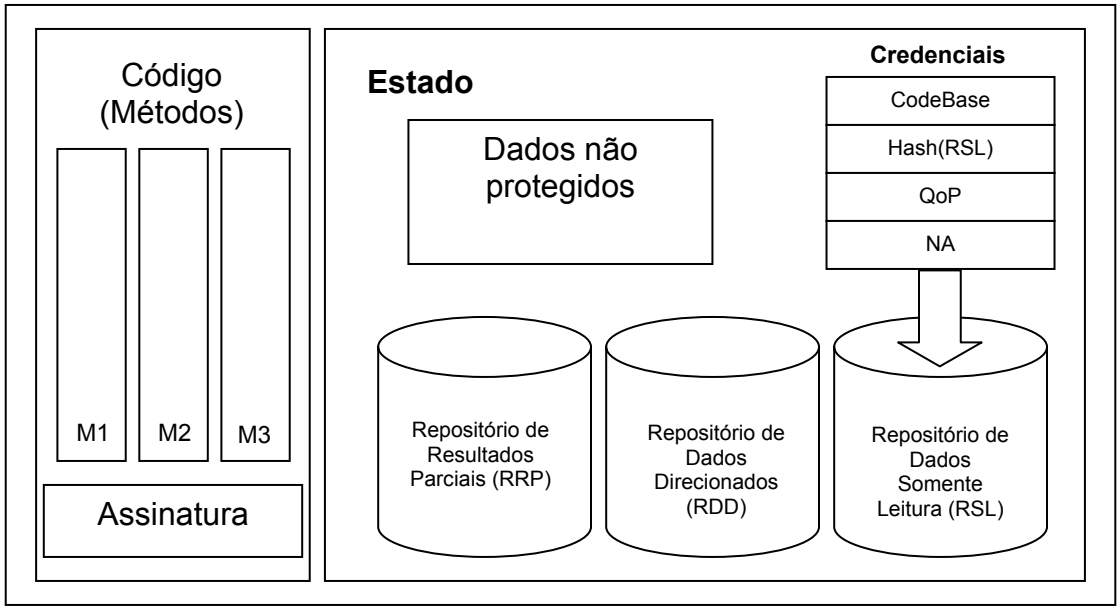


Figura 3-2: *Estrutura Proposta para um Agente Móvel*

O proprietário do agente deve assinar um núcleo estático que pode incluir o código do agente e uma redundância suficiente para distinguir entre duas instâncias de um mesmo agente (objeto Credenciais). Cada agente deve carregar parte de seu estado dentro de um repositório de dados somente-leitura, chamado de Repositório Somente Leitura (RSL). O objeto credenciais distingue as instâncias do agente e a sua assinatura ligando-o a seu proprietário, auxiliando o processo de verificação da integridade do agente e o processo de controle de acesso do proprietário do agente.

3.5 Considerações Gerais

Este capítulo apresentou alguns conceitos de criptografia e suas aplicações. Existem três tipos de criptografia a simétrica, assimétrica e a híbrida. A criptografia simétrica possui um excelente desempenho, porém o problema é garantir que a chave secreta seja distribuída sem que nenhum nó não autorizado a obtenha. A criptografia assimétrica utiliza o conceito de chave pública e privada, uma mensagem é codificada com a chave pública e decodificada com a privada (ou vice versa). Por possuir um algoritmo complexo, este esquema possui problemas de desempenho. A criptografia híbrida utiliza as vantagens da criptografia simétrica e assimétrica,

neste esquema a criptografia assimétrica é utilizada para garantir a distribuição segura da chave secreta. Uma vez distribuída com segurança, a chave secreta é responsável por codificar e decodificar as mensagens. Por ter estas características a criptografia híbrida foi a escolhida para ser utilizada neste trabalho.

O esquema de contêiner somente leitura foi exposto como solução a alguns problemas de segurança dos agentes móveis. O contêiner somente leitura permite que o nó receptor tenha garantia de que os dados que trafegaram no agente não foram modificados. O vetor de dados direcionados consegue garantir a confidencialidade dos dados, permitindo que somente o nó destino possa ler os dados. O conceito de repositório somente leitura para agentes móveis vem para aplicar o conceito de contêiner somente leitura a agentes com itinerário livre.

Se considerarmos agentes móveis como sendo um tipo de pacote ativo de uma rede ativa, podemos considerar que algumas soluções de segurança aplicadas com sucesso em agentes móveis possam servir para resolver problemas de segurança em redes ativas.

O próximo capítulo abordará a aplicação do esquema de repositório somente-leitura, repositório seguro de dados direcionados e criptografia híbrida apresentados neste capítulo, para tratar os problemas de segurança no gerenciamento de redes ativas baseado em políticas. Mais especificamente para garantir a integridade dos repositórios de políticas que trafegam dentro de uma cápsula ativa.

4 Repositório Seguro de Políticas para Redes Ativas (RSPRA)

4.1 Introdução

O objetivo deste trabalho é propor uma solução para os problemas identificados na arquitetura DEPMA. O desafio aqui é garantir a segurança do repositório de políticas, pois devido às características das redes ativas as informações que trafegam na rede podem sofrer alteração por elementos mal intencionados. A solução proposta é baseada no esquema de Repositório Seguro de Dados para Proteção de Agentes Móveis [WANGHAM 04], utilizando os conceitos de Repositório Somente Leitura (RSL) e Repositório Seguro de Dados Direcionados (RSDD). O Repositório de Resultados Parciais (repositorioRP) não será utilizado na solução proposta por não ser aplicável à solução dos problemas detectados.

No esquema proposto, o RSL utiliza assinatura digital dos dados de modo que qualquer alteração nos dados transportados é facilmente detectada. Além disso, será utilizada criptografia híbrida pelo seu desempenho e segurança. A chave pública será utilizada para garantir a entrega da chave secreta gerada pelo SPDP-A (servidor) ao SPEP-A (nó), e a partir daí será utilizada a chave secreta (simétrica), por causa de seu desempenho.

Quando um servidor envia uma política para os nós, esta política é armazenada em um RSL dentro da cápsula ativa. Este repositório será assinado pelo servidor, que irá distribuir uma chave pública de verificação desta assinatura a todos os nós da rede (contêiner somente leitura).

Nos casos em que um cliente faz uma requisição de política (*outsourcing*), o servidor deverá fazer as validações para identificar o cliente como confiável e deverá utilizar o esquema de Repositório Seguro de Dados Direcionados (RSDD) para enviar os dados, esta medida é utilizada para que nenhum outro nó tome conhecimento do tipo de política que o nó requisitante utiliza.

Nos próximos itens veremos com detalhe a iteração entre os módulos de segurança cliente (SPEP-A) e servidor (SPDP-A), além dos componentes que fazem parte desta nova arquitetura.

4.2 A Arquitetura do RSPRA

Se fizermos uma divisão entre os mecanismos de segurança que cuidam da integridade das cápsulas de uma rede ativas, semelhante ao que fez [WANGHAM 04] para agentes móveis, teríamos dois tipos distintos de mecanismos:

- **Mecanismos de Prevenção:** tentam tornar mínima a possibilidade de acesso e/ou modificação das cápsulas ativas.
- **Mecanismos de Detecção:** tentam descobrir se e quanto um ataque foi realizado à uma cápsula ativa.

A solução de segurança proposta por este trabalho pertence ao grupo dos mecanismos de detecção. A arquitetura RSPRA (Repositório Seguro de Políticas para Redes Ativas) cria um módulo de segurança no servidor de políticas (MSS – Módulo de Segurança Servidor), que é responsável por criar o contêiner somente-leitura e dados direcionados para políticas e um outro módulo de segurança nos nós ativos da rede (MSC- Módulo de Segurança Cliente) que faz a verificação dos contêineres enviados pelo servidor. A figura descreve o esquema da arquitetura proposta.

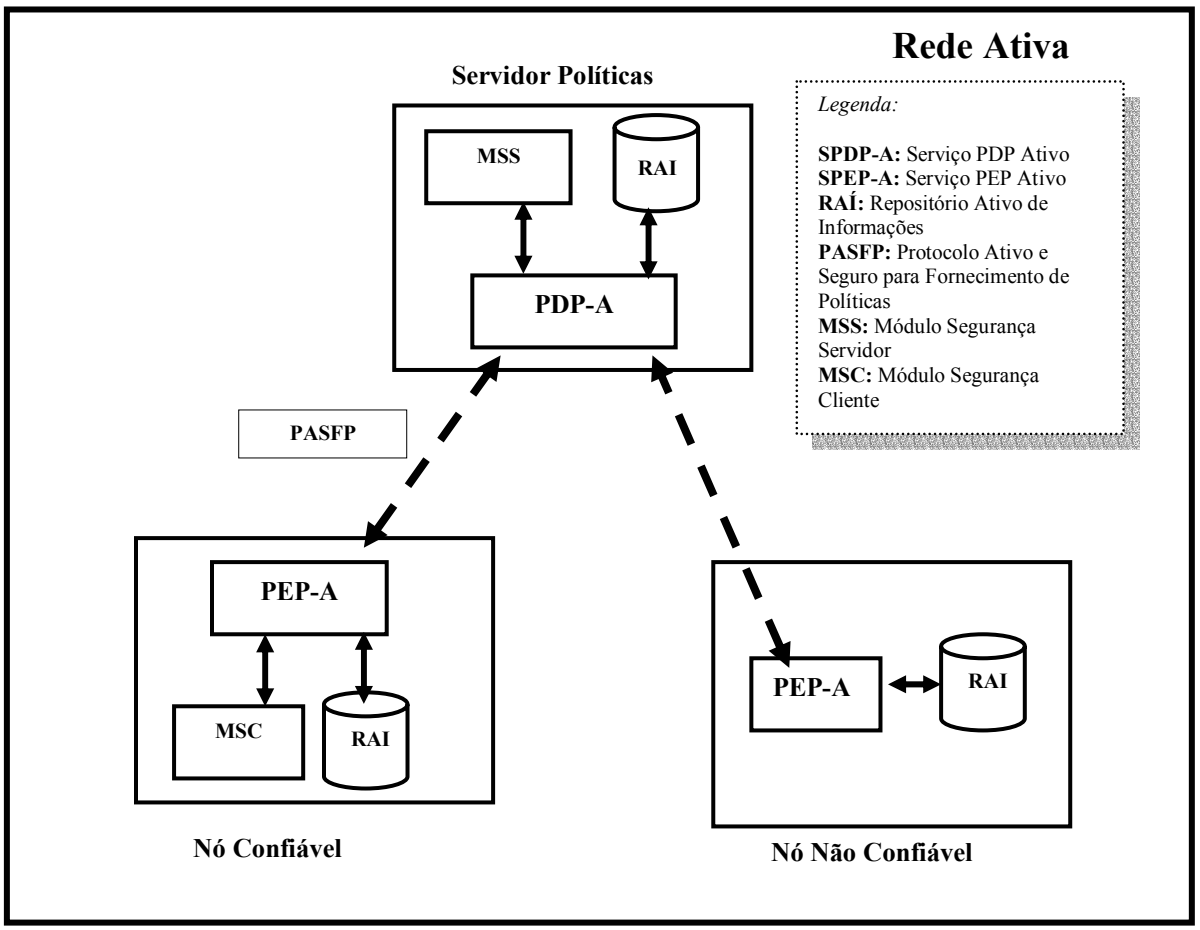


Figura 4-1: *Esquema de segurança proposto para problema de troca de políticas em uma rede ativa com gerenciamento baseado em políticas da IETF.*

Nesta arquitetura, além dos elementos originais da arquitetura DEPMA, foram incluídos novos elementos, são eles:

- **O Módulo de Segurança Servidor (MSS):** localizado no servidor de políticas da rede (SPDP-A), é responsável por criar o RSL e o RSDD que garantirão a integridade das informações (políticas) que serão passadas para os nós clientes (SPEP-A) da rede. Além disso, o MSS cria as chaves publicas/privadas do servidor, chave secreta de comunicação (criptografia simétrica), armazena as chaves públicas dos nós e autentica os nós confiável.
- **Módulo de Segurança Cliente (MSC):** localizado nos nós ativos da rede (SPEP-A), é responsável pelo tratamento e validação do RSL e RSDD que o servidor

enviará, criação das chaves pública/privada do nó, autenticação do servidor no RAI local e pelo envio de solicitação de novas políticas ao servidor.

- **PASFP:** protocolo ativo seguro para fornecimento de políticas, é uma variação do protocolo PAFP criado na arquitetura DEPMA. Este novo protocolo trata questões de segurança que não são tratadas no protocolo PAFP. Foram incluídos mecanismos de segurança nas cápsulas para garantir a integridade dos dados transportados.

Como podemos perceber, a arquitetura inseriu módulos de segurança no servidor de políticas e nos nós confiáveis e incluiu segurança no protocolo de troca de políticas. Estes módulos são responsáveis por garantir a integridade dos repositórios de políticas espalhados pela rede. A seguir será detalhado o funcionamento de cada um destes elementos.

4.2.1 Módulo de Segurança Servidor (MSS)

O Módulo de Segurança Servidor (MSS) está localizado no servidor de políticas da rede (SPDP-A), e é responsável por criar o repositório somente leitura e o repositório de dados direcionados, que garantirão a integridade das informações (políticas) durante o transporte. O trabalho do MSS consiste em:

- Gerar chaves pública/privada do servidor
- Fornecer a todos os elementos da rede sua chave pública para verificação da assinatura do RSL e RSDD.
- Autenticar os clientes confiáveis, gerar a chave secreta para cada um deles e armazenar o identificador do nó, sua chave pública e a chave secreta de comunicação.
- Montar o RSL, com as políticas, na cápsula ativa e enviar para todos os nós (disseminação de políticas).
- Montar o RSDD, com as políticas solicitadas pelo cliente, na cápsula ativa e enviar para o nó solicitante.

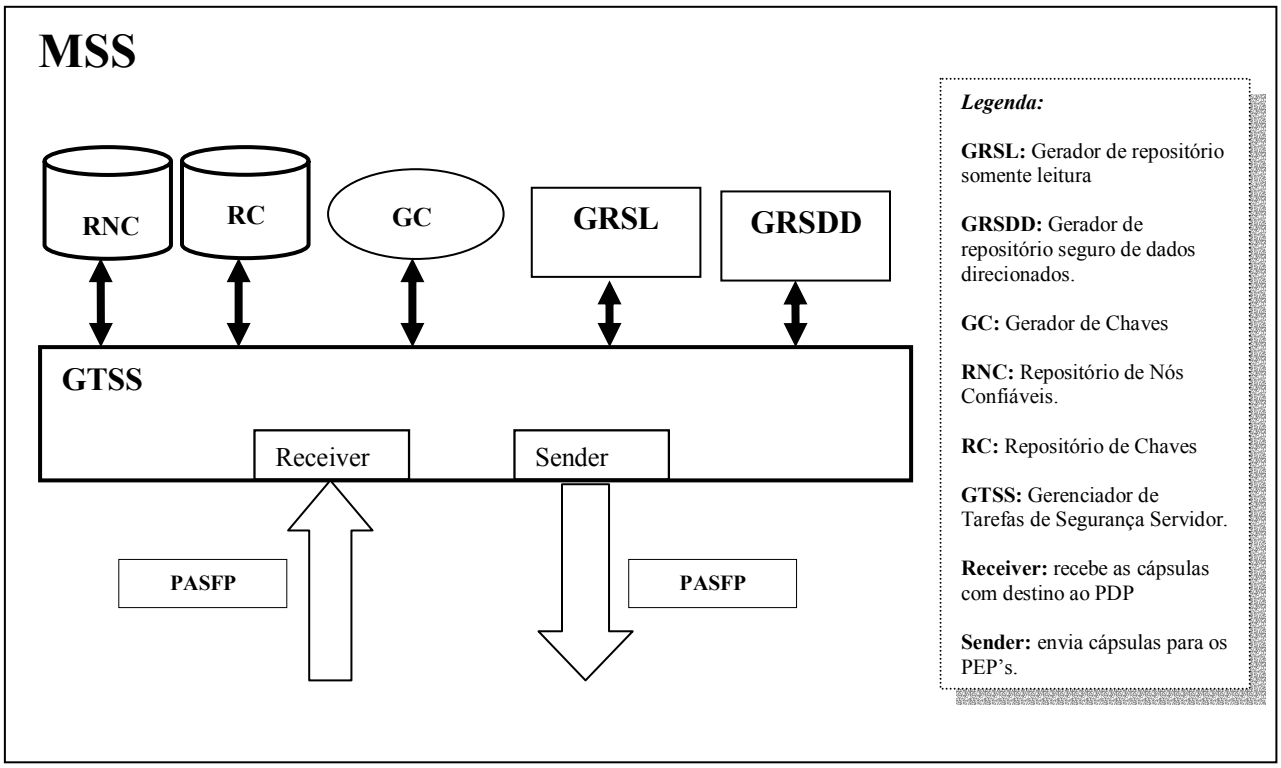


Figura 4-2: *Módulo de Segurança do Servidor SPDP-A*

- **GRSL:** É responsável pela criação do repositório somente leitura.
- **GRSDD:** É responsável pela criação do repositório seguro de dados direcionados.
- **GC:** Gera as chaves pública/privada do servidor e as chaves secretas para cada um dos nós confiáveis
- **RNC:** O repositório de nós confiáveis armazena todos os nós confiáveis da rede. Este cadastro é mantido pela administrador da rede.
- **RC:** O repositório de chaves armazena todas as chaves secretas e seus respectivos nós.
- **GTSS:** O gerenciador de tarefas de segurança do servidor é responsável por controlar todas as tarefas do módulo de segurança localizado no SPDP-A.
- **Receiver:** módulo de recebimento de cápsulas ativas.
- **Sender:** módulo de envio de cápsulas ativas.

4.2.1.1 Gerador de Repositório Somente Leitura (GRSL)

O GRSL (Gerador de Repositório Somente Leitura) é responsável pela geração do repositório somente leitura (RSL). O GRSL assina os dados a serem transportados utilizando a chave privada do servidor, gerando um recibo de assinatura. Com este recibo e os dados originais, o Gerenciador de Tarefas de Segurança do Servidor (GTSS) monta a cápsula e envia para o destino. As chaves necessárias para realizar tais operação são recuperadas pelo GTSS do RC e enviadas ao GRSL.

4.2.1.2 Gerador de Repositório Seguro de Dados Direcionados (GRSDD)

O GRSDD (Gerador de Repositório Seguro de Dados Direcionados) é responsável pela geração do repositório seguro de dados direcionados (RSDD). A principal função do RSDD é fazer a troca de chaves secretas entre o SPDP-A e o SPEP-A com segurança, além disso ele é responsável pela troca de informações entre o SPEP-A e o SPDP-A de forma confidencial. Para formar o RSDD, o GRSDD assina os dados utilizando a chave privada do SPDP-A, gerando com isso um recibo de assinatura. Este recibo juntamente com a informação original são codificados utilizando-se a chave secreta de comunicação. Nos casos em que o GRSDD gera o RSDD para efetuar a troca da chave secreta entre o SPDP-A e o SPEP-A (abertura de comunicação de um SPEP-A), o recibo da chave e a própria chave secreta são codificadas com a chave pública do SPEP-A destino. As chaves necessárias para realizar tais operação são recuperadas pelo GTSS do RC e enviadas ao GRSL.

4.2.1.3 Gerador de Chaves (GC)

O gerador de chaves (GC) possui uma função bastante específica e importante dentro da arquitetura. Sua função é a de gerar as chaves pública e privada do SPDP-A e as chaves secretas de comunicação para cada um dos SPEP-A's confiáveis da rede. O GTSS aciona o GC na inicialização do SPDP-A e a cada requisição de abertura de conexão dos SPEP-A's confiáveis da rede.

4.2.1.4 Repositório de Nós Confiáveis (RNC)

O Repositório de Nós Confiáveis (RNC) possui o cadastro de todos os nós (SPEP-A's) confiáveis da rede. Este repositório é utilizado pelo GTSS a cada pedido de conexão de um SPEP-A ao SPDP-A para verificar se este SPEP-A é confiável.

4.2.1.5 Repositório de Chaves (RC)

O Repositório de Chaves (RC) é responsável por armazenar as chaves pública e secreta de um determinado SPEP-A, além de um identificador com o qual o GTSS possa recuperar as informações. O RC é acionado pelo GTSS todas as vezes que um RSL ou um RSDD for gerado.

4.2.1.6 Gerenciador de Tarefas de Segurança Servidor (GTSS)

O GTSS é responsável pelas tarefas de segurança do servidor. É o elemento principal da arquitetura de segurança do SPDP-A e é responsável por coordenar as tarefas dos demais elementos de segurança. O GTSS é responsável por verificar a confiabilidade dos SPEP-A's que requisitam políticas ao SPDP-A, coordenar a geração das chaves públicas/privadas e secretas, armazenar as chaves pública e secreta de comunicação com o SPEP-A, montar o RSDD e o RSL.

A verificação da confiabilidade de um SPEP-A é realizada através de consulta de um repositório de nós confiáveis. O SPDP-A obtém o identificador do SPEP-A requisitante e verifica, no RNC, se está cadastro. Se constatado que o SPEP-A é confiável, a comunicação segura é iniciado, caso contrário o requisitante é tratado como um SPEP-A qualquer sem nenhum tipo de tratamento de segurança. O cadastro dos nós confiáveis é feito previamente pelo administrador da rede.

Uma vez constatado que o SPEP-A é confiável o GTSS pede ao GC que gere uma chave secreta de comunicação para este SPEP-A. A chave secreta é gerada e armazenada juntamente com o identificador do SPEP-A, sua chave pública e a chave secreta de comunicação. Estas informações serão recuperadas pelo identificador do SPEP-A todas as vezes que o GTSS necessitar. Na inicialização do SPDP-A o GTSS é responsável por requisitar ao GC a geração das chaves públicas e privadas do SPDP-A que são armazenadas para uso posterior.

O *Receiver* e *Sender* são respectivamente elementos que recebem e enviam cápsulas de e para os SPEP-A's. Na arquitetura DEPMA o *Sender* e o *Receive* eram módulos independentes que tinham o objetivo único de enviar e receber cápsulas. Na arquitetura proposta, estes elementos foram incorporados ao GTSS para que a segurança seja tratada desde o momento do recebimento de uma cápsula. Foi incorporada ao *receiver* a capacidade de distinguir as cápsulas e suas finalidades.

4.2.2 Módulo de Segurança Cliente (MSC)

O Módulo de Segurança Cliente (MSC) é localizado nos nós ativos da rede (SPEP-A), e será responsável pelo tratamento e validação do RSL e do RSDD que o servidor envia. Além disse, o MSC será responsável pela geração das chaves pública/privada do nó, por enviar solicitação de novas políticas ao servidor, e autenticar o servidor antes que este se registre em seu RAI local. O trabalho do MSC consiste em:

- Gerar chaves pública/privada do nó.
- Codificar e enviar mensagens de requisição para o servidor.
- Reconhecer mensagens contendo RSL provenientes do servidor e fazer as devidas validações de segurança.
- Reconhecer mensagens contendo RSDD provenientes do servidor e fazer a decodificação e verificação de segurança do repositório.
- Armazenar as chaves públicas e secreta concedida pelo servidor.

O MSC cria uma camada de segurança com a capacidade de detectar possíveis violações das políticas enviadas pelo servidor. Esta camada impede, por exemplo, que um nó malicioso adultere um pacote ativo contendo políticas a fim de alterar o comportamento do nó, fazendo com que este passe a funcionar de maneira diferente da definida pelo servidor.

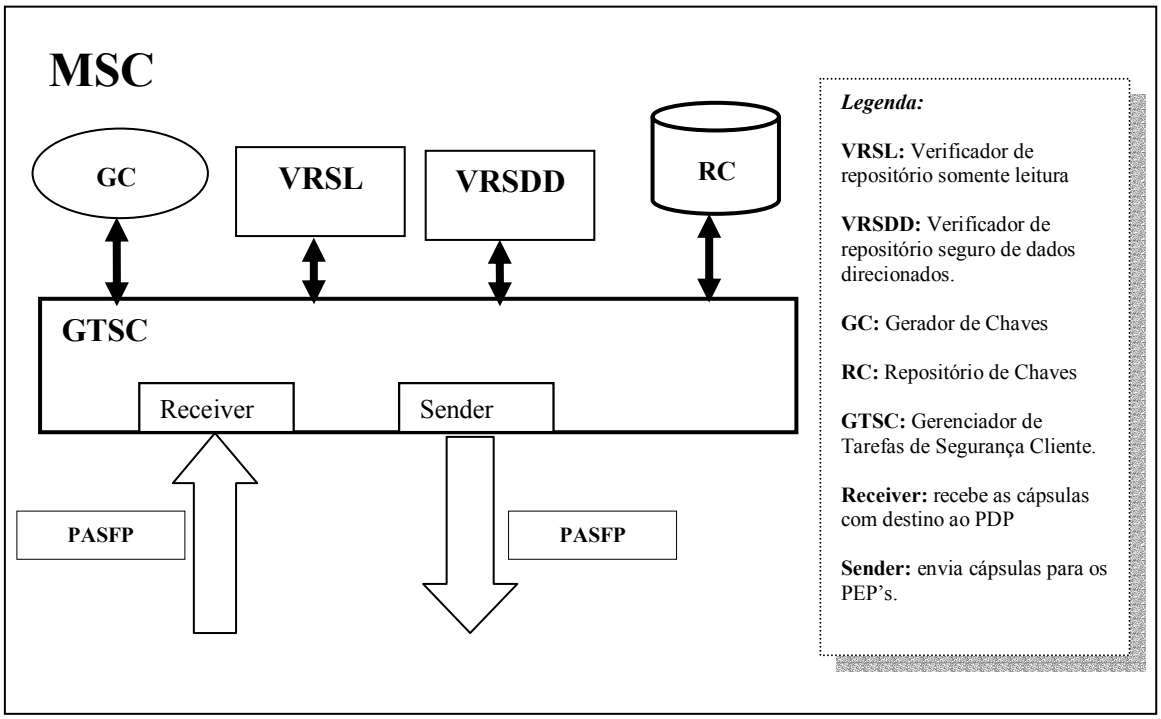


Figura 4-3: A *arquitetura do Módulo de Segurança Cliente*

- **VRSL:** é responsável por verificar a consistência do RSL enviado pelo SPDP-A.
- **VRSDD:** é responsável por verificar a consistência e extração dos dados do RSDD enviado pelo SPDP-A.
- **GC:** gera as chaves pública/privada do nó
- **RC:** Repositório de chaves, armazena a chave privada do nó, pública do servidor e a chave secreta de comunicação
- **GTSC**: O gerenciador de Tarefas de Segurança do Cliente é responsável por controlar todas as tarefas do módulo de segurança localizado no SPDP-A.
- **Receiver:** módulo de recebimento de cápsulas ativas.
- **Sender:** módulo de envio de cápsulas ativas.

4.2.2.1 VRSL

O Verificador de Repositório Somente Leitura (VRSL) é responsável por verificar a integridade dos dados, enviados pelo SPDP-A, que chegam por uma cápsula ativa em um RSL.

Seu objetivo é detectar alguma modificação sofrida pelos dados transportados durante seu percurso. Assim que uma cápsula é recebida pelo SPEP-A e identificado que ela possui um RSL, o GTSC imediatamente aciona o VRSL que faz a verificação de consistência do RSL. O VRSL possui um verificador de assinatura que utiliza da chave pública do servidor. Este verificador gera o recibo dos dados originais e compara com o recibo enviado no RSL juntamente com os dados.

4.2.2.2 VRSDD

O Verificador de Repositório Seguro de Dados Direcionados (VRSDD) é responsável por verificar a integridade dos dados, enviados em um RSDD, pelo SPDP-A através de uma cápsula ativa. O GTSC aciona o VRSDD assim que uma cápsula contendo um RSDD é recebida. Assim que é ativado, o VRSDD decodifica o conteúdo do RSDD, utilizando a chave secreta de comunicação enviada pelo SPDP-A na sua inicialização. Com os dados decodificados, o SPEP-A tem acesso aos dados originais e ao recibo da assinatura. O recibo obtido é entregue para o verificador de assinatura que regera o recibo dos dados originais e compara com o recibo recebido no RSDD. O verificador faz uso da chave pública do SPDP-A, enviada por este como resposta ao pedido de abertura de conexão.

4.2.2.3 Gerador de Chaves (GC)

Assim como no SPDP-A, o gerador de chaves (GC) possui uma função bastante específica e importante. Sua função é a de gerar as chaves pública e privada do SPEP-A, ao contrário do SPDP-A o GC do SPEP-A não gera chave secreta de comunicação. O GC é acionado, na inicialização do SPEP-A, para gerar as chaves publica/privada que servirão basicamente para gerar/verificar o RSDD.

4.2.2.4 Repositório de Chaves (RC)

O repositório de chaves (RC) é responsável por armazenar as chaves pública/privada gerada pelo SPEP-A, a chave pública do SPDP-A e a chave secreta de comunicação. A chave

pública do SPDP-A é utilizada pelo SPEP-A para autenticar e verificar a assinatura dos dados enviados pelas cápsulas.

4.2.2.5 Gerenciador de Tarefas de Segurança Cliente (GTSC)

O GTSC é responsável por coordenar as tarefas do módulo de segurança do SPEP-A. O GTSC coordena a geração as chaves pública/privada do SPEP-A, a validação de integridade dos RSDD's e RSL's enviados pelo SPDP-A e tem a tarefa de autenticar o SPDP-A antes deste se registrar em seu RAI local. Além disso, o GTSC coordena o armazenamento de suas chaves pública/privada e da chave pública do servidor no RC.

Assim que recebe uma cápsula, o *receiver* informa ao GTSC a qual tipo ela pertence. O GTSC, for sua vez, encaminha os dados da cápsula ao VRSL ou VRSDD (dependendo do tipo), passando também como parâmetros as chaves recuperadas do RC. Após receber do SPDP-A a chave pública do servidor e a chave secreta de comunicação, o SPEP-A recebe uma cápsula pedindo conexão ao seu RAI local, contendo um RSL com o identificador do SPDP-A. Este RSL é repassado para o VRSL que verifica a autenticidade e integridade dos dados. Uma vez constatado que o SPDP-A é confiável, o SPEP-A faz o registro do SPDP-A em seu RAI.

Na inicialização do SPEP-A o GTSS requisita ao GC a geração das chaves públicas e privadas do SPEP-A que são armazenadas para uso posterior. Assim como no MSS, o *Receiver* e *Sender* foram incorporados ao MSC .

4.2.3 Protocolo Ativo Seguro para Fornecimento de Políticas (PASFP)

O Protocolo Ativo Seguro para Fornecimento de Políticas (PASFP) é uma variação do protocolo PAFP criado na arquitetura DEPMA. O protocolo PASFP é responsável pela troca de políticas entre o SPEP-A e o SPDP-A, assim como na versão original. Porém, nesta nova versão do protocolo, esta troca é realizada de modo seguro.

O PASFP possui estrutura semelhante ao PAFP, ou seja, é formado por cápsulas ativas que realizam as interações necessárias para o intercâmbio de políticas. O protocolo PASFP é dividido entre funcionalidades comuns e dinâmicas, assim como o protocolo original. As cápsulas que representam tais funcionalidades sofreram modificações para garantir segurança no

transporte. Os elementos pertencentes as funcionalidades comuns, além de transportar as solicitações, atualizações e exclusões, transportam também elementos que são responsáveis pelas verificações de segurança como chaves públicas, privadas e secretas. Além disso algumas cápsulas foram modificadas para poderem transportar o RSDD e o RSL. O PASFP é o grande responsável pela inclusão de segurança na nova arquitetura, pois graças a modificações de suas cápsulas ativas, tornou-se possível executar todas as operações de troca de chaves e transporte de repositórios seguros.

4.2.4 Dinâmica da Arquitetura

Nos próximos itens será apresentada a dinâmica dos componentes da arquitetura de segurança: inicialização do sistema, requisição de políticas (*Outsourcing*), disseminação de políticas (*Provisioning*), composição do RSDD, Verificação do RSDD, composição do RSL e verificação do RSL.

4.2.4.1 Inicialização do Sistema

O modelo de criptografia utilizado nesta proposta é o modelo híbrido (junção dos modelos simétrico e assimétrico), onde a chave pública é utilizada somente para garantir que a entrega da chave secreta seja efetuada com segurança. Por este motivo o SPEP-A possui um gerador de chaves públicas onde, na requisição de *login*, esta chave é enviada ao SPDP-A. O SPDP-A verifica se o nó é confiável, gera a chave secreta, cifra-a com a chave pública recebida e a envia para o SPEP-A. A inicialização do sistema é feita da seguinte forma.

Um SPEP-A, ao ser inicializado, requisita *login* ao servidor de políticas (SPDP-A) da rede, enviando sua chave pública. Ao receber a requisição, o SPDP-A verifica se o SPEP-A é confiável através de uma lista de nós confiáveis inserida pelo administrador da rede. Se o SPEP-A for confiável, o SPDP-A gera uma chave secreta para este SPEP-A, cifra-a e assina com a chave pública enviada pelo SPEP-A requisitante, gerando assim um RSDD. O SPDP-A então armazena no RC a chave secreta e a chave pública vinculada ao SPEP-A (o SPDP-A gerará uma chave secreta para cada SPEP-A confiável da rede).

O SPDP-A envia, para o SPEP-A requisitante, uma mensagem contendo um RSDD com a chave secreta e uma chave pública do próprio servidor de políticas. Ao receber a cápsula, o SPEP-A requisitante verifica a consistência do RSDD. O SPEP-A obtém a chave secreta, gerada pelo SPDP-A, e a chave pública do SPDP-A e as armazena em seu RC para futuras trocas de mensagens. Se o SPEP-A não for confiável o SPDP-A não gerará nenhuma chave secreta, porém a chave pública do SPDP-A será enviada. O processo de inicialização é apresentado na figura 4.4 :

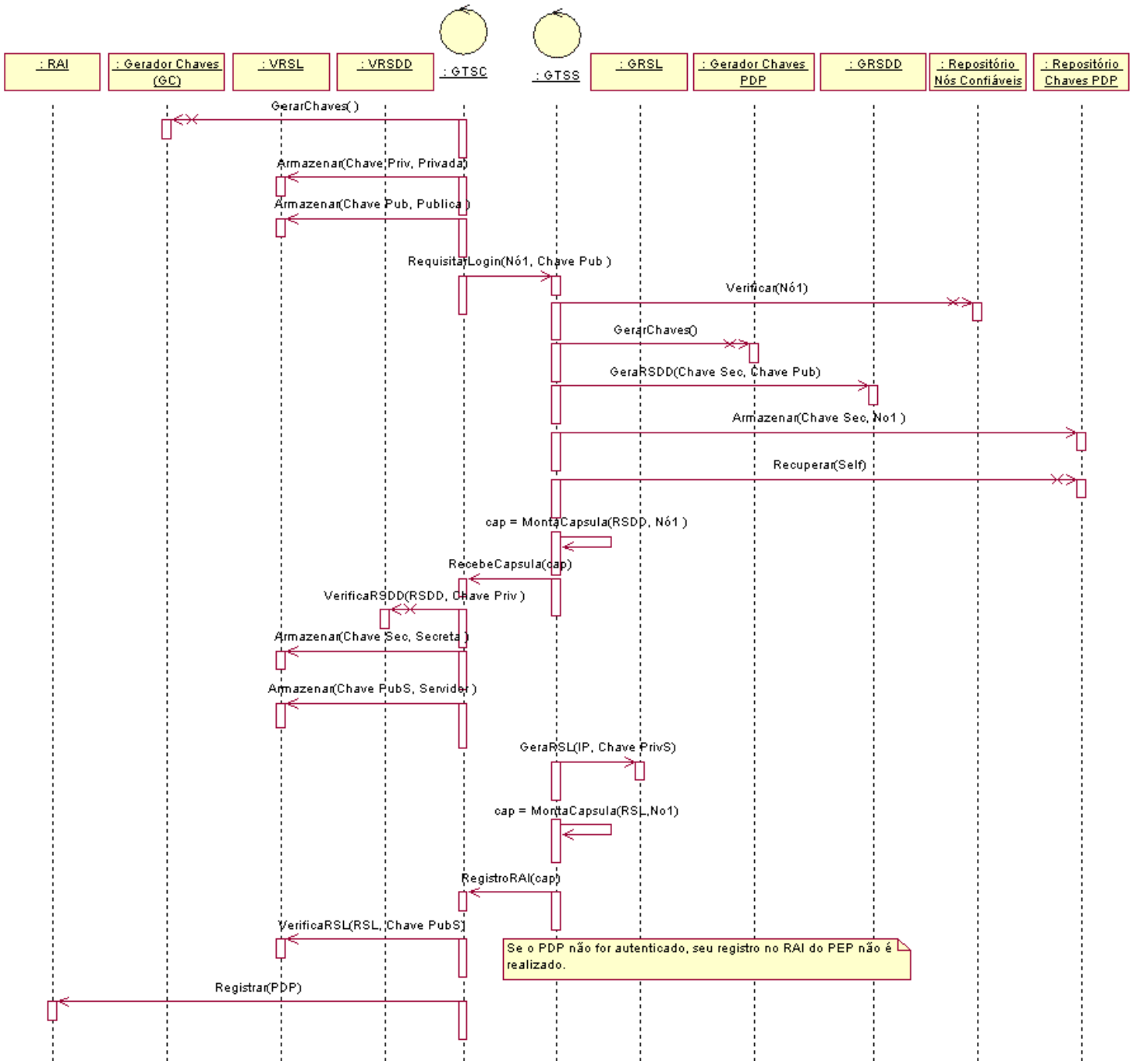


Figura 4-4: *Apresenta a seqüência de Inicialização do Sistema de um SPEP-A confiável.*

4.2.4.2 Requisição de Políticas (*Outsourcing*)

A requisição de políticas (*Outsourcing*) é realizada pelo SPEP-A todas as vezes que uma nova necessidade surge. Este processo é necessário para que o SPEP-A obtenha do SPDP-A a política correta para esta nova situação. O SPEP-A envia ao SPDP-A a requisição de políticas. O SPDP-A verifica se o SPEP-A requisitante é confiável consultando a lista de nós confiáveis inserida pelo administrador da rede. Se o SPEP-A for confiável, o SPDP-A recupera as políticas que interessam a este SPEP-A, gera um RSDD e envia a cápsula ao requisitante (para maiores detalhes sobre a geração do RSDD, vide o item Composição do Repositório Seguro de Dados Direcionados descrito no figura 4.7).

Ao receber a cápsula, o SPEP-A faz a verificação do RSDD. Se o repositório for validado, o SPEP-A insere/exclui/altera as políticas no repositório de políticas local de acordo com as determinações do SPDP-A. Caso contrário, as políticas são descartadas e uma nova requisição é enviada ao SPDP-A. Se o SPEP-A não for confiável, o SPDP-A envia as políticas sem nenhum esquema de segurança. O processo de requisição de políticas é apresentado na figura 4.5.

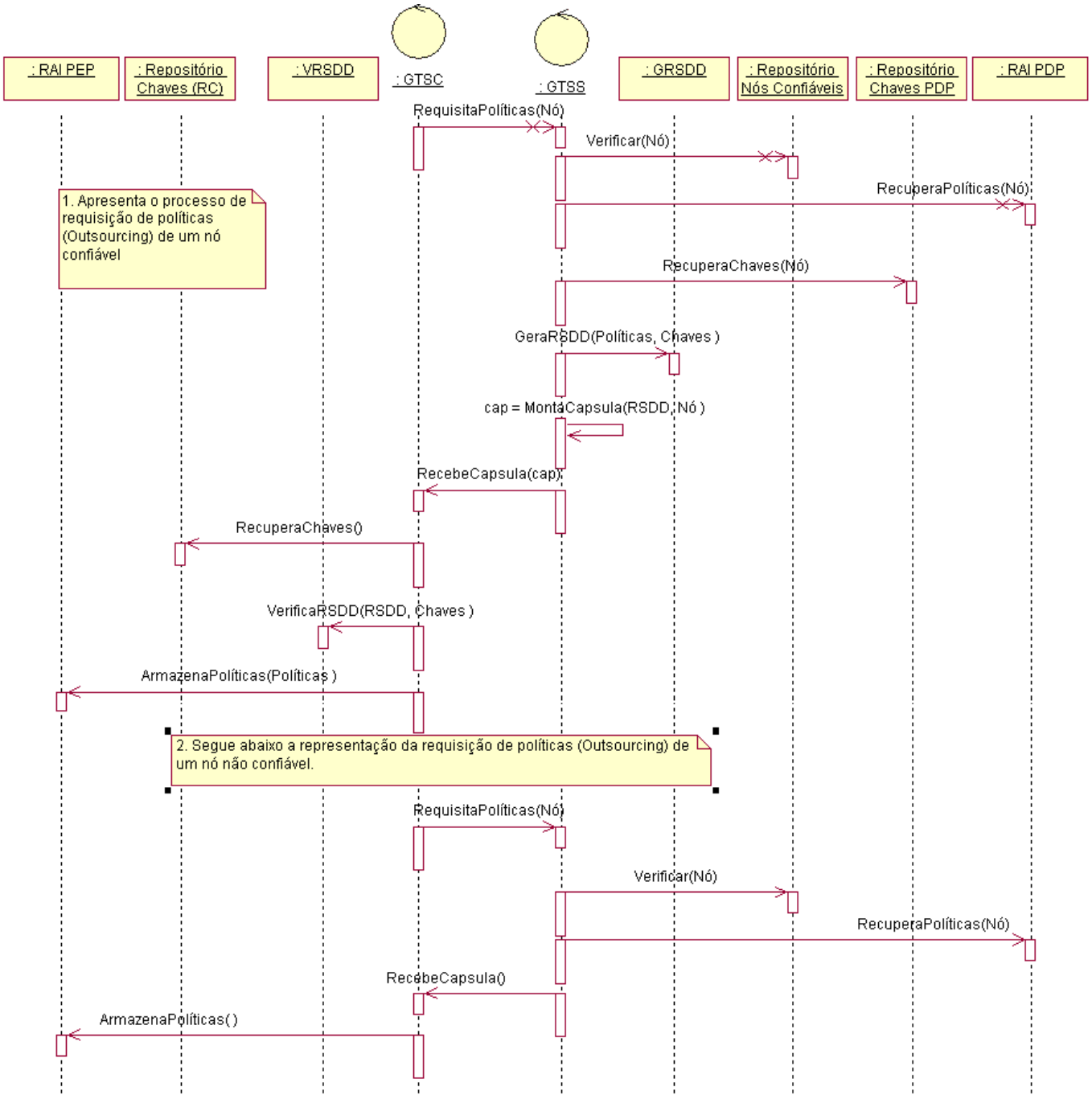


Figura 4-5: *Apresenta a requisição de Políticas de SPEP-A's confiáveis e não confiáveis*

4.2.4.3 Disseminação de Políticas (*Provisioning*)

Após o administrador do sistema incluir/excluir/altera uma nova política no repositório, o SPDP-A deverá enviar esta alteração de política para o repositório local do SPEP-A em questão. Esta disseminação é feita da seguinte forma: o administrador da rede insere/exclui/altera a política no servidor de políticas (SPDP-A) que deve ser disseminada para o SPEP-A. Com tal alteração, o SPDP-A cria um RSL (vide composição do RSL descrita na figura 4.9). O RSL é inserido em uma cápsula ativa e enviado para o SPEP-A. Os SPEP-A's confiáveis, ao receber a cápsula, verificam se o RSL sofreu alguma alteração no seu caminho. Caso o RSL tenha sido alterado, a política é descartada e uma requisição de obtenção de políticas é enviada ao SPDP-A. Se o RSL não foi alterado, a política é inserida/excluída/alterada no repositório local. Os SPEP-A's não confiáveis, ao receber a cápsula, inserem/excluem/alteram as políticas direto no seu repositório local sem fazer qualquer consistência de segurança, pois tais nós não possuem esquema de segurança. O processo de disseminação de políticas (*Provisioning*) é apresentado na figura 4.6.

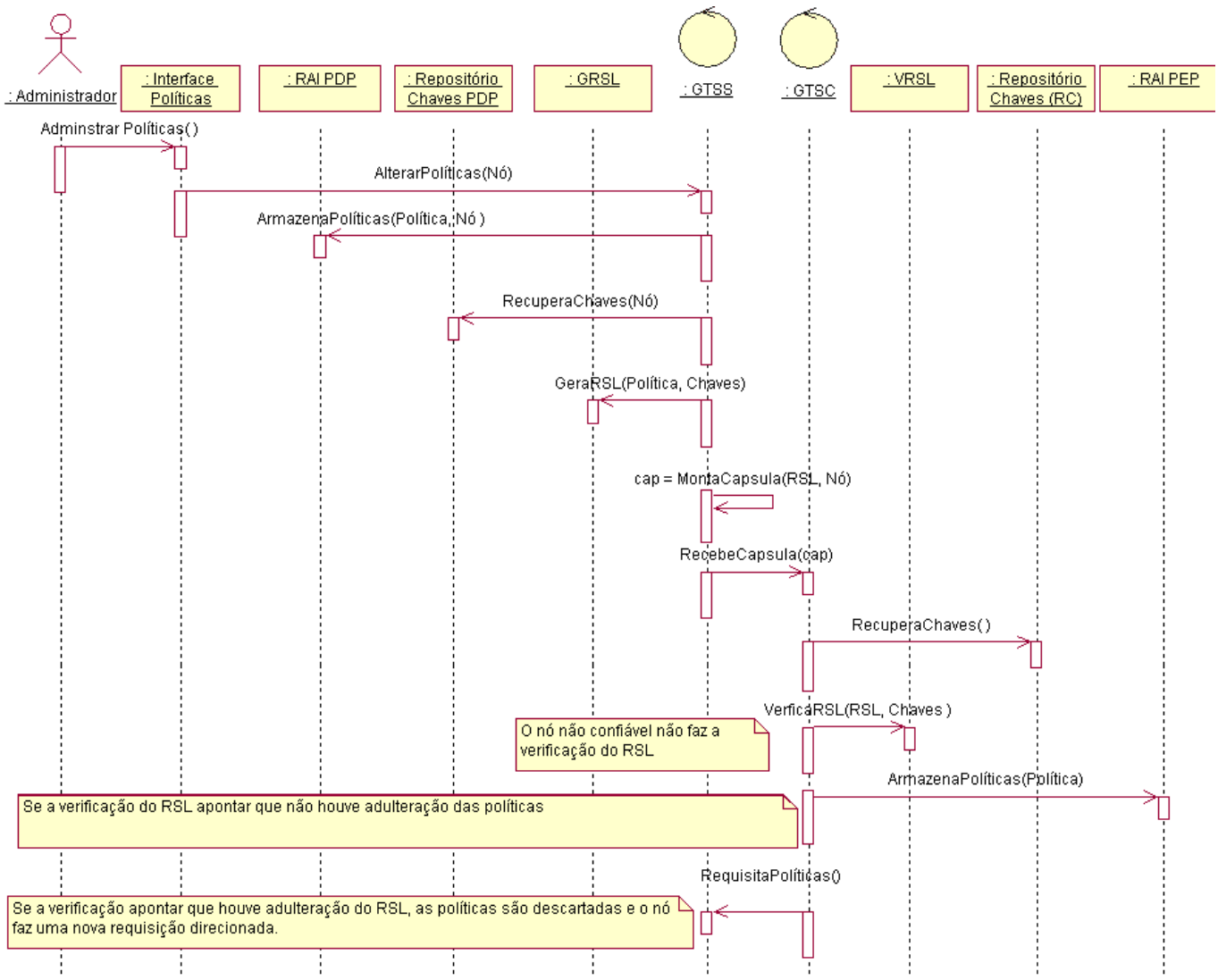


Figura 4-6: *Apresenta a disseminação de políticas para um nó confiável*

4.2.4.4 Composição do Repositório Seguro de Dados Direcionados (RSDD)

A composição do RSDD faz parte do processo de requisição de políticas (*Outsourcing*) e distribuição da chave secreta de comunicação na inicialização do sistema, e foi desmembrada dos referidos diagramas para proporcionar um melhor entendimento. O RSDD é um repositório de políticas seguro assinado e cifrado, e tem o objetivo de proteger os dados e fornecer confidencialidade (confiabilidade fundamental na distribuição da chave secreta de comunicação). Após receber a requisição do SPEP-A, o GTSS recupera as políticas que

interessam ao SPEP-A requisitante e recupera, de seu repositório de chaves (RC), a chave secreta de comunicação e a chave privada do servidor. Após recuperar a chave, o GTSS envia ao GRSDD as políticas e as chaves pública e secreta. Se o RSDD estiver sendo gerado para a troca da chave secreta de comunicação o GTSS envia ao GRSDD a chave secreta (como informação à ser cifrada) e a chave pública do SPEP-A. O GRSDD aplica uma função *hash* sobre as políticas utilizando a chave privada do servidor, gerando assim um recibo de assinatura digital. O recibo juntamente com as políticas são codificados utilizando a chave secreta de comunicação, gerando desta forma o RSDD. No caso em que o RSDD está sendo gerado para a troca de chaves, o GRSDD aplica uma função *hash* da mesma forma que na geração do RSDD para as políticas, porém para codificar o recibo e a chave secreta o GRSDD utiliza a chave pública do SPEP-A. O processo de criação do RSDD é apresentado na figura 4.7.

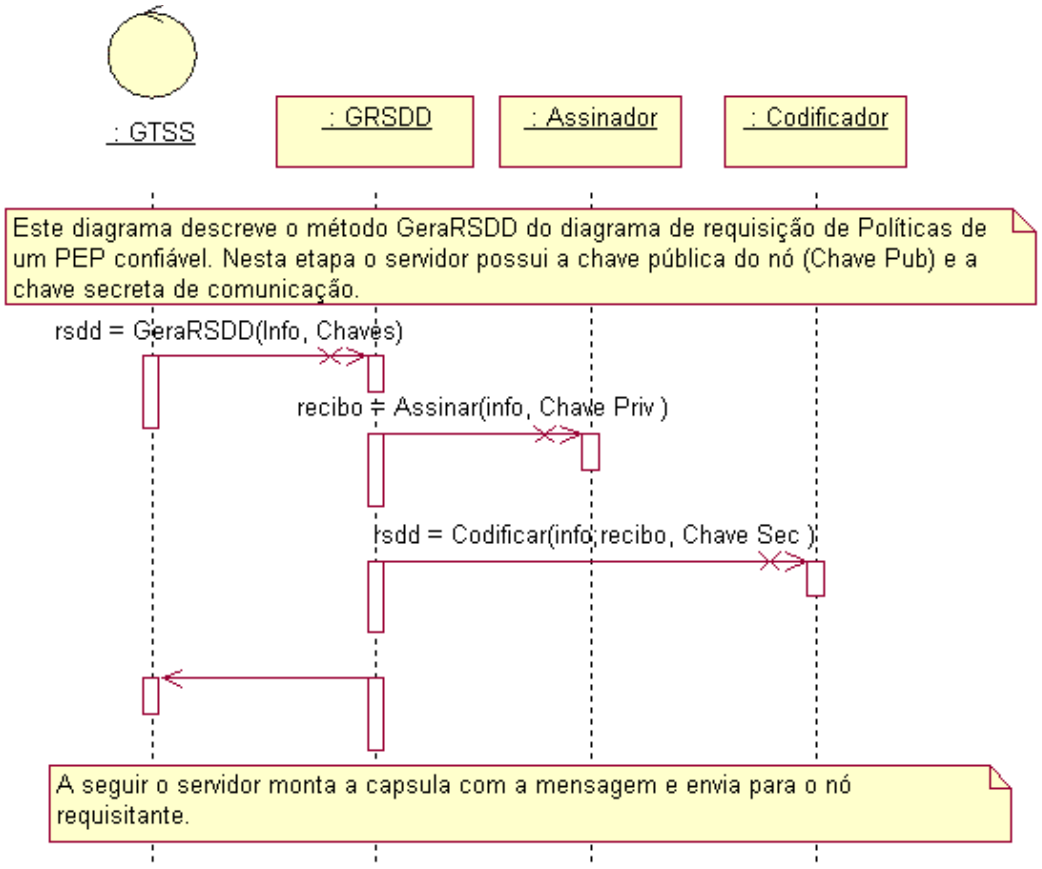


Figura 4-7: *Apresenta a criação do Repositório Seguro de Dados Direcionados*

4.2.4.5 Validação do Repositório Seguro de Dados Direcionados (RSDD)

A verificação do RSDD faz parte dos diagramas de inicialização do sistema e requisição de políticas, e foi separado do processo original para melhorar o entendimento.

Ao receber a cápsula, o GTSC passa ao VRSDD o RSDD e as chaves recuperadas do repositório de chaves (RC). O VRSDD, utilizando a chave secreta de comunicação, decifra o RSDD obtendo a informação original e o recibo de assinatura enviado. Caso o VRSDD não consiga decifrar o RSDD, as informações devem ser descartadas, pois não foi o servidor SPDP-A quem enviou a mensagem. Se o RSDD for decifrado, o VRSDD prossegue com a operação de validação. Após fazer a decodificação do RSDD e obter os dados originais e o recibo, o VRSDD aplicada, sobre os dados originais, uma função *hash* utilizando a chave pública do servidor, gerando um recibo de assinatura. Este recibo é comparado ao recibo que foi enviado no RSDD, se for o mesmo, as políticas são aplicadas. Caso a conferência dos recibos falhe, as políticas são descartadas e uma nova solicitação de políticas deve ser enviada ao SPDP-A. Nos casos em que o RSDD estiver portando a chave secreta de comunicação (na inicialização do sistema), o RSDD é decifrado utilizando-se a chave privada do SPEP-A e não a chave secreta de comunicação.

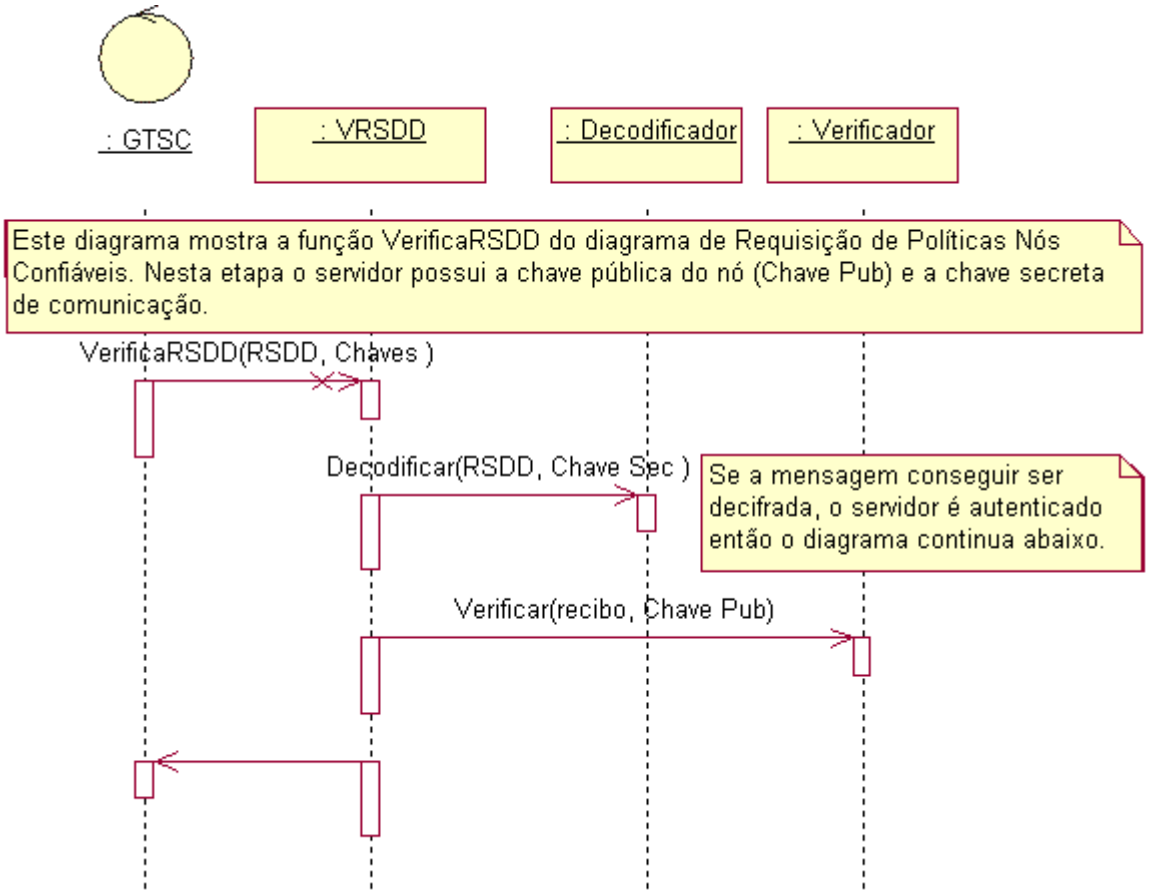


Figura 4-8: *Validação do RSDD*

4.2.4.6 Composição do Repositório Somente Leitura (RSL)

O RSL é gerado no processo de disseminação de políticas (*Provisioning*) e foi separado do processo principal para prover um maior entendimento. O RSL é gerado com as políticas que precisam ser enviadas para os repositórios locais dos SPEP-A's, devido a uma inclusão, alteração ou exclusão de políticas. O objetivo do RSL é fornecer meios para que o SPEP-A possa detectar alguma alteração nos dados gerados pelo SPDP-A, realizada por algum nó malicioso da rede. Além disso o RSL é utilizado pelo servidor para enviar solicitação de registro no RAI local dos SPEP-A's.

Após verificar que à necessidade de fazer a disseminação de políticas, o GTSS recupera, de seu repositório, as políticas que devem ser enviadas para os SPEP-A's, recupera sua chave privada e os repassa ao GRSL. O GRSL aplica uma função *hash* nas políticas utilizando-se a

chave privada do servidor, obtendo um recibo de assinatura digital. As políticas e o recibo da assinatura digital são encapsuladas e enviadas aos SPEP-A's. As políticas não são cifradas para que os SPEP-A's não confiáveis possam lê-las. O processo de composição do RSL é apresentado na figura 4.9.

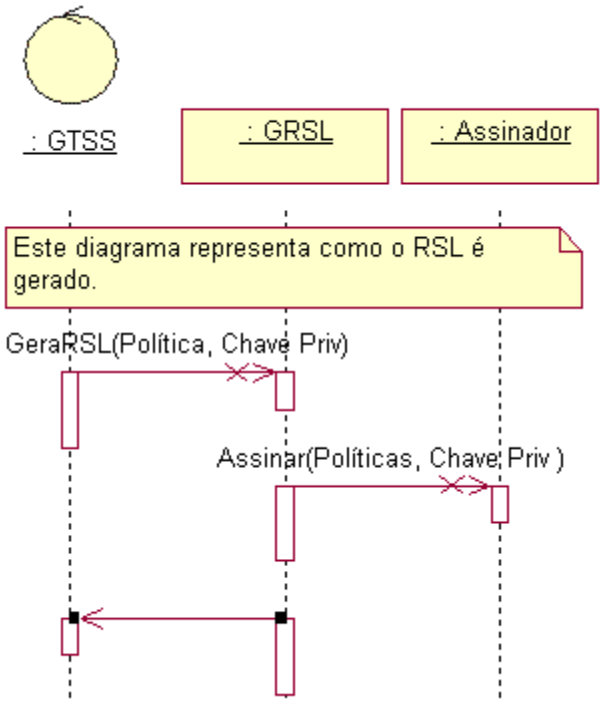


Figura 4-9: *Criação do RSL*

4.2.4.7 Validação do Repositório Somente Leitura (RSL)

Após receber o RSL, o SPEP-A precisa fazer todas as verificações para se certificar de que os dados que chegaram são os mesmos que foram enviados pelo SPDP-A. Esta verificação é realizada durante o processo de disseminação de políticas e foi desmembrada do processo principal para prover um melhor entendimento.

Após receber o RSL, o GTSC repassa ao VRSL a chave pública do servidor e o RSL. O VRSL aplica a função *hash*, utilizando a chave pública do servidor, para obter o resumo das políticas. O recibo gerado é comparado com o recibo recebido na cápsula. Se forem iguais o

GTSC aplica as políticas, caso contrário as informações são descartadas e uma nova requisição de política é enviada ao SPDP-A. O processo de validação do RSL é apresentado na figura 4.10.

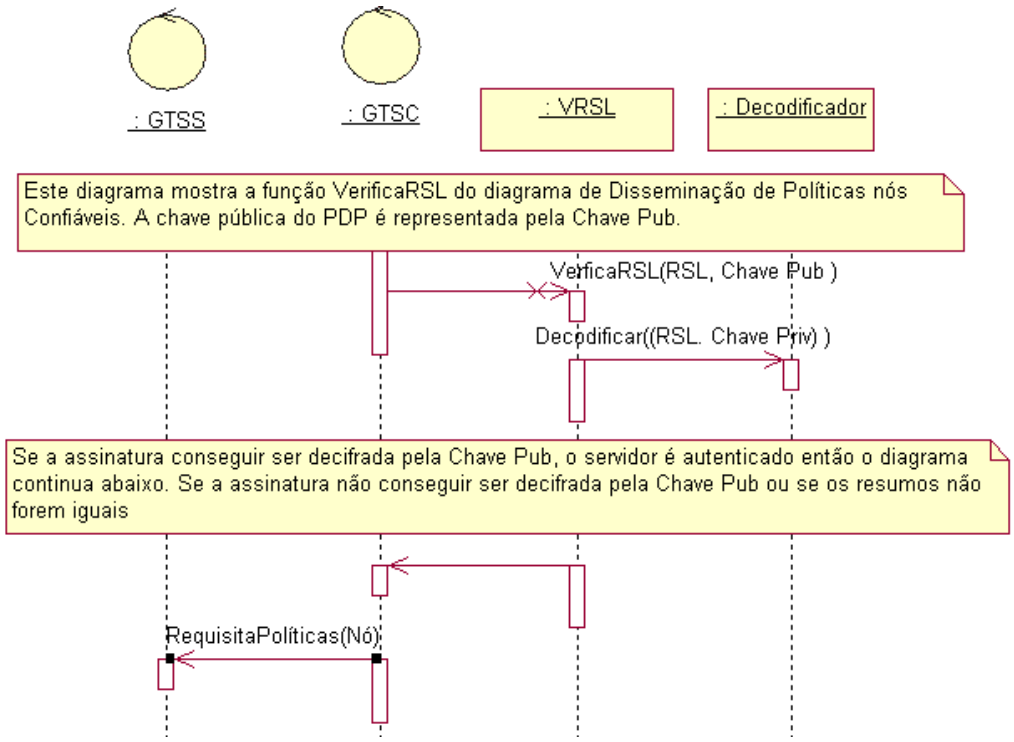


Figura 4-10: *Validação do RSL*

4.3 Considerações Gerais

Este capítulo apresentou o modelo de segurança proposto para o gerenciamento de políticas em redes ativas, mais especificamente para a arquitetura DEPMA [FERRASA 04]. Os problemas de segurança foram levantados, e devido a semelhança entre redes ativas e agentes móveis, buscou-se uma solução aplicada a agentes móveis que pudesse resolver tais problemas. A idéia foi transformar uma cápsula ativa em um repositório seguro somente leitura, onde se pode detectar qualquer tipo de adulteração, e em repositório seguro de dados direcionados, onde se pode detectar adulterações e ainda garantir a confidencialidade dos dados. Para chegar em tal resultado aplicou-se o conceito de repositórios seguros [WANGHAM 04] baseados em contêineres somente leitura [KARNIK 98].

O repositório seguro de políticas para redes ativas (RSPRA) soluciona os problemas de segurança detectados na arquitetura DEPMA, são eles:

- **Acesso não Autorizado:** Uma cápsula ativa gerada por um nó mal intencionada poderia remover, do repositório dos nós ativos (clientes), todos os objetos de política. Isso é possível se um nó mal intencionado se passar pelo servidor de políticas e enviar uma requisição de remoção de determinadas políticas aos seus nós. Este problema foi solucionado através da autenticação realizada pelo SPEP-A no RSL enviado pelo servidor. Na inicialização, o SPEP-A recebe do SPDP-A sua chave pública que servirá para verificar a assinatura do RSL e também fazer sua autenticação, desta forma impedindo tal tipo de ataque. Além disso, quando o SPDP-A requisita registro no RAI do SPEP-A, sua identidade é verificada através do RSL.
- **Adulteração de Informação:** O conteúdo da cápsula poderia ser modificado por um nó mal intencionado, modificando políticas enviadas pelo servidor aos nós ativos. Isso é possível pois enquanto um ambiente de execução (nó ativo) está executando o conteúdo de uma cápsula, o nó tem total controle sobre as informações de tal cápsula. Esta adulteração de informações é tratada pelo RSL e RSDD, esses dois repositórios possuem mecanismos para impedir este tipo de ataque.
- **Roubo de Informação:** Dados podem ser lidos (obtidos) dos RAI's por um nó mal intencionado que se passa por um nó ativo confiável da rede. Um nó mal intencionado poderia se autenticar no RAÍ do servidor de políticas e enviar um pedido de configuração de políticas para o servidor, e desta forma obter todas as informações de políticas que o servidor passa aos seus nós. O RSDD impede que este tipo de operação seja realizada, pois quando um SPEP-A requisita políticas, estas são codificadas e enviadas utilizando uma chave secreta de comunicação trocada na inicialização, de forma que só o SPEP-A original possa decifrá-las.
- **Mascaração:** Um SPEP-A mal intencionado, se passando por um SPEP-A confiável, poderia solicitar ao SPDP-A a exclusão de uma política que não utiliza mais. Este tipo de ataque é resolvido através da inclusão de uma lista de nós confiáveis da rede e da verificação do RSL. As solicitações de um SPEP-A ao SPDP-A são enviadas

através de um RSL, desta forma o SPDP-A pode fazer a autenticação do remetente através de sua chave pública armazenas no RC juntamente com a chave secreta de comunicação.

Para tornar possível a inclusão de todos os itens de segurança apresentados neste capítulo, algumas modificações foram realizadas na estrutura da arquitetura DEPMA e até mesmo na estrutura de algumas classes do ANTS. A inclusão de segurança para a criação da arquitetura RSPRA aumenta o custo de processamento por conta de controles e verificações que são demandadas. O próximo capítulo apresentará os detalhes de implementação da solução proposta e o custo extra de processamento (medido na forma de tempo e variação dos tamanhos das cápsulas) adicionado em comparação a arquitetura original.

5 Implementação da Arquitetura RSPRA e Resultados Obtidos

5.1 Introdução

Este capítulo apresenta os detalhes de implementação da arquitetura RSPRA (Repositório Seguro de Políticas para Redes Ativas) , as modificações realizadas na arquitetura DEPMA e em algumas classes do ANTS. Além disso apresenta os diversos cenários utilizados e os resultados obtidos em cada um deles.

5.2 Implementação

Assim como na arquitetura DEPMA, o ambiente de redes ativas escolhido foi o ANTS versão 2.0.2. O motivação para a escolha deste ambiente é sustentada pelo fato de que a base da arquitetura RSPRA, proposta neste trabalho, é a arquitetura DEPMA [FERRASA 04]. A versão do Java utilizada foi a 1.4.2 e o provedor de criptografia escolhido foi o Bouncy Castle [BOUNCY 05].

Vários elementos da arquitetura DEPMA e do próprio ANTS foram modificados para que o desenvolvimento do RSPRA fosse possível. Outros elementos de apoio foram incluídos para auxiliar no suporte a segurança. Nos próximos itens serão apresentadas as modificações realizadas. O anexo B ilustra o diagrama de classes da arquitetura original e as classes modificadas para criação da nova arquitetura.

5.2.1 Transformação do PAFP para PASFP

As cápsulas ativas do PAFP foram modificadas para que pudessem suportar o esquema de segurança proposto. Cada uma das cápsulas é representada por uma classe Java. As cápsulas dinâmicas foram modificadas através das seguintes classes:

1. *DF00Capsule*: esta cápsula é utilizada para efetuar o registro de uma aplicação no RAI de um determinado nó ativo, ou seja, é utilizada pelo SPDP-A para se registrar no RAI local do SPEP-A. Para tornar tal operação segura, a cápsula foi alterada para transportar um RSL. Ao chegar no destino o RSL é validado pelo GTSC do

SPEP-A destino, se a validação for bem sucedida o SPDP-A é registrado no RAI, caso contrário o registro é recusado.

2. *DF01Cápsule*: esta cápsula permite inserir novas funcionalidades em um nó ativo. As modificações deram a esta cápsula a capacidade de transportar um RSDD. Esta alteração foi necessária para garantir a integridade e confidencialidade das novas funcionalidades, enviadas pelo SPDP-A, que serão incluídas no RAI do SPEP-A. Além disso, foi criado na cápsula um *flag* que indica quando a cápsula está transportando dado criptografado ou não. Este *flag* tem o objetivo de distinguir o tratamento desta cápsula pelo GTSC (SPEP-A), ou seja, se o *flag* estiver ligado o GTSC saberá que as informações transportadas estão protegidas por um RSDD, caso contrário as informações são convencionais.

As classes que representam cápsulas comuns do PAFP também sofreram modificações, pois são responsáveis pelas solicitações e troca de informações. Originalmente existe apenas uma cápsula comum representada pela classe *CF00Capsule*. As necessidades da nova arquitetura fizeram com que esta classe sofresse modificações e além disso fosse criada uma nova cápsula comum representada pela classe *CF01Capsule*.

1. *CF00Capsule*: Esta cápsula foi modificada para transportar o RSL, RSDD, a chave pública do SPEP-A ou SPDP-A e o tipo da repositório (RSDD ou RSL). Com estas alterações, a cápsula incorporou a funcionalidade de troca de chaves e de informações utilizando os repositórios seguros. A informação sobre o tipo de repositório que a cápsula está transportando e vital para o tipo de tratamento que o GTSC (SPEP-A) dará a cápsula.

2. *CF01Capsule*: Esta cápsula foi criada especialmente para transportar a chave secreta de comunicação entre o SPDP-A e o SPEP-A. e a chave pública do servidor. É utilizada na inicialização do SPEP-A. Logo após o SPDP-A (através do GTSS) verificar que o SPEP-A é confiável, o GTSS inclui na cápsula a chave secreta de comunicação e o RSDD com a chave secreta de comunicação.

5.2.2 Modificações no SPDP-A e SPEP-A e as classes de apoio

Assim como na arquitetura original [FERRASA 04], as classes PDPImpl e PEPImpl representam respectivamente o SPDP-A e o SPEP-A. Além das modificações realizadas nestas classes, foram criadas algumas classes de apoio à segurança: RSDD, RSL, StoreKey e GeradorChave.

5.2.2.1 Classes de Apoio a Segurança

- RSDD: esta classe encapsula todas as operações realizadas em um RSDD. Possui basicamente dois métodos, o geraRSDD e verificaRSDD, que são responsáveis por gerar o RSDD e verificar o RSDD respectivamente. Esta classe possui dois construtores, o primeiro recebe uma chave pública e uma secreta e o segunda recebe uma chave privada e uma secreta. Estes construtores são invocados em diferentes momentos dependendo do tipo de operação que se quer que a classe efetue.

- RSL: semelhante ao RSDD, porém aplicado ao RSL. Encapsula todas as operações realizadas em um RSL. Possui dois métodos principais, geraRSL e verificaRSL que são responsáveis por gerar e verificar o RSL respectivamente. Possui dois construtores, um que recebe, como parâmetro de entrada, uma chave pública e o outro uma chave privada.

- StoreKey: esta classe tem o objetivo de armazenar as chaves publicas e secretas com seus respectivos SPEP-A's., representa o Repositório de Chaves (RC) da arquitetura.

- GeradorChave: o gerado de chave é fundamental para a arquitetura, esta classe é responsável por gerar as chaves pública/privada do SPEP-A e SPDP-A e a chave secreta de comunicação. Possui basicamente dois métodos, geradorChavePublica e

geradorChaveSecreta que são responsáveis respectivamente por gerar chaves pública/privada e a chave secreta de comunicação. Esta classe é invocada pelas classes PEPImpl e PDPImpl.

As classes PEPImpl e PDPImpl foram modificadas para fazer uso das classes de apoio a segurança.

5.2.2.2 Modificações da classe PEPImpl

A inclusão das novas funcionalidades atribuídas ao GTSC fizeram com que três métodos desta classe fossem modificados: initPEP(), sendMSG() e receive(). A descrição das modificações de cada um destes métodos foram as seguintes.

O método iniPEP() é responsável pela inicialização do SPEP-A. Foi incluído neste método uma chamada para a classe de apoio GeradorChave que gera as chaves pública/privada do SPEP-A. Após gerar as chaves o iniPEP() faz a armazenagem destas chaves no RC.

O método sendMSG() é responsável por montar a cápsula CF00Capsule e enviá-la ao SPDP-A. Este método foi modificado para incluir na cápsula a chave pública do SPEP-A. O método recupera a chave pública do RC e a inclui na cápsula CF00Capsule para distribuí-la ao SPDP-A.

O método receive() é responsável pelo recebimento das cápsulas ativas e por fazer o tratamento devido a cada uma delas. Este método praticamente foi reescrito para poder suportar os vários tratamentos de segurança propostos pela arquitetura. O método foi modificado para reconhecer que tipo de repositório seguro as cápsulas estão transportando e desta forma fazer o tratamento correto utilizando as classes RSDD e RSL, através da invocação dos métodos verificaRSDD e verificaRSL respectivamente. O *receive* passou a tratar dois tipos de cápsulas que não tratava na arquitetura original:

- *CF01Capsule*: Foi criado um tratamento especial para a nova cápsula *CF01Capsule* que faz a troca da chave secreta entre o SPDP-A e o SPEP-A. Após receber a

cápsula, o método verificaRSDD da classe RSDD é invocado para fazer a validação do repositório antes de armazenar a chave secreta no RC.

- *DF00Capsule*: Outra cápsula que não era tratada pelo método receive() é a cápsula *DF00Capsule*. Na arquitetura original esta cápsula acessava diretamente o método register() da classe RAI para fazer o registro do SPDP-A, o SPEP-A não a recebia. Na nova arquitetura este fluxo mudou, a cápsula é enviada ao SPEP-A que faz a autenticação do servidor através da verificação do RSL (método verificaRSL da classe RSL). Se o repositório for validado o SPEP-A faz a chamada ao método register() do RAI, caso contrário a solicitação de registro é ignorada.

5.2.2.3 Modificações da classe PDPImpl

A inclusão do modelo de segurança no SPDP-A, na forma do GTSS, resultou na modificação de 4 métodos desta classe, são eles: initPDP(), careOPN(), sendMSG(),e addClass().

Foi incluído no método initPDP() um gerador de chaves pública/privada, assim como no PEPImpl. Na inicialização do SPDP-A, as chaves pública/privada são geradas e armazenadas para utilização futura. O initPDP() invoca o método GeradorChavePublica() da classe GeradorChave para fazer a geração.

O método careOPN() foi o que sofreu mais modificações, pois trata as requisições de abertura de conexão com o SPDP-A. Este método é invocado pelo método *receive* do SPDP-A, ou seja, quando uma cápsula ativa chega com um pedido de conexão ao SPDP-A, o método *receive* repassa o pedido para o método careOPN(). Na arquitetura original, a conexão era concedida para o SPEP-A sem nenhum tipo de controle e/ou verificação. Na nova arquitetura, foi incluída a verificação do SPEP-A que checa se este nó é confiável ou não. Dependendo do resultado da verificação, o tratamento dado ao SPEP-A é diferenciado. Se o SPEP-A for confiável, o SPDP-A gerará um chave secreta de comunicação, invocando o método geradorChaveSecreta() da classe GeradorChave, e armazenará a chave secreta e a pública do SPEP-A no RC através da classe StoreKey. Em seguida, o SPDP-A monta um RSDD com a

chave pública do servidor e a chave secreta de comunicação e envia para o SPEP-A utilizando a cápsula *CF01Capsule*. Após fazer a troca de chaves, o SPDP-A monta um RSL e envia um pedido de registro no RAI do SPEP-A utilizando a cápsula *DF00Capsule*. Se o nó não for confiável o SPDP-A envia, da forma tradicional, o pedido de registro no RAI do SPEP-A solicitante sem nenhum tipo de validação de segurança. A montagem dos repositórios e envio das cápsulas são realizados pelo método sendMSG().

O método sendMSG() foi modificado para reconhecer o tipo de repositório de dados que irá trafegar na cápsula e invocar o método de encapsulamento dos dados dependendo desta informação. Este método é responsável por invocar as rotinas de montagem dos repositórios RSL e RSDD (através dos métodos geraRSL e geraRSDD das classes RSL e RSDD respectivamente) e pela montagem da cápsula de transporte. Após montado a cápsula com o respectivo repositório, o sendMSG() envia-a para o destino.

A classe addClass() é invocada logo após a inserção/modificação de políticas, realizada pelo FGP. Tem o objetivo de disseminar esta inserção/modificação para os nós afetados. A modificação realizada foi a inclusão da chamada ao método geraRSDD que fará a inclusão das políticas no RSDD para ser enviada ao SPEP-A, caso o nó em questão seja confiável.

5.3 Metodologia de Análise

A solução utilizada neste trabalho foi primeiramente aplicada à agentes móveis [WANGHAM 04], como mostrado no capítulo 3, e adaptada a realidade de redes ativas com o intuito de solucionar os problemas da arquitetura DEPMA. Para demonstrar o resultado do trabalho, optou-se por comparar alguns parâmetros entre a arquitetura original DEPMA e a arquitetura RSPRA. Basicamente, os parâmetros comparados foram tamanha das cápsulas e tempos de processamento extra (processamento utilizado para prover segurança).

Nas simulações utilizando o RSPRA, o algoritmo de criptografia simétrica e o algoritmo de assinatura dos dados foram modificados para que pudéssemos compará-los entre si. Os algoritmos de criptografia simétrica utilizados foram: DES [SCHNEIER 04], AES [MATHIAS 06] , 3DES [BERKELEY 05], BlowFish [SCHNEIER 06] e o Idea [CRIPTO 05]. Os algoritmos de assinatura utilizados foram MD5 e SHA-2 [COLOURIS 00]. O algoritmo de

criptografia assimétrica escolhido foi a RSA [COUTINHO 97]. O Anexo A traz algumas informações adicionais sobre os algoritmos utilizados neste trabalho.

5.3.1 Ambiente de Teste

Os testes foram realizados no laboratório LASD do CEFET. Foram utilizados também três computadores desta rede: Dominique, Asterix e Wireless1. Os computadores possuem as seguintes configurações:

- Dominique: Pentium 4 com 3.2 Ghz de processador e 448 MegaBytes de memória RAM. Possui uma placa de rede com endereço IP 172.16.5.211, o sistema operacional linux, a máquina virtual Java versão 1.4.2-11 e o ANTS versão 2.0.2 instalado.
- Wireless1: AMD Athlon XP 2400 com 2.0 Ghz de processador e 512 MegaBytes de memória RAM. Possui uma placa de rede com endereço IP 192.168.2.5, o sistema operacional linux, a máquina virtual Java versão 1.4.2-11 e o ANTS versão 2.0.2 instalado.
- Asterix: Pentium com 133 Mhz de processador e 32 MegaBytes de memória RAM. Possui duas placa de rede, a eth0 e eth1 com os respectivos endereços IP's 172.16.5.70 e 192.168.2.2. O sistema operacional instalado é o linux, a versão da máquina virtual Java é 1.2.2 e possui o ANTS versão 2.0.2 instalado.

A arquitetura da rede utilizada é ilustrada pela figura 5.1

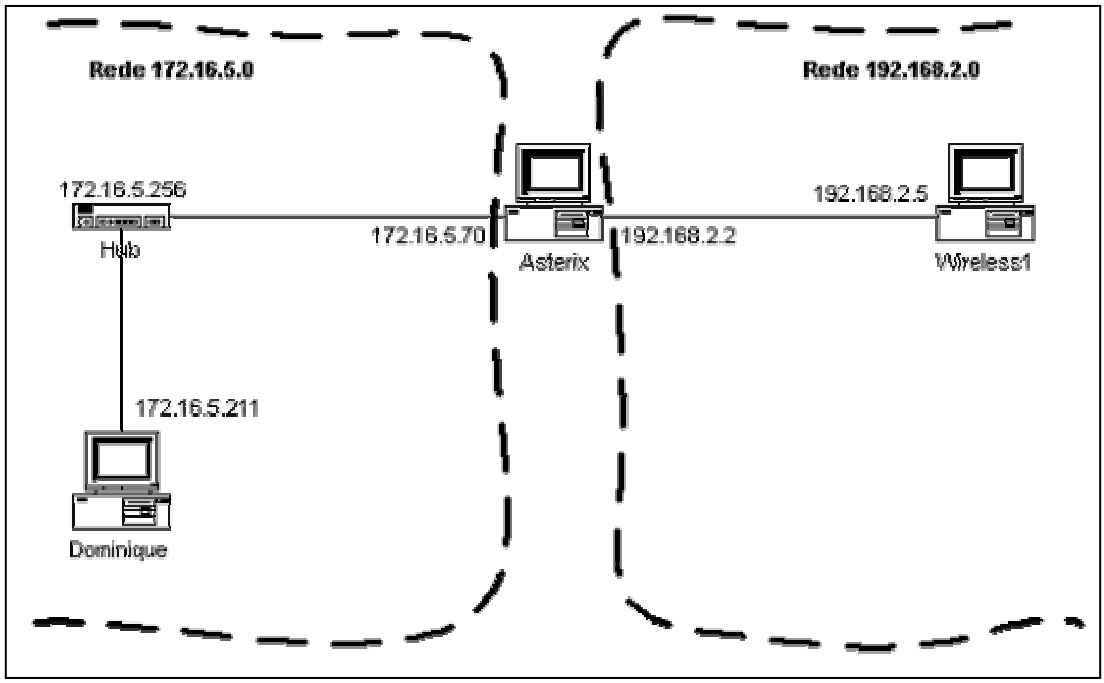


Figura 5-1: *Arquitetura de rede utilizada para os testes*

Nesta arquitetura, o máquina Asterix foi utilizada como roteador, é ela quem faz a ligação entre a rede 192.168.2.0 e 172.16.5.0. A máquina Dominique está na rede 172.16.5.0, que é a rede do LASD. Foi criada uma nova rede (192.168.2.0) na qual foi inserida a máquina Wireless1. A tabela de roteamento do Asterix foi modificada para que este pudesse fazer o roteamento dos pacotes trocados entre as máquinas Domique e Wireless1.

Na máquina Dominique foi instalado os fontes do SPEP-A (local de aplicação de políticas). O SPDP-A (servidor de políticas) foi instalado na máquina Wireless1 e o roteador ativo instalado na máquina Asterix. Os componentes da arquitetura são apresentados na figura 5.2.

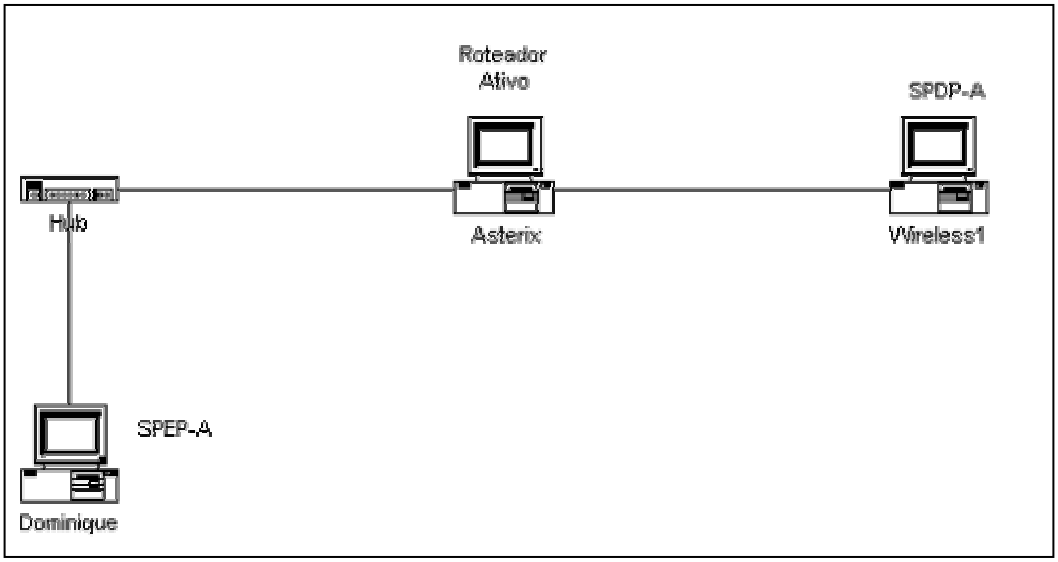


Figura 5-2: *SPDP-A, SPEP-A e Roteador Ativo*

5.3.2 Cenário de Simulação

A arquitetura foi configurada para repetir sempre um mesmos conjunto de eventos, para que pudéssemos fazer as comparações sempre nas mesmas condições. Desta forma, qualquer variação nas medições, provocada pelas mudanças de algoritmo, ficariam evidentes.

O cenário escolhido para os testes é descrito da seguinte forma:

1. O SPEP-A e o SPDP-A são inicializados. Neste momento cada um cria sua chave pública/privada.
2. O SPEP-A faz pedido de conexão do SPDP-A, enviando sua chave pública.
3. Como o nó SPEP-A é confiável, o SPDP-A cria uma chave secreta de comunicação e à armazena juntamente com o IP do SPEP-A e sua chave pública.
4. O SPDP-A criará um RSDD, armazenando a chave secreta criada e sua chave pública, encapsula o repositório e envia ao SPEP-A.
5. Ao receber a cápsula, o SPEP-A verifica a segurança no RSDD, extrai a chave secreta de comunicação e a armazena.
6. O SPEP-A requisita as políticas ao SPDP-A.
7. O SPDP-A envia um RSL para o SPEP-A com seu IP requisitando registro ao seu RAI.
8. O SPEP-A verifica o RSL e faz o registro do SPDP-A em seu RAI.

9. O SPDP-A recupera as políticas ligadas ao SPEP-A em questão, cria um RSDD e envia ao requisitante.
10. Ao receber a cápsula, o SPEP-A faz as verificações de segurança no RSDD, extrai e aplicará as políticas.

A figura 5.3 apresenta a seqüência de 1 à 10.

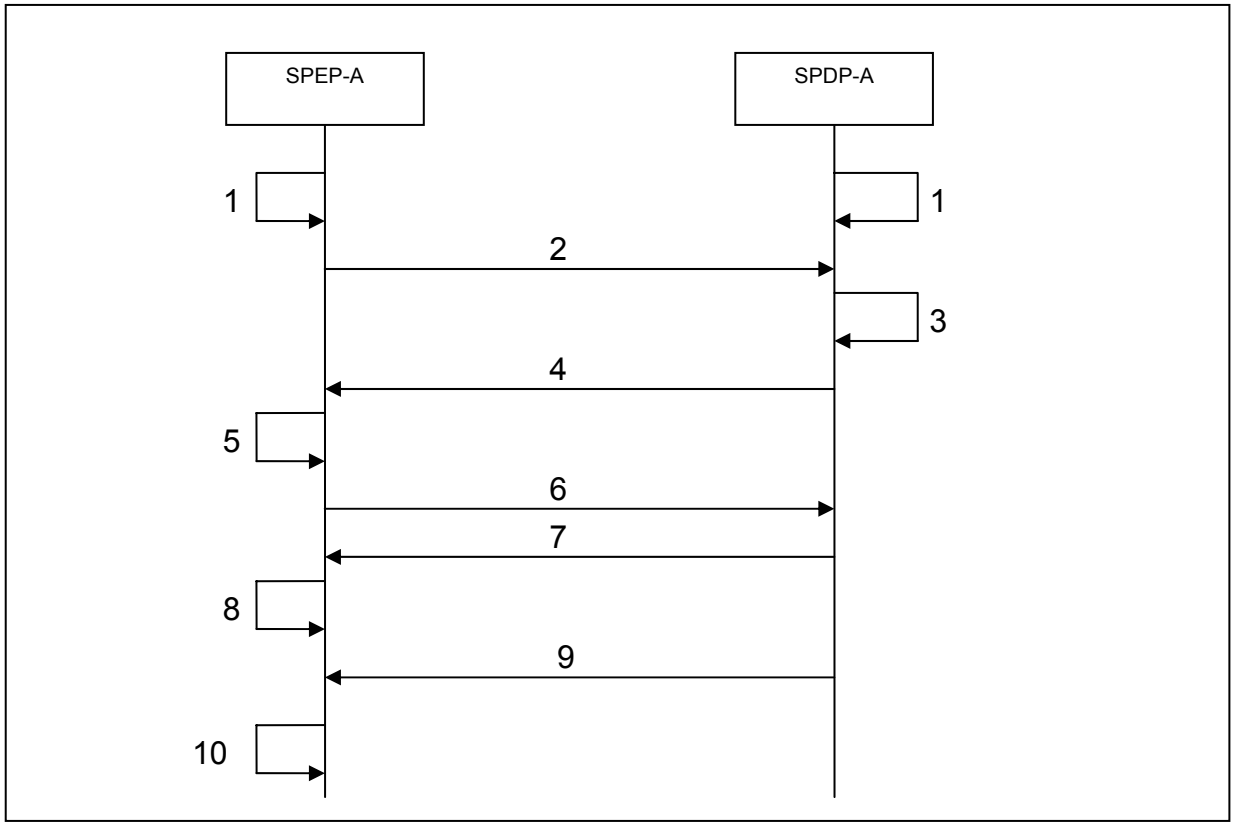


Figura 5-3: *Eventos do cenário de teste*

5.3.3 Descrição dos Testes Realizados

Os testes consistem na medição dos seguintes parâmetros: tamanho das cápsulas trocadas entre o SPEP-A e o SPDP-A, os tempos de criação das chaves pública/privada e da chave secreta de comunicação e os tempos de geração e verificação do RSL e do RSDD.

Em primeiro lugar executou-se a arquitetura DEPMA original, onde o tamanho das cápsulas foram anotados. Os tempos de geração de chaves e criação/verificação de repositórios não foram medidos, pois a arquitetura original não possui tais recursos.

Em seguida a arquitetura RSPRA foi executada, variando-se os algoritmos de geração de chave secreta, assinatura/verificação e codificação/decodificação. Os algoritmos de criptografia simétrica utilizados foram: DES, AES, 3DES, BlowFish e o Idea. Os algoritmos de assinatura utilizados foram MD5 e SHA-2. O único algoritmo de criptografia assimétrica utilizada foi o RSA, que não foi comparado a outros devido ao fato de não ter sido encontrado na literatura outro algoritmo que pudesse substituí-lo de forma a atender as necessidades da arquitetura proposta. A chave de 2048 bits utilizada no RSA é extremamente grande. A razão para a utilização desta chave deu-se pelo fato de que o RSA não consegue cifrar um bloco de dados que seja maior do que o tamanho de sua chave [BOUNCY 05].

5.3.4 Resultados

Para fazermos comparação entre a arquitetura original e a arquitetura resultante da inclusão de segurança (RSPRA), optou-se por medir o tamanho das cápsulas da arquitetura original e da nova arquitetura. Além disso, serão medidos os tempos acrescidos no processo original para prover segurança. O cenário foi executado com a nova arquitetura combinando algoritmos de segurança diferentes para que pudéssemos compará-los entre si. Portanto os testes possuem três objetivos, o primeiro é comparar o tamanho das cápsulas da arquitetura original com a arquitetura RSPRA. O segundo objetivo é medir o tempo que foi inserido no processo com a inclusão de segurança, e o terceiro objetivo é comparar a arquitetura RSPRA com vários algoritmos de segurança diferentes.

5.3.4.1 Resultados da Arquitetura DEPMA (Original)

A arquitetura DEPMA foi executada com o cenário descrito acima para podermos medir o tamanhos das cápsulas trocadas entre o SPEP-A e o SPDP-A. Os resultados obtidos são apresentados na tabela 1.

Cápsulas	Função	Tamanho(Bytes)
SPEP-A->OPN->SPDP-A	Cápsula que solicita abertura de conexão	52
SPEP-A->REQ->SPDP-A	Cápsula que requisita políticas	116
SPDP-A->DEC->SPEP-A	Cápsula que transporta as políticas	228

Tabela 1: *Tamanho (em Bytes) das cápsulas da arquitetura DEPMA*

5.3.4.2 Resultados da Arquitetura RSPRA

Na arquitetura RSPRA, além do tamanho das cápsulas modificadas e incluídas, foram medidos os tempos de geração de chave secreta de comunicação do SPDP-A, a geração e verificação do RSDD e RSL e o tempo de geração da chave pública/privada tanto no SPEP-A como no SPDP-A. Em todos os casos, o algoritmo de criptografia assimétrica é o RSA com chave de 2048 bits.

Cápsulas	Função	Tamanho(Bytes)
SPEP-A->OPN->SPDP-A	Cápsula que solicita abertura de conexão e transporta a chave pública do SPEP-A	756
SPEP-A->REQ->SPDP-A	Cápsula que requisita políticas	116
SPDP-A->CF01->SPEP-A	Cápsula que envia ao SPEP-A a chave pública do SPDP-A e o RSDD com a chave secreta de comunicação.	991
SPDP-A->DF00->SPEP-A	Cápsula que envia ao SPEP-A requisição para registro em seu RAI local.	303
SPDP-A->CF00->SPEP-A	Cápsula que envia as políticas ao SPEP-A	2014
PDP->DF01->PEP	Cápsula que envia um RSDD ao SPEP-A quando o administrador da rede cria uma nova política	449

Tabela 2: *Tamanho das cápsula RSPRA, utilizando os algoritmos DES, 3DES, IDEA e BlowFish com assinatura MD5 e SHA2*

Cápsulas	Função	Tamanho(Bytes)
SPEP-A->OPN->SPDP-A	Cápsula que solicita abertura de conexão e transporta a chave pública do SPEP-A	756
SPEP-A->REQ->SPDP-A	Cápsula que requisita políticas	116

SPDP-A->CF01->SPEP-A	Cápsula que envia ao SPEP-A a chave pública do SPDP-A e o RSDD com a chave secreta de comunicação.	991
SPDP-A->DF00->SPEP-A	Cápsula que envia ao SPEP-A requisição para registro em seu RAI local.	303
SPDP-A->CF00->SPEP-A	Cápsula que envia as políticas ao SPEP-A	2030
PDP->DF01->PEP	Cápsula que envia um RSDD ao SPEP-A quando o administrador da rede cria uma nova política	449

Tabela 3: *Tamanho das cápsulas RSPRA, utilizando o algoritmo AES com assinatura MD5 e SHA2*

Na medição do tamanho das cápsulas, constatou-se que não houve nenhuma modificação com a variação dos algoritmos de criptografia/assinatura, a não ser no caso do algoritmo de criptografia AES. Isso se dá pelo fato de que quem dita o tamanho da cápsula, nestes casos, é o tamanho da chave pública/privada RSA (2048 bits), como o tamanho desta chave não variou o tamanho das cápsulas acompanhou esta tendência. A variação do algoritmo de assinatura (MD5 e SHA2) não provocou nenhuma modificação no tamanho das cápsulas.

5.3.4.2.1 Geração da Chave Pública/Privada RSA 2048 bits

A chave RSA é gerada em dois momentos distintos na execução do cenário, a primeira na inicialização do SPEP-A e a segunda na inicialização do SPDP-A. A chave RSA é utilizada como chave pública e privada dos elementos da arquitetura, onde a chave pública é distribuída para outros nós da rede e a chave privada e armazenada para que nenhum outro nó a conheça. Os tempos de geração destas chaves foram colhidos em todas as execuções do cenário, independente da mudança de algoritmos de cifra e assinatura, pois a modificação destes não afeta a geração desta chave. Desta forma obteve-se 120 leituras de tempo para cada um dos elementos (SPEP-A e SPDP-A). A tabela com os 120 valores coletados encontra-se no Anexo C.

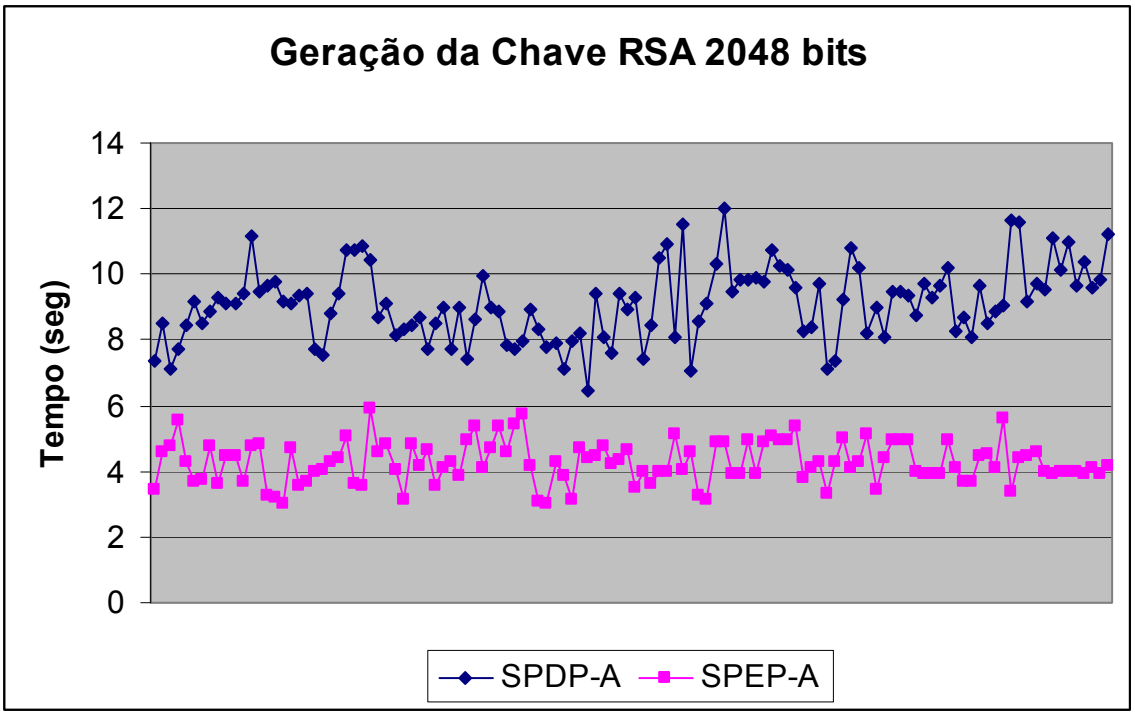


Figura 5-4: *Tempos de geração da chave RSA de 2048 bits*

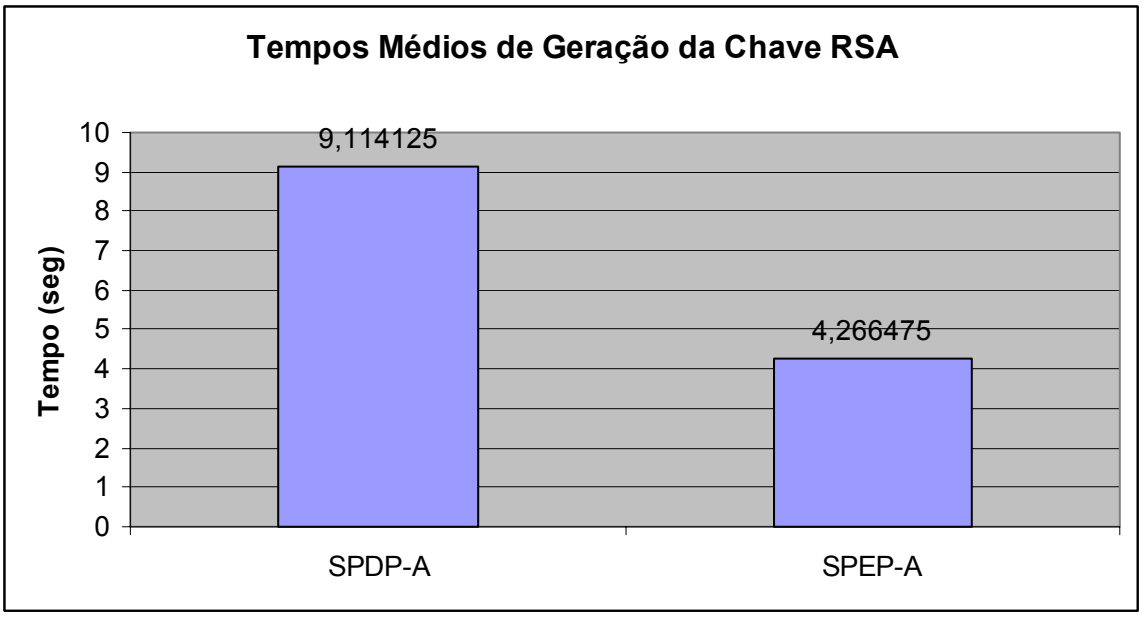


Figura 5-5: *Tempos Médios de Geração da Chave RSA no SPDP-A E SPEP-A*

A tarefa de geração da chave RSA foi mais rápida no SPEP-A. Isto se deu pelo fato da máquina Dominique (SPEP-A) ter uma configuração mais potente do que a máquina Wireless1 (SPDP-A).

5.3.4.2.2 Geração da Chave Secreta de Comunicação

A chave secreta é utilizada pelo SPEP-A e pelo SPDP-A para cifrar os dados trocados entre eles. Esta chave é utilizada, ao invés da chave pública/privada RSA, devido ao custo de processamento da codificação/decodificação ser menor com a chave secreta (criptografia simétrica) do que com a chave pública/privada (criptografia assimétrica) [COLOURIS 00]. Na arquitetura RSPRA, o responsável por gerar a chave secreta de comunicação é o SPDP-A. Os tempos obtidos foram:

	DES 56 bits	**AES 128 bits**	**BlowFish 64 bits**	**IDEA 128 bits**	**BlowFish 128 bits**	**3DES 128 bits**
1	2,481	2,731	2,535	2,468	2,485	2,481
2	2,831	2,47	2,471	2,426	2,470	2,492
3	2,492	2,466	2,470	2,490	2,464	2,513
4	2,489	2,466	2,467	2,494	2,470	2,488
5	2,492	2,470	2,486	2,491	2,461	2,516
6	2,491	2,437	2,509	2,491	2,505	2,519
7	2,499	2,470	2,504	2,490	2,468	2,511
8	2,480	2,464	2,475	2,461	2,485	2,487
9	2,500	2,473	2,503	2,462	2,466	2,488
10	2,454	2,466	2,466	2,810	2,468	2,519
11	2,492	2,468	2,470	2,632	2,476	2,521
12	2,520	2,469	2,500	2,457	2,468	2,488
13	2,497	2,461	2,493	2,489	2,461	2,491
14	2,496	2,433	2,481	2,490	2,473	2,522
15	2,511	2,466	2,472	2,462	2,473	2,485
16	2,511	2,469	2,475	2,424	2,474	2,517
17	2,485	2,465	2,506	2,461	2,467	2,488
18	2,491	2,495	2,504	2,494	2,472	2,480
19	2,481	2,464	2,507	2,500	2,496	2,489
20	2,704	2,458	2,604	2,632	2,853	2,896
Tempos Médios						
	2,51985	**2,47805**	**2,4949**	**2,5062**	**2,49275**	**2,51955**

Tabela 4: *Tempos de Geração da Chave Secreta de Comunicação*

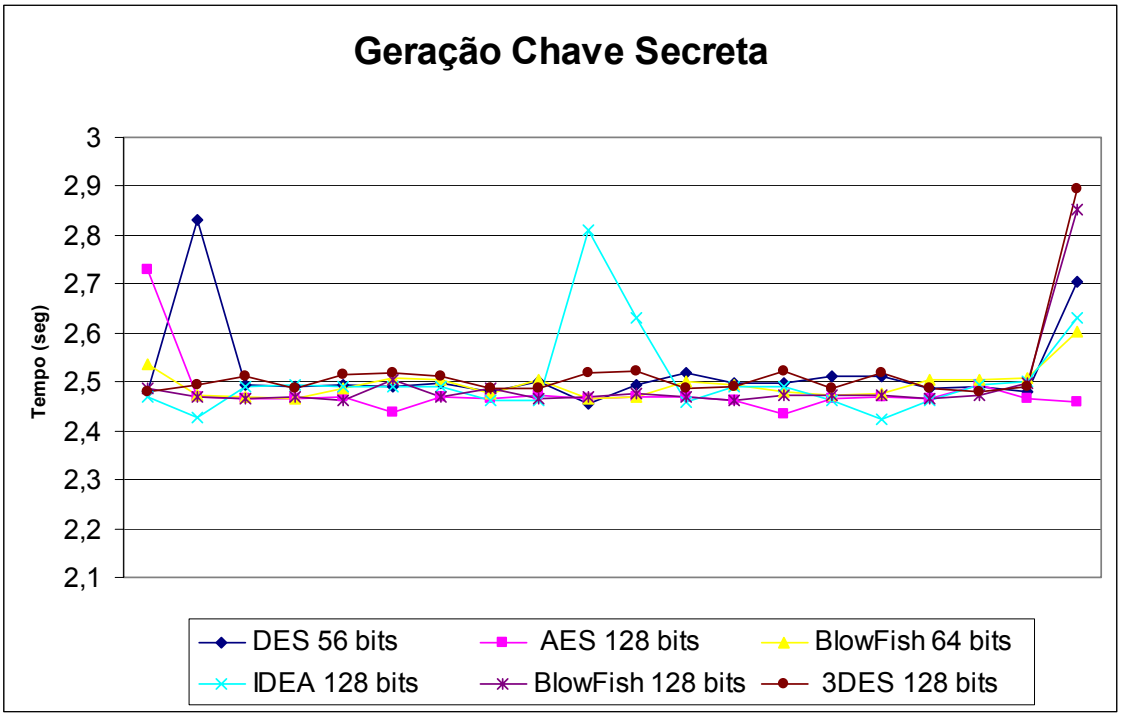


Figura 5-6**:** *Comparativo entre os Tempos de Geração da Chave Secreta de Comunicação*

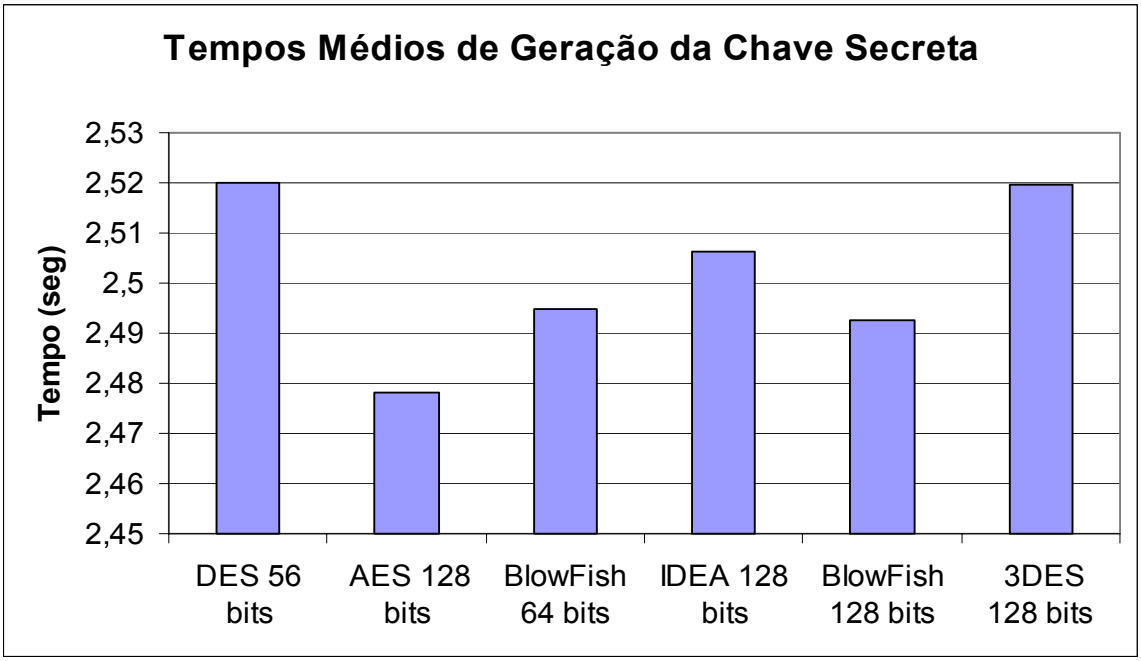


Figura 5-7: *Tempos Médios de Geração da Chave Secreta*

5.3.4.2.3 Geração do Repositório Somente Leitura (RSL)

O RSL consiste na assinatura, encapsulamento e envio dos dados ao destino de forma a detectar qualquer alteração realizada nos dados durante o percurso da cápsula. O RSL é gerado pelo SPDP-A, que utilizou vários algoritmos de assinatura e tipos de chaves para executar o cenário e obter tempos para comparação. Os algoritmos de assinatura utilizados foram o MD5 e SHA-2, e os tipos de chaves foram DES de 56 bits, AES de 128 bits, 3DES de 128 bits, BlowFish de 64 e 128 bits e IDEA de 128 bits. Os tempos obtidos foram:

	SHA-2						**MD5**					
	DES 56	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**	**DES 56**	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**
1	0,907	0,75	0,743	0,714	0,506	0,673	0,462	0,636	0,518	0,534	0,464	0,914
2	0,701	0,53	0,552	0,538	0,695	0,702	0,798	0,636	0,631	0,677	0,637	0,486
3	0,725	0,535	0,805	0,523	0,737	0,77	0,616	0,626	0,646	0,701	0,47	0,793
4	0,696	0,68	0,766	0,765	0,689	0,802	0,458	0,634	0,644	0,695	0,467	0,636
5	0,755	0,686	0,867	0,692	0,533	0,678	0,609	0,619	0,645	0,542	0,637	0,641
6	0,866	0,707	0,705	0,534	0,745	0,693	0,791	0,630	0,637	0,999	0,476	0,465
7	0,699	0,798	0,695	0,522	0,693	0,734	0,620	0,630	0,631	0,711	0,634	0,475
8	0,709	0,782	0,641	0,695	0,677	0,756	0,623	0,621	0,721	0,696	0,641	0,628
9	0,695	0,696	0,701	0,688	0,704	0,704	0,451	0,636	0,652	0,680	0,477	0,671
10	0,530	0,716	0,952	0,631	0,696	0,637	0,617	0,469	0,492	0,683	1,102	0,656
Tempos Médios												
	0,728	**0,688**	**0,742**	**0,630**	**0,667**	**0,715**	**0,605**	**0,613**	**0,621**	**0,691**	**0,600**	**0,636**

Tabela 5: *Tempos de Geração do RSL*

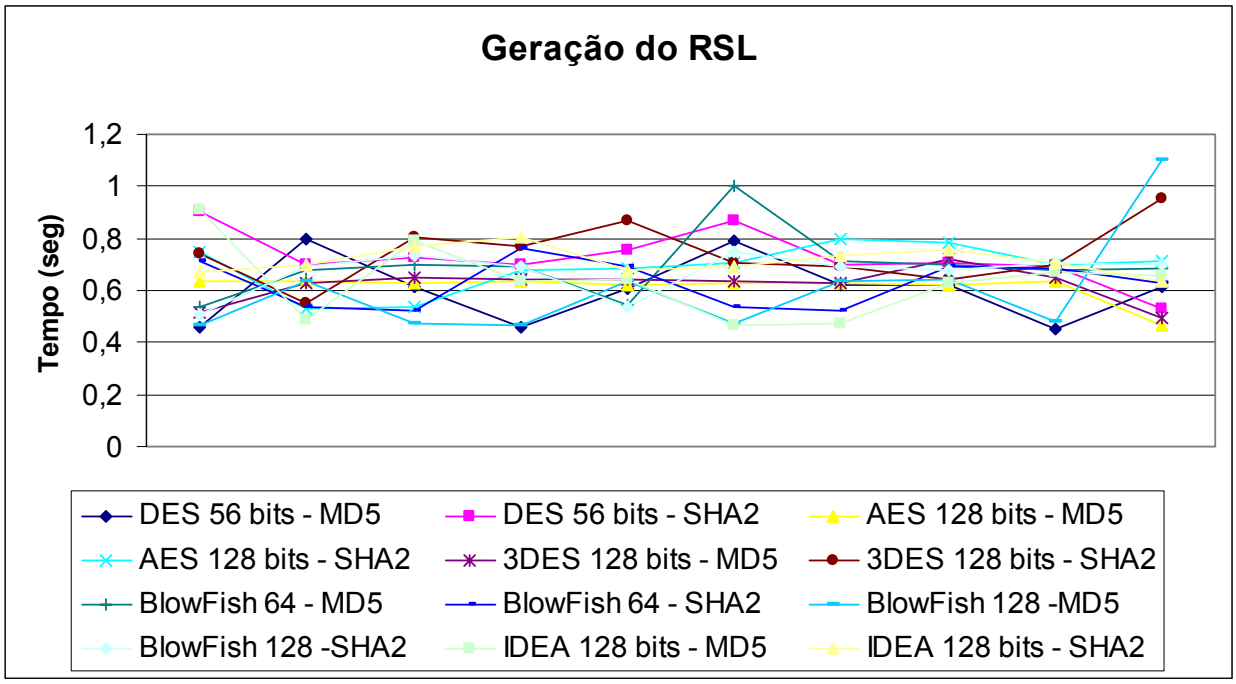


Figura 5-8: *Comparativo entre os Tempos de Geração do RSL*

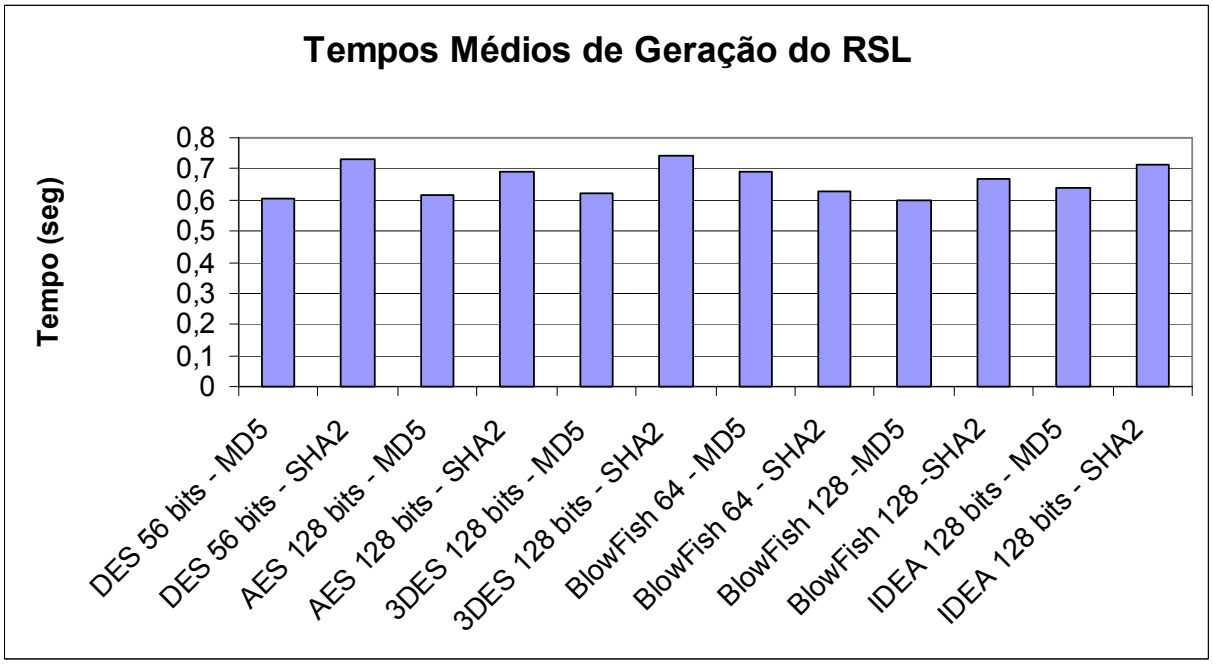


Figura 5-9: *Tempos Médios da Geração do RSL*

5.3.4.2.4 Geração do Repositório Seguro de Dados Direcionados(RSDD)

O RSDD consiste na assinatura e criptografia dos dados, encapsulamento e envio dos dados ao destino de forma a não permitir que qualquer outro nó, que não seja o destino, tome conhecimento do conteúdo do repositório. O RSDD é gerado pelo SPDP-A. Foram utilizados vários algoritmos de assinatura e de criptografia simétrica para executar o cenário e obter tempos para comparação. Os algoritmos de assinatura utilizados foram o MD5 e SHA-2, e os algoritmos de criptografia simétrica / geração de chave secreta foram DES de 56 bits, AES de 128 bits, 3DES de 128 bits, BlowFish de 64 e 128 bits e IDEA de 128 bits. Os tempos obtidos foram:

	SHA-2						**MD5**					
	DES 56	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**	**DES 56**	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**
1	0,390	0,399	0,357	0,776	0,563	0,749	0,665	0,25	0,499	0,676	0,521	0,292
2	0,566	0,72	0,545	0,765	0,884	0,564	0,688	0,451	0,683	0,694	0,717	0,307
3	0,586	0,947	0,838	0,764	0,535	0,384	0,724	0,63	0,707	0,541	0,95	0,277
4	0,799	0,794	0,964	0,793	0,778	0,563	0,685	0,449	0,694	0,458	0,773	0,279
5	0,822	0,794	0,684	0,799	0,563	0,646	0,677	0,65	0,702	0,682	0,751	0,458
6	0,822	0,864	0,643	0,778	0,756	0,693	0,678	0,814	0,853	0,463	0,584	0,669
7	0,577	0,821	0,964	0,774	0,784	0,391	0,686	0,432	0,728	0,69	0,752	0,463
8	0,722	0,786	0,871	0,781	0,773	0,362	0,681	0,806	0,682	0,676	0,698	0,47
9	0,742	0,788	0,912	0,758	0,785	0,372	0,659	0,661	0,618	0,692	0,775	0,684
10	0,601	0,817	0,748	0,531	0,813	0,381	0,682	0,63	0,728	0,289	0,763	0,915
Tempos Médios												
	0,663	**0,773**	**0,753**	**0,752**	**0,723**	**0,511**	**0,683**	**0,577**	**0,689**	**0,586**	**0,728**	**0,481**

Tabela 6: *Tempos de Geração do RSDD*

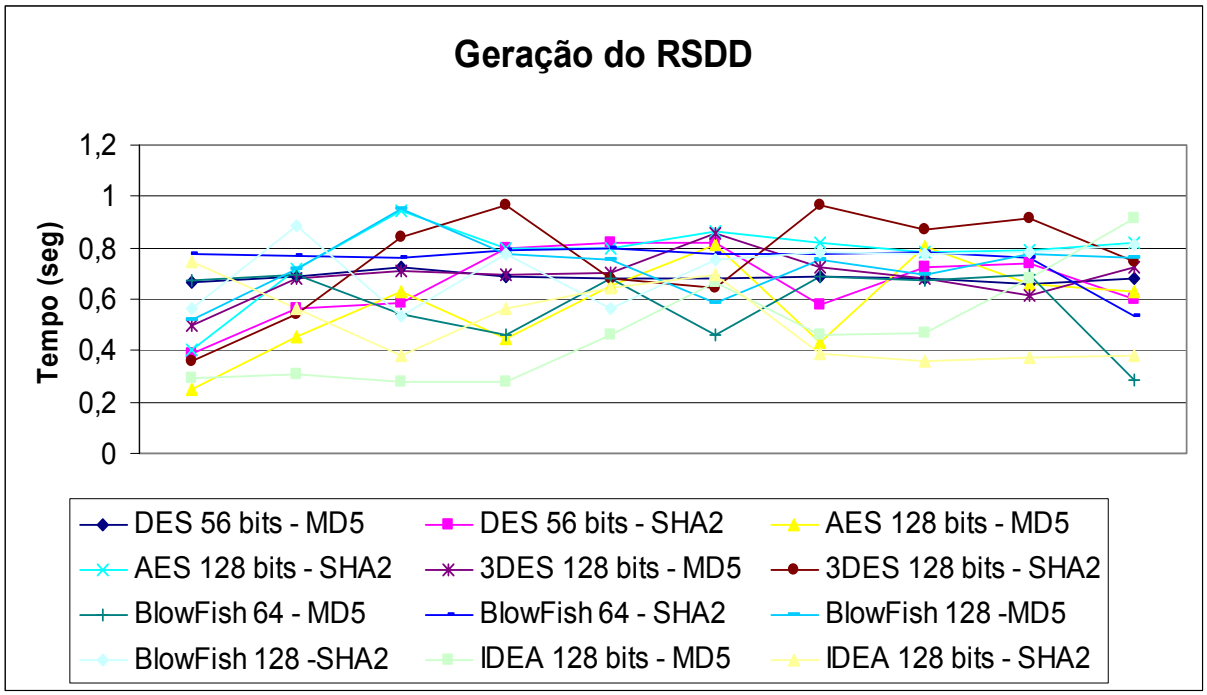


Figura 5-10: *Gráfico com os Tempos de Geração do RSDD*

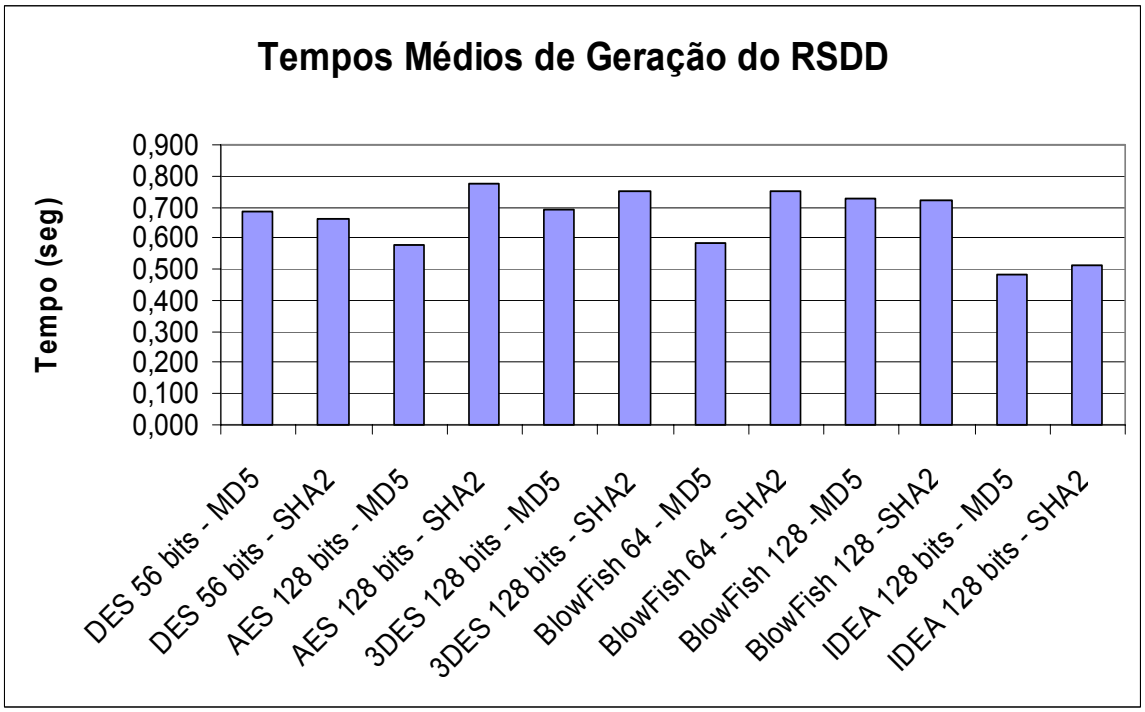


Figura 5-11: *Tempos Médios de Geração do RSDD*

5.3.4.2.5 Validação do Repositório Somente Leitura (RSL)

A validação do RSL ocorre no SPEP-A e foram realizados os mesmos critérios de sua geração. Os tempos obtidos foram:

	SHA-2						MD5					
	DES 56	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**	**DES 56**	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**
1	0,299	0,293	0,293	0,341	0,224	0,332	0,240	0,422	0,580	0,431	0,564	0,355
2	0,885	0,672	0,247	0,202	0,261	0,277	0,236	0,250	0,255	0,248	0,256	0,275
3	0,885	0,239	0,624	0,286	0,262	0,341	0,253	0,238	0,273	0,249	0,253	0,353
4	0,303	0,248	0,504	0,368	0,261	0,266	0,238	0,258	0,239	0,243	0,718	0,481
5	0,235	0,303	0,395	0,401	0,269	0,324	0,246	0,242	0,250	0,242	0,380	0,257
6	0,224	0,656	0,288	0,544	0,266	0,316	0,242	0,239	0,270	0,240	0,796	0,735
7	0,241	0,248	0,677	0,552	0,264	0,403	0,243	0,247	0,430	0,248	0,426	0,243
8	0,241	0,250	0,355	0,577	0,262	0,312	0,246	0,250	0,878	0,508	0,749	0,256
9	0,243	0,480	0,255	0,267	0,263	0,356	0,235	0,257	0,878	0,607	0,251	0,703
10	0,226	0,672	0,251	0,462	0,259	0,373	0,239	0,241	0,235	0,355	0,527	0,239
Tempos Médios												
	0,378	**0,406**	**0,389**	**0,400**	**0,259**	**0,330**	**0,242**	**0,264**	**0,429**	**0,337**	**0,492**	**0,390**

Tabela 7: *Tempos de Validação do RSL pelo SPEP-A*

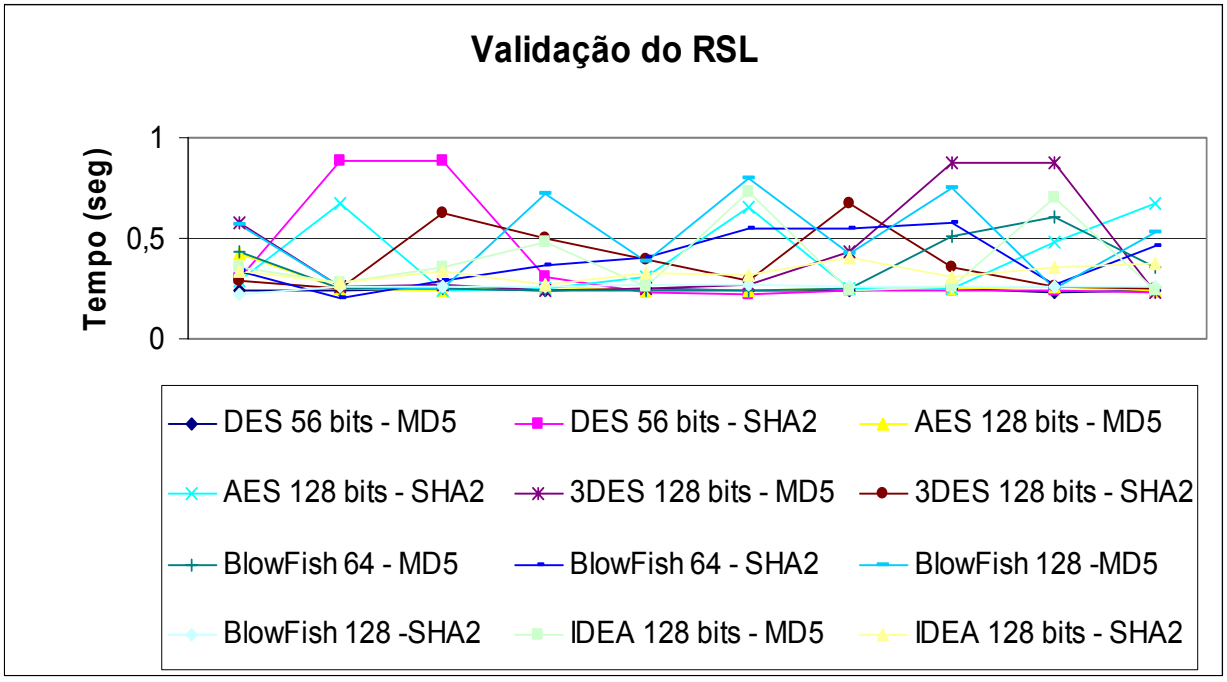


Figura 5-12: *Gráfico com os Tempos de Validação do RSL*

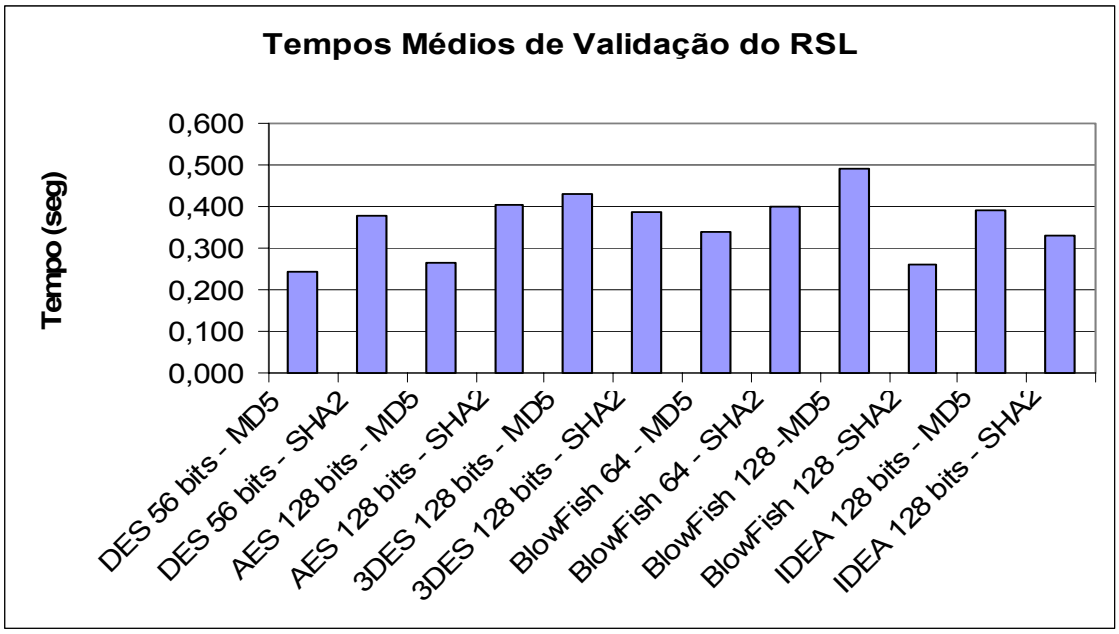


Figura 5-13: *Tempos Médios de Validação do RSL pelo SPEP-A*

5.3.4.2.6 Validação do Repositório Seguro de Dados Direcionados (RSDD)

A validação do RSDD ocorre no SPEP-A e foram realizados os mesmos critérios de sua geração. Os tempos obtidos foram:

	SHA-2						**MD5**					
	DES 56	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**	**DES 56**	**AES 128**	**3DES 128**	**BlowFish 64**	**BlowFish 128**	**IDEA 128**
1	0,387	0,240	0,220	0,148	0,243	0,245	0,377	0,423	0,148	0,140	0,139	0,181
2	0,365	0,479	0,158	0,165	0,205	0,173	0,379	0,420	0,146	0,131	0,398	0,225
3	0,806	0,177	0,172	0,164	0,244	0,206	0,381	0,429	0,134	0,137	0,129	0,159
4	0,407	0,445	0,149	0,205	0,150	0,171	0,351	0,418	0,147	0,121	0,172	0,304
5	0,378	0,393	0,424	0,186	0,179	0,145	0,349	0,435	0,144	0,172	0,439	0,157
6	0,376	0,367	0,161	0,202	0,229	0,270	0,349	0,428	0,132	0,127	0,551	0,185
7	0,376	0,175	0,226	0,179	0,177	0,151	0,350	0,427	0,135	0,134	0,551	0,177
8	0,370	0,175	0,143	0,221	0,183	0,224	0,350	0,426	0,137	0,163	0,143	0,116
9	0,376	0,208	0,149	0,178	0,178	0,233	0,360	0,429	0,145	0,137	0,129	0,134
10	0,378	0,419	0,140	0,264	0,173	0,245	0,349	0,431	0,146	0,273	0,139	0,123
Tempos Médios												
	0,422	**0,308**	**0,194**	**0,191**	**0,196**	**0,206**	**0,360**	**0,427**	**0,141**	**0,154**	**0,279**	**0,176**

Tabela 8: *Tempos de Verificação do RSDD pelo SPEP-A*

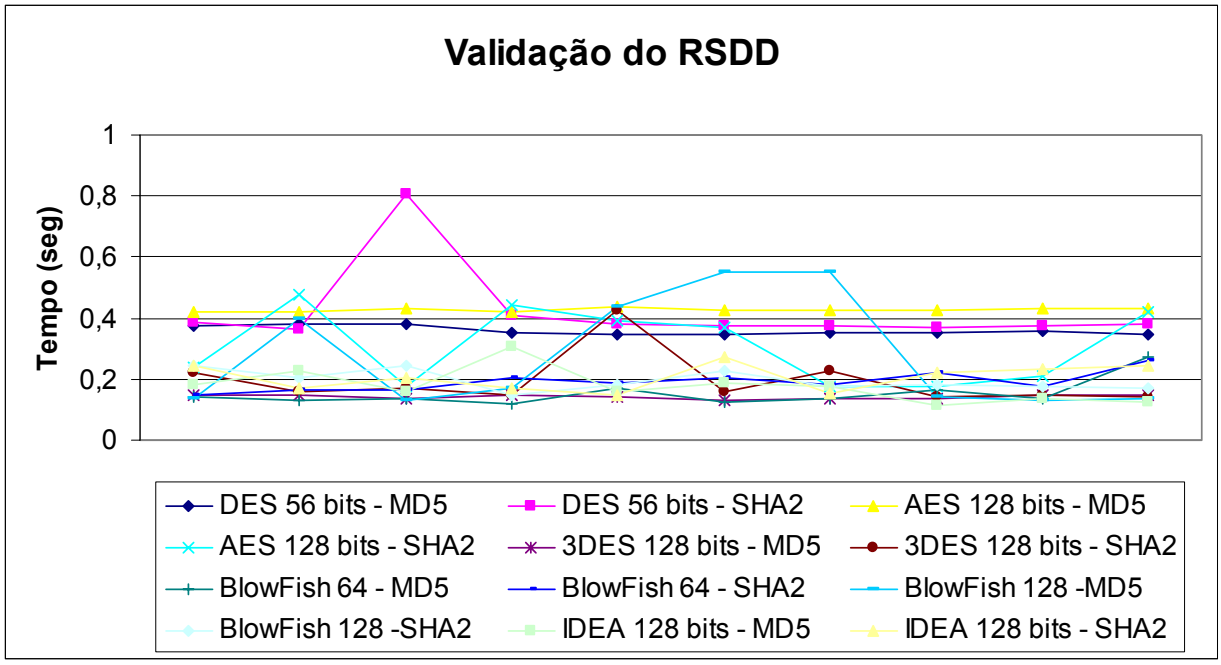


Figura 5-14: *Gráfico dos Tempos de Validação do RSDD*

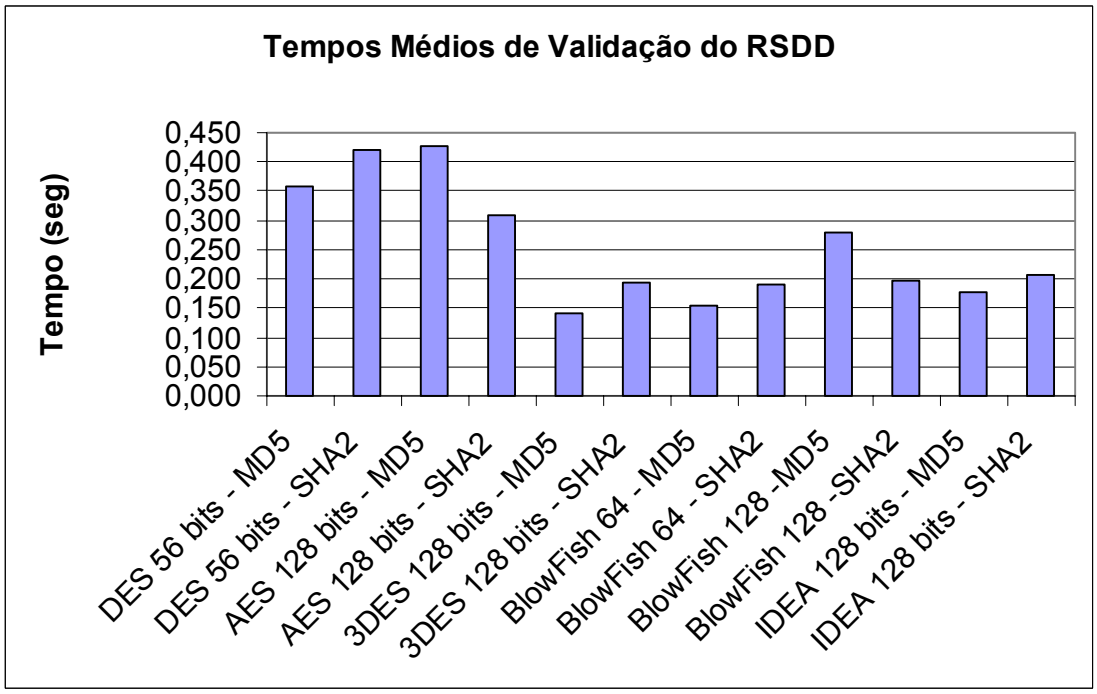


Figura 5-15: *Tempos Médios de Validação do RSDD pelo SPEP-A*

5.3.5 Analise dos Resultados

Para compararmos as varias combinações entre os algoritmos de criptografia simétrica e de assinatura, adotamos o critério de somar os tempos médios de cada um dos eventos dos algoritmos, ou seja, somamos os tempos médios da geração da chave secreta, geração do RSL, geração do RSDD, verificação do RSDD e verificação do RSDD. Como resultado obtivemos o tempo total dos eventos citados acima para cada um dos algoritmos. Estes tempos servirão para que possamos encontrar a combinação de algoritmos que gerou os elementos em um menor tempo.

Eventos	DES MD5	DES SHA2	AES MD5	AES SHA2	3DES MD5	3DES SHA2	BlowFish 64 MD5	BlowFish 64 SHA2	BlowFish 128 MD5	BlowFish 128 SHA2	IDEA MD5	IDEA SHA2
Ger. RSL	0,605	0,728	0,614	0,688	0,622	0,743	0,692	0,630	0,601	0,668	0,637	0,715
Ger. RSDD	0,683	0,663	0,577	0,773	0,689	0,753	0,586	0,752	0,728	0,723	0,481	0,511
Verif. RSL	0,242	0,378	0,264	0,406	0,429	0,389	0,337	0,400	0,492	0,259	0,390	0,330
Verif. RSDD	0,360	0,422	0,427	0,308	0,141	0,194	0,154	0,191	0,279	0,196	0,176	0,206
Chave Secr.	2,519	2,519	2,478	2,478	2,519	2,519	2,494	2,494	2,492	2,492	2,506	2,506
Tempo Médio Total	**4,408**	**4,711**	**4,360**	**4,653**	**4,401**	**4,598**	**4,263**	**4,468**	**4,593**	**4,339**	**4,190**	**4,268**

Tabela 9: *Tabela com a soma dos tempos médios de todos os eventos utilizando algoritmos de criptografia simetria e assinatura.*

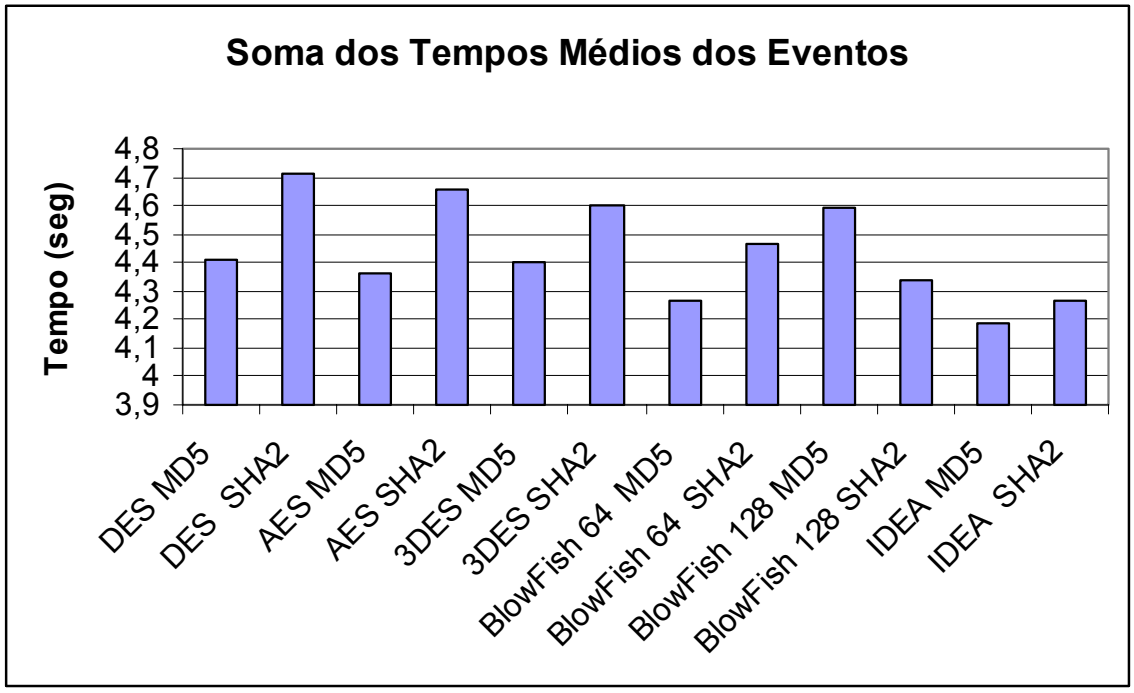


Figura 5-16**:** *Gráfico com a Soma dos Tempos Médios dos Eventos de cada um dos algoritmos*

Pela soma dos tempos médios de todos os eventos de cada um dos algoritmos, podemos observar que a combinação que gerou os elementos em menor tempo o algoritmo IDEA com chave de 128 bits e o algoritmo de assinatura MD5 com tempo total de 4,19 segundos. Seguindo o mesmo critério, quem apresentou o pior resultado foi a combinação entre os algoritmos DES com chave de 56 bits e o algoritmo de assinatura SHA-2 com tempo 4,711 segundos, um acréscimo de 12,4% em relação ao tempo do algoritmo de melhor desempenho.

Levando-se em consideração que um SPEP-A em uma rede ativa pode ser um roteador e por conta disso possui limitações de processamento, fizemos uma analise em separado dos tempos dos eventos gerados no SPDP-A e SPEP-A para verificar qual o algoritmo de criptografia possui melhor desempenho em cada um destes elementos.

A analise foi feita somando-se, por algoritmo, os eventos gerado em cada um dos elementos. No SPEP-A foram somados os tempos de verificação do RSL e do RSDD. No SPDP-A foram somados os tempos de geração do RSL e RSDD e o tempo de geração da chave secreta de comunicação.

SPDP-A												
Eventos	**IDEA MD5**	**AES MD5**	**IDEA SHA2**	**BlowFish 64 MD5**	**DES MD5**	**BlowFish 128 MD5**	**3DES MD5**	**BlowFish 64 SHA2**	**BlowFish 128 SHA2**	**DES SHA2**	**AES SHA2**	**3DES SHA2**
Ger. RSL	0,637	0,614	0,715	0,692	0,605	0,601	0,622	0,63	0,668	0,728	0,688	0,743
Ger. RSDD	0,481	0,577	0,511	0,586	0,683	0,728	0,689	0,752	0,723	0,663	0,773	0,753
Chave Sec.	2,506	2,478	2,506	2,494	2,519	2,492	2,519	2,494	2,492	2,519	2,478	2,519
Tempo Médio	**3,624**	**3,669**	**3,732**	**3,77**	**3,81**	**3,821**	**3,83**	**3,876**	**3,883**	**3,91**	**3,94**	**4,015**

Tabela 10: *Tabela com a soma dos tempos médios de todos os eventos gerados no SPDP-A*

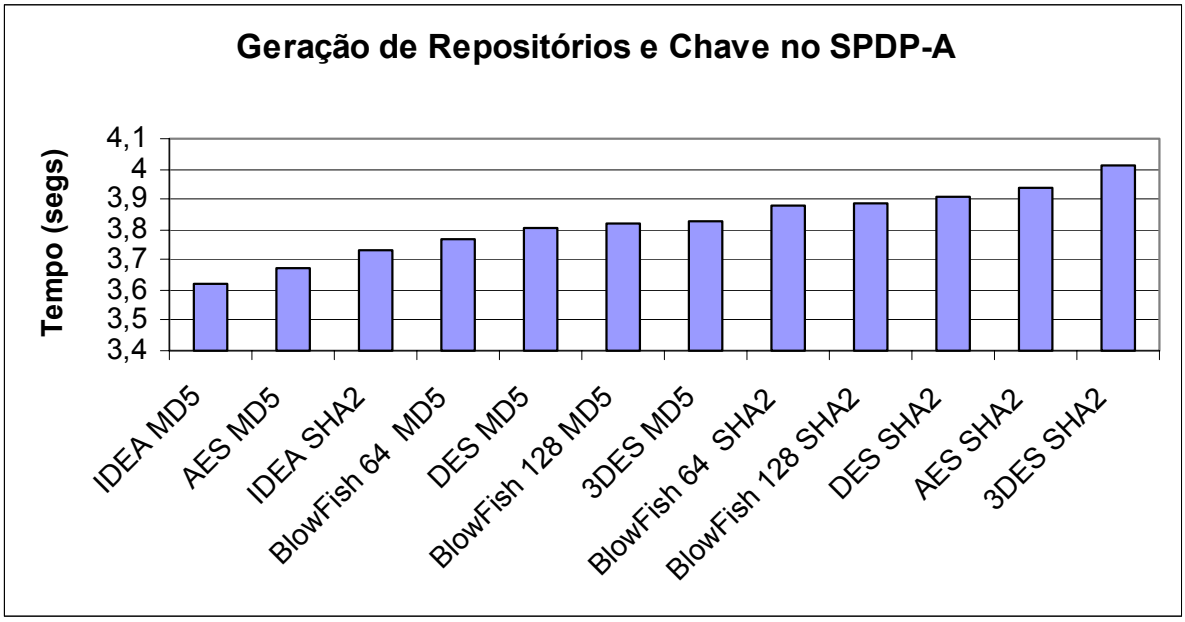


Figura 5-17: *Gráfico com os tempos total dos eventos realizados no SPDP-A, por algoritmo.*

SPEP-A												
Eventos	**BlowFish 128 SHA2**	**BlowFish 64 MD5**	**IDEA SHA2**	**IDEA MD5**	**3DES MD5**	**3DES SHA2**	**BlowFish 64 SHA2**	**DES MD5**	**AES MD5**	**AES SHA2**	**BlowFish 128 MD5**	**DES SHA2**
Verif. RSL	0,259	0,337	0,330	0,390	0,429	0,389	0,400	0,242	0,264	0,406	0,492	0,378
Verif. RSDD	0,196	0,154	0,206	0,176	0,141	0,194	0,191	0,360	0,427	0,308	0,279	0,422
Tempo Médio	**0,455**	**0,491**	**0,536**	**0,566**	**0,570**	**0,583**	**0,59**	**0,602**	**0,691**	**0,714**	**0,77**	**0,800**

Tabela 11: *Tabela com a soma dos tempos médios de todos os eventos gerados no SPEP-A*

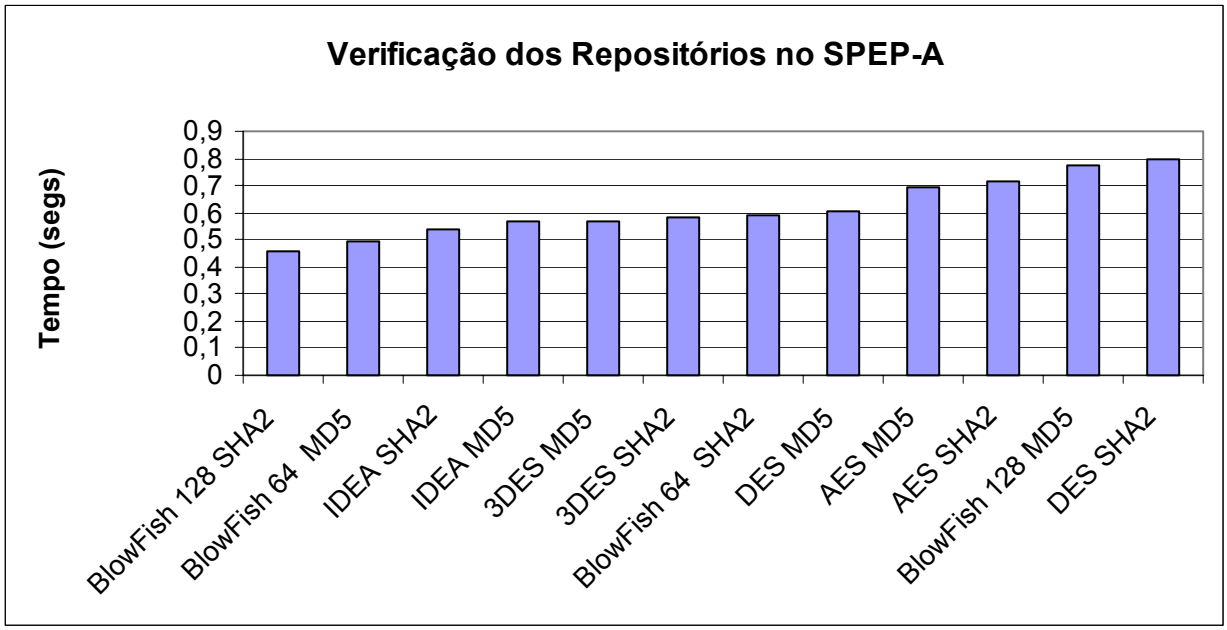


Figura 5-18: *Gráfico com os tempos total dos eventos realizados no SPEP-A, por algoritmo*

Na analise dos elementos em separado, concluímos que no SPDP-A o algoritmo com melhor desempenho foi o IDEA utilizando assinatura MD5 com um tempo total de 3,624 segundos e o de pior desempenho foi o algoritmo 3 DES utilizando assinatura SHA2, com o

tempo de 4,015 segundos, aumento de 10,8% em relação ao tempo do algoritmo com melhor desempenho. No elemento SPEP-A, o algoritmo com melhor desempenho foi o BlowFish utilizando chave de 128 bits e assinatura SHA2 com o tempo de 0,455 segundos, e o de pior desempenho foi o DES com assinatura SHA2 com o tempo de 0,800 segundos, um aumento de 75,8% em relação ao tempo do algoritmo de melhor desempenho.

5.4 Considerações Gerais

Este capítulo expôs todo o trabalho realizado para a implementação do protótipo da arquitetura RSPRA, e apresentou simulações onde se verificou toda sua eficácia.

A implementação do protótipo foi realizada em uma máquina linux com o Java 1.4.2 e o ANTS 2.0.2. instalados. E os primeiros testes foram realizados localmente, fazendo uso de um esquema que o ANTS possui de simular vários elementos de rede em uma única máquina. Porém, esta simulação local não era apropriado para o tipo de medição que optou-se fazer, por isso foi montado o ambiente de rede descrito neste capítulo. Os testes em rede só foram realizados após a validação da arquitetura no ambiente de simulação acima citado.

A criptografia híbrida foi escolhida pois se adaptou muito bem ao tipo de solução proposta. O algoritmo RSA foi escolhido para criptografia assimétrica por ser o mais conhecido do mercado e praticamente o único disponível. Sua utilização sanou as necessidades de segurança da nova arquitetura, com o custo de termos que utilizar uma chave de 2048 bits que aumentou o custo de processamento. No início dos testes simulados da arquitetura, durante a coleta de tempos, observou-se que o tempo de geração da chave RSA de 2048 bits variava muito entre uma execução e outra (até 400%). Num estudo mais aprofundado, observou-se que a semente do número aleatório (utilizado para a geração da chave RSA), quando não informada explicitamente, fazia com que o tempo de criação da chave variasse consideravelmente. A solução encontrada foi a de utilizar uma semente fixa para criação do número aleatório, desta forma o tempo de geração apresentou uma certa estabilidade.

A modificação das cápsulas foi fundamental para o implementação do protótipo do RSPRA. O tamanho das cápsulas aumentou em relação à arquitetura original, isto foi provocado pela inclusão de controles de segurança nas cápsulas. A cápsula que faz a abertura de conexão entre o SPEP-A e o SPDP-A (OPN) aumentou de 52 bytes para 756 bytes. Este aumento se deu

pelo fato de que o SPEP-A passou a enviar sua chave pública (RSA de 2048 bits) ao SPDP-A. A cápsula que faz a entrega das políticas ao SPEP-A (DEC) aumentou de 228 bytes para 2014 bytes, aumento este provocado pela inclusão do RSDD na cápsula. As trocas de algoritmos de criptografia simétrica, realizadas durante o experimento, praticamente não alteraram o tamanho das cápsulas. Isso se deu pelo fato de que o aumento do tamanho das cápsulas está vinculado ao tamanho da chave RSA, que não variou durante o experimento.

A maior preocupação durante a implementação não foi com a solução adotada dos repositórios seguros, pois esta já se mostrou eficaz quando aplicada à agentes móveis, conforme demonstrado em [WANGHAM 04]. A preocupação voltou-se para o tempo extra de processamento que seria adicionado a arquitetura com a inclusão dos controles de segurança. Foi realizada uma analise entre os diversos algoritmos de assinatura e criptografia simétrica para verificar os que apresentaram melhor desempenho. Neste experimento observou-se que a combinação do algoritmo de criptografia simétrica IDEA utilizando assinatura MD5 apresentou melhor desempenho na soma dos tempos dos eventos do SPDP-A e na soma dos tempos totais (eventos realizados no SPDP-A somados com os tempos dos eventos realizados no SPEP-A), porém esse algoritmo ficou apenas em quarto lugar na soma dos tempos dos eventos realizados no SPEP-A. O algoritmo BlowFish utilizando chave de 128 bits e assinatura SHA2 ficou em primeiro lugar em desempenho na soma dos tempos dos eventos do SPEP-A e apenas em nono lugar na soma de todos os tempos e nos tempos dos eventos do SPDP-A.

6 Considerações Finais

Este trabalho propôs a solução dos problemas de segurança da arquitetura DEPMA, que é a união de gerenciamento baseado em políticas e redes ativas. Para isso fez uso de repositórios seguros que é uma adaptação dos contêineres somente leitura e contêineres de dados direcionados aplicados à agentes móveis. O paradigma de redes ativas é bastante flexível e permite modificações dinâmicas no comportamento de rede. Por possuir esta característica, além da flexibilidade, redes ativas possuem problemas de segurança. Tais problemas são tratados por vários trabalhos em nível e de formas distintas, cito *"A Secure Method For Transferring Active Packet Using Digital Signature Schemes"* [KIM 03], "*Research and Implementation of a Scalable Secure Active Network Node*" [WANG 02], *"The Price of Safety in a Active Network"* [ALEXANDER 01], "*Research and Implementation of a Scalable Secure Active Network Node*" [WANG 02], *"A Prototype of Security for Active Network"* [YANAN 02] e *"The Research on Security Architecture for Active Networks and Security Mechanism for Active Nodes"* [WU 03]. O presente trabalho é mais um que trata de segurança de redes ativas, porém focado em garantir a confiabilidade das políticas que gerenciam a rede.

6.1 Contribuições

A principal contribuição deste trabalho foi a definição da arquitetura chamada Repositório Seguro de Políticas para Redes Ativas (RSPRA), a qual é capaz de prover gerenciamento dinâmico a uma rede ativa, estendendo novas funcionalidade aos nós gerenciáveis, em tempo de execução, garantindo a confiabilidade, confidencialidade e a segurança das políticas que trafegam nesta rede ativa. Este trabalho adaptou a solução proposta em [WANGHAM 04] para solucionar problemas de segurança do trabalho de [FERRASA 04]. A arquitetura RSPRA complementou a arquitetura DEPMA, unindo todas as vantagens desta com as garantias de segurança dos repositórios seguros, tornando-se uma arquitetura mais robusta.

A arquitetura RSPRA manteve as duas formas de tratamento da arquitetura DEPMA onde garante, de forma segura, a descentralização da tomada de decisões proposta em [FERRASA 04], e trata os nós que trabalham da forma tradicional, onde existe um elemento na

rede que sempre toma as decisões finais (forma centralizada). Além disso, a arquitetura RSPRA pode servir de base para trabalhos mais complexos ligados a segurança de redes ativas e descentralização de decisões de políticas.

6.2 Sugestões de Melhorias para Trabalhos Futuros

A principal preocupação que se deve ter com a nova arquitetura é o tempo de processamento extra que foi embutido na arquitetura RSPRA (em relação a arquitetura DEPMA) que deve ser tratado com muita atenção. Além disso, o fato de ter sido utilizada uma chave RSA de 2048 bits para criptografia assimétrica, contribuiu diretamente para o acréscimo no tempo de execução. A impossibilidade de utilizar outro tamanho de chave, se deu pelo fato de que o algoritmo RSA, utilizado através do provedor de criptografia Bouncy Castle, não consegue cifrar informações que sejam maior do que a sua chave [BOUNCY 05]. Este problema aliado ao fato de não serializar as cápsulas ativas contribuíram para o resultado obtido. Outras abordagens podem ser utilizadas onde formas de serializar as informações possam ser encontradas, o que diminuiria o tamanho da chave.

Outros algoritmos de criptografia simétrica, geração de chave secreta e algoritmos de assinatura podem ser utilizados para garantir maior segurança ou diminuir o tempo de processamento. A forma de definição dos nós confiáveis da rede pode ser desenvolvida de forma mais genérica. Além disso pode ser criada uma ferramenta que calcule a quantidade de cápsulas recusadas por violação de integridade, e que possua uma interface homem-máquina, o que poder levar a uma analise gerencial da rede por parte do administrador. A aplicabilidade da arquitetura RSPRA foi direcionada a uma área de aplicação específica, sendo assim considera-se que aspectos e requisitos específicos foram abordados. Novas formas de aplicação, em diferentes áreas podem ser exploradas a fim de aprimorar a arquitetura com base em novos aspectos e requisitos.

7 Referências

[AHN 03] AHN,G., et.al., *"Security Policy Decison for Automation of Security Network Configuration"*, IEEE Network, p.1057-1061, 2003

[ALEXANDER 01] ALEXANDER, D.S., et al. *"The Price of Safety in a Active Network"*, J.Commun., vol.3 nº1, p.4-18, Mar.2001.

[BERKELEY 05] http://dnclab.berkeley.edu/~kenneth/courses/sims250/des.html. Acessado em 20/10/2005

[BOYLE 00] BOYLE, J. et.al. *"The COPS (Common Open Policy Service) Protocol"*. RFC 2748, Janeiro 2000.

[BOUNCY 05] *"Bouncy Castle Org"*. Disponível em <www.bouncycastle.org>. Acessado em Dezembro de 2005

[CALVERT 99] CALVERT, K. L. Architectural Framework for Active Networks. Draft - AN Architecture Working Group. Disponível em http://protocols.netlab.uky.edu/~calvert/arch-docs.html>. Acessado em 17/02/2005

[COLOURIS 00] COLOURIS, G., DOLLIMORE J., KINDBERG T., *"Distributed Systems Concepts and Design"* . Addison Wesley – Third Edition. June 2000

[COUTINHO 97] COUTINHO, S. C. "*Números inteiros e criptografia RSA*", Série de Computação e Matemática, IMPA, Rio de Janeiro, 1997.

[CRIPTO 05] "*Criptografia de Dados*", http://www.delphibr.com.br/artigos/criptografia.htm. - Acessado em 20/10/2005

[FERRASA 04] FERRASA, A.. *"Uma Arquitetura Utilizando Redes Ativas Para Gerenciamento Baseado Em Políticas "*. Curitiba, 2004. 127f.. Dissertação (Mestrado em Ciências) – Centro Federal de Educação Tecnológica do Paraná – CEFET Pr.

[HOWES 97] HOWES,T., KILLE, S., WAHL, M. *"Lightweight Directory Access Protocol"*. RFC 2251, Dezembro 1997

[KARNIK 98] KARNIK,N. *"Security in Móbile Agent System"*. Tese de Doutorado, University of Minnesota.

[KIM 03] KIM, Y., NA, J. and SOHN, S., *"A Secure Method For Transferring Active Packet Using Digital Signature Schemes"*, IEEE Computer Society, p.66-69, 2003

[LIU 03] LIU, Z.; CAMPBELL, R.H.; MICKUNAS, M.D., *"Active Security Support for Active Network"*, IEEE Transactions on System, Man and Cybernetics – Part C: Apllications ans Reviews, Vol.33, Nº 4, p.432-445, November 2003.

[MATHIAS 06] MATHIAS, L.A.P. *"Algoritmo de Criptografia AES"* , http://www.gta.ufrj.br/grad/05_2/aes/ - Acessado em 11/01/2006

[MENEZES 05] MENEZES, E.S., SADOK, D.F.H., PEREIRA,P.R. *"Uma Abordagem para Implementação de Gerenciamento de Políticas em Redes de Serviços Diferenciados"* – Acessado em Fevereiro/2005

[PUTTINI 05] PUTTINI, R.S, SOUSA R.T.Jr, *"Criptografia, Autenticação e Assinatura Digital"* – http://www.redes.unb.br/security/cripto.pdf - Acessado em 10/09/2005

[RAP 05] IETF Resource Allocation Protocol Working Group. Disponível em <http://www.ietf.org/html.charters/rap-charter.html>. Acessado em Fevereiro de 2005.

[RIVEST 78] RIVEST,R.L., et al. *"A Method of Obtaining Digital Signatures and Public Key Cryptosystem"*. ACM, Vol.21, pp.120-6.

[SCHNEIER 04] SCHNEIER, B. "*A weblog covering security and security technology*", http://www.schneier.com/blog/archives/2004/10/the_legacy_of_d.html - Acessado em 23/03/2006

[SCHNEIER 06] SCHNEIER, B. "*The Blowfish Encryption Algorithm*" , http://www.schneier.com/paper-blowfish-fse.html - Acessado em 23/03/2006

[SLOMAN 98] SLOMAN, M,; "*Policy Based Management of Telecommunication Systems and Networks*", presented at First UK Programmable Networks and Telecommunications Workshop, Hewlett-Packard Laboratories, Bristol, 1998.

[SMITH 04] SMITH, J.M., NETTLES, S.M., *"Active Networking: One View of the Past, Present and Future",* IEEE Transactions on System, Man and Cybernetics – Part C: Apllications ans Reviews, Vol.34, Nº 1, p.4-18, February 2004.

[TEIXEIRA 00] TEIXEIRA, J. H.; MORAIS, L. F. M.; TEIXEIRA, S. R. "*Uma proposta para o emprego de Tecnologias de Redes Ativas no Gerenciamento de Redes",* WTMN'00: 5th Workshop Telecommunications Management Network, 2000.

[TENNENHOUSE 97] TENNENHOUSE, D.L. et al. *"A Survey of Active Network Research"*, IEEE Communications Magazine, vol.35, pp.80-86, Jan 1997.

[WANG 02] WANG,J., LI Z., KOU Y., "*Research and Implementation of a Scalable Secure Active Network Node*", IEEE Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, p.111-115, 4-5 November 2002

[WANGHAM 04] WANGHAM, M.S. et al. *"Repositório Seguro de Dados para Proteção de Agentes Móveis contra Plataformas Maliciosas"*. SBRC 2004

[WETHERALL 96] WETHERALL, D. J.; TENNENHOUSE, D. L. *"Towards an Active Network Architecture"*. ACM SIGCOMM Computer Communication Review, v. 26, n. 2, April 1996.

[WU 03] WU,Y, et al., *"The Research on Security Architecture for Active Networks and Security Mechanism for Active Nodes"* IEEE, p.58-65, 2003

[YANAN 02] YANAN, K., ZENGZHI, L. and ZHIGANG, L., *"A Prototype of Security for Active Network"*, IEEE Computer Society, Proceedings of the Fifth Internacional Conference on Algorithms and Architectures for Parallel Processing, p.1-4, 2002

[YAVATKAR 00] YAVATKAR, R. et.al. *"A Framework for Policy-based Admission Control"* . RFC 2753, Janeiro 2000.

[ZEBIANE 03] ZEBIANE, D., et al. *"Active Network and Policy Based Management",* IEEE Network, p.70-76, 2003

Anexo A Algoritmos de Criptografia

A 1. Algoritmo DES

O DES (Data Encryption Standard) foi criado em meados dos anos 70 pelo National Bureau of Standards e foi o primeiro algoritmo moderno, pública e grátis. Por muitas décadas, DES foi aplicado em ampla escala comercial. Este algoritmo toma como entrada uma chave de 64 bits, dos quais somente 56 bits são usados. Destes 56 bits, 16 sub-chaves de 48 bits são criados. A mensagem é dividida em pedaços de 64 bits, e uma complexa série de passos cifram a mensagem usando cada uma das sub-chave [SCHNEIER 04]. DES deve primeiro criar 16 sub-chaves. A string de 8 caracteres que representa a chave é girada em dois inteiros que são passados à função des_createKeys. Os bits destes dois inteiros são reorganizados de acordo com PC1 (permuted choice 1). Desta forma, o 57º bits da chave original se transforma no primeiro bit, etc.. A permutação é realizada utilizando uma seqüência da permutação, que é uma maneira inteligente de girar e de comutar bits entre 2 inteiros [SCHNEIER 04].

A 2. Algoritmo IDEA

O IDEA (International Data Encryption Algorithm) foi projetado por dois pesquisadores na Suíça. Ele utiliza uma chave de 128 bits, que o tornará imune à qualquer técnica ou máquina conhecida atualmente.A estrutura básica do algoritmo se assemelha à do DES no que diz respeito ao fato dos blocos de entrada de texto simples de 64 bits serem deturpados em uma seqüência de interações parametrizadas para produzir blocos de saída de texto cifrado com 64 bits. Devido à extensiva deturpação dos bits (em cada iteração, cada bit de saída depende de cada bit de entrada), são necessárias oito interações [CRIPTO 05].

A 3. Algoritmo AES

Numa proposta de substituir o DES, o NIST (National Institute of Standards and Technology dos E. U.) promoveu uma competição para que fosse feito um algoritmo que seria chamado AES (Advanced Encryption Standard) que atendesse as seguintes especificações:

- Algoritmo publicamente definido
- Ser uma cifra simétrica de bloco
- Projetado para que o tamanho da chave possa aumentar
- Implementável tanto em hardware quanto em software
- Disponibilizado livremente ou em acordo com termos ANSI

onde foi julgado:

- Segurança (esforço requerido para criptoanálise)
- Eficiência computacional
- Requisitos de memória
- Adequação a hardware e software
- Simplicidade

O processo seletivo teve inicio em 1997 e terminou em 2000 com a vitória do algoritmo Rijndael escrito por Vincent Rijmen e Joan Daemen. Este algoritmo criptografa e descriptografa usando chave criptografada e blocos, ambos de tamanhos de 128, 192 ou 256 bits. Pode ser implementado em um SmartCard usando pouco código e memória e como a cifra não emprega operações aritméticas, não exige muito poder de processamento. O algoritmo não baseia sua segurança em interações obscuras e não bem compreendidas entre operações aritméticas, não permitindo assim, espaço para esconder um trap-door O projeto permite a especificação de variantes com o comprimento do bloco e da chave, ambos variando de 32 em 32 bits no intervalo de 128 até 256 bits. Embora o número de rodadas seja fixo na especificação, pode ser modificado como um parâmetro caso haja problemas de segurança [MATHIAS 06].

A 4. Algoritmo 3DES

O 3DES, sigla para Triple Data Encryption Standard é um padrão de criptografia baseado no algoritmo de criptografia DES desenvolvido pela IBM em 1974 e adotado como padrão em 1977. 3DES usa 3 chaves de 64 bits (o tamanho máximo da chave é de 192 bits), os dados são cifrados com a primeira chave, decifrados com a segunda chave e finalmente cifrados novamente com a terceira chave. Isto faz com que o 3DES seja mais lento que o DES original, mas oferece maior segurança. Em vez de 3 chaves, podem ser utilizadas apenas 2, fazendo-se K1 = K3 [BERKELEY 05].

A 5. Algoritmo BlowFish

O Blowfish é um algorítmo criptográfico simétrico de blocos que pode ser usado em substituição ao DES ou IDEA. Ele toma uma chave de tamanho variável, de 32 a 448 bits. O Blowfish foi desenvolvido em 1993 por Bruce Schneier como uma alternativa grátis mais rápida para os algorítmos criptográficos existentes. Desde então ele vem sendo analisado de forma considerável e está conquistando a aceitação do mercado como um algorítmo forte. O Blowfish não é patenteado, tem sua licença grátis e está a disposição para todos.

O Blowfish possui uma chave tamanho variável, cifra 64-bit do bloco. O algoritmo consiste em duas porções: uma parte que é a chave de expansão e outra que são os dados cifrados. A expansão chave converte uma chave de 448 bits em diversas sub chaves que totalizam 4168 bytes. O cifra de dados ocorre através de 16 votas de Feistel Network. Cada volta consiste em uma permutação chave dependente, e em uma substituição de chave e dados dependentes. Todas as operações são XORs e adições em palavras 32-bit [Schneider 06].

A 6. Algoritmo RSA

O RSA consiste em um sistema de criprografia de chave pública tanto para codificação/decodificação quanto para autenticação digital, tendo sido inventado em 1977 por Ron Rivest, Adi Shamir e Leonard Adleman, todos pesquisadores do MIT. A principal vantagem da criptografia baseada em chave pública/privada (criptografia assimétrica) em relação a criptografia baseada em chave secreta (criptografia simétrica) é a segurança no mecanismo de troca de chaves. No sistema baseado em chave pública/privada as chaves privadas nunca precisam ser transmitidas. Num sistema de chave secreta, sempre existe o risco de que um intruso descubra a chave secreta durante a transmissão. Outra vantagem do sistema baseado em chave pública/privada é que ele pode fornecer um método para assinaturas digitais. Por outro lado, são muito mais lentos computacionalmente nos processos de codificação/decodificação que os algoritmos de criptografia simétrica [COUTINHO 97].

O procedimento de codificação e decodificação utilizado pelo sistema RSA funciona da seguinte forma [PUTTINI 05]:

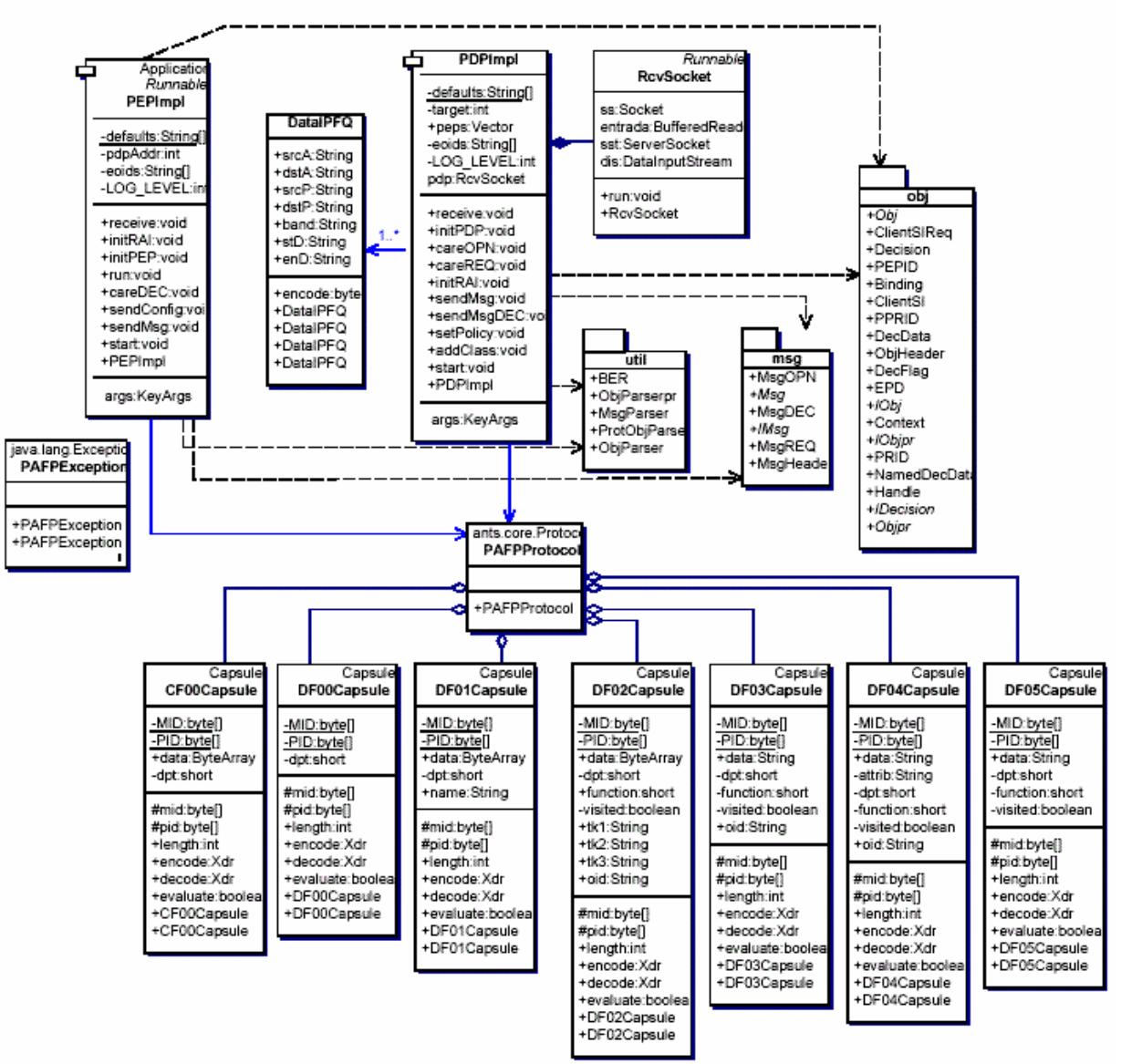
1. Estabelece-se dois números primos grandes p e q e calcula-se o módulo n definido por n=pq. Os fatores p e q devem ser mantidos em segredo ou serem destruídos.
2. Deve-se arbitrar um inteiro aleatório qualquer d que seja relativamente primo ao inteiro $\phi(n)=(p-1)(q-1)$. Feito isso, encontra-se o valor de e na faixa $1 \leq e \leq (p-1)(q-1)$, que satisfaça a: $ed \equiv 1 \bmod \phi(n)$.
3. Obtém-se, desta forma, as chaves pública e privada. Por convenção o par de números inteiros (e,n) é definido como sendo a chave pública e o par (d,n) como sendo a chave privada.

A segurança do sistema está na dificuldade de fatorar n em p e q. O algoritmo mais rápido de fatoração, leva T = exp(sqrt (ln(n) ln(ln(n)))) passos, onde ln denota o logaritmo natural e exp seu inverso. A segurança do sistema depende ainda de duas suposições críticas, a primeira é que a fatoração é necessária para quebrar o sistema, e a segunda é que a fatoração é intratável computacionalmente [PUTTINI 05].

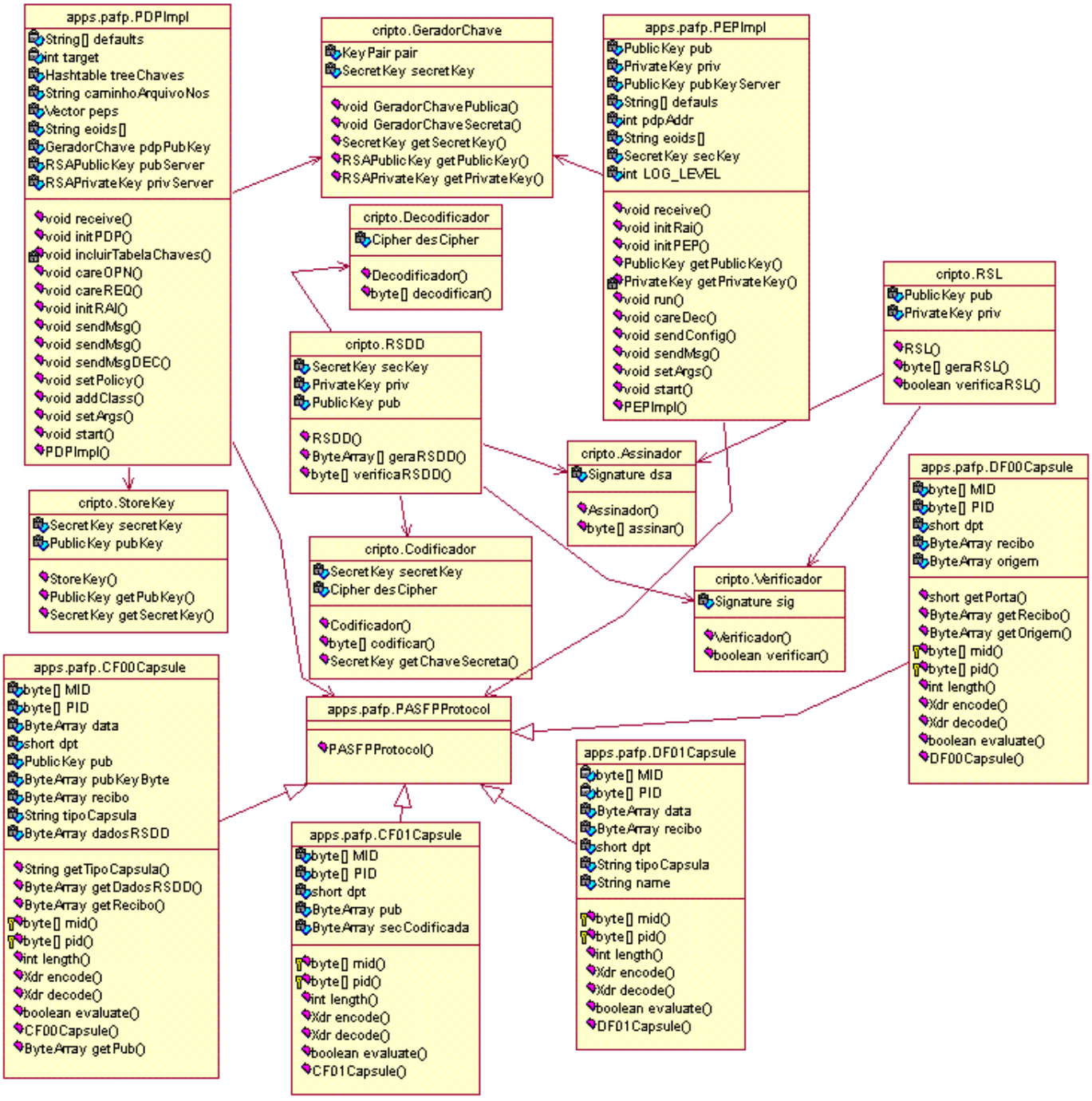
Anexo B Diagrama de Classes

O anexo B apresenta o diagrama das principais classes da arquitetura original (DEPMA) e um diagrama com as classes novas e as classes que foram modificadas para dar origem a nova arquitetura (RSPRA).

B 1. Pacote Apps.Pafp Original



B 2. Classes Modificadas da arquitetura RSPRA



Anexo C Tempos Resultantes das Execuções do Cenário

C1. Tabela com Tempos de Geração do Chave RSA de 2048 bits

Nº Execuções	Tempo (seg) de Geração da chave no SPDP-A	Tempo (seg) de Geração da chave no SPEP-A
1	7,356	3,426
2	8,529	4,568
3	7,148	4,746
4	7,748	5,565
5	8,442	4,284
6	9,201	3,664
7	8,504	3,73
8	8,867	4,782
9	9,306	3,623
10	9,113	4,453
11	9,141	4,477
12	9,43	3,66
13	11,156	4,738
14	9,463	4,827
15	9,645	3,262
16	9,805	3,221
17	9,182	3,035
18	9,122	4,692
19	9,337	3,549
20	9,407	3,657
21	7,749	3,959
22	7,521	4,021
23	8,816	4,281
24	9,405	4,432
25	10,745	5,089
26	10,732	3,641
27	10,872	3,59
28	10,448	5,899
29	8,717	4,566
30	9,091	4,824
31	8,13	4,06
32	8,346	3,147
33	8,447	4,822
34	8,7	4,169
35	7,733	4,651
36	8,514	3,542
37	8,997	4,116
38	7,695	4,267
39	8,993	3,86
40	7,443	4,923
41	8,611	5,355
42	9,935	4,132
43	8,972	4,701
44	8,844	5,399

45	7,845	4,568
46	7,751	5,416
47	7,962	5,746
48	8,92	4,138
49	8,339	3,063
50	7,777	3,039
51	7,878	4,3
52	7,122	3,836
53	7,995	3,133
54	8,2	4,695
55	6,435	4,409
56	9,437	4,438
57	8,107	4,75
58	7,577	4,254
59	9,427	4,364
60	8,922	4,67
61	9,279	3,503
62	7,41	3,977
63	8,425	3,611
64	10,501	4
65	10,937	4,001
66	8,102	5,155
67	11,523	4,042
68	7,04	4,607
69	8,586	3,231
70	9,102	3,16
71	10,306	4,907
72	11,991	4,916
73	9,476	3,92
74	9,827	3,948
75	9,817	4,929
76	9,876	3,919
77	9,764	4,917
78	10,722	5,086
79	10,269	4,936
80	10,13	4,937
81	9,624	5,352
82	8,293	3,799
83	8,407	4,093
84	9,709	4,303
85	7,122	3,303
86	7,386	4,277
87	9,21	5,008
88	10,826	4,108
89	10,181	4,281
90	8,193	5,158
91	8,984	3,45
92	8,069	4,435
93	9,489	4,929
94	9,479	4,919
95	9,381	4,966
96	8,721	4,009
97	9,71	3,932
98	9,276	3,938

99	9,658	3,921
100	10,21	4,92
101	8,266	4,118
102	8,687	3,674
103	8,069	3,68
104	9,684	4,484
105	8,525	4,503
106	8,842	4,092
107	9,066	5,595
108	11,63	3,402
109	11,567	4,396
110	9,179	4,442
111	9,687	4,592
112	9,56	3,97
113	11,083	3,952
114	10,123	3,964
115	10,961	3,978
116	9,654	3,954
117	10,362	3,95
118	9,621	4,11
119	9,865	3,945
120	11,201	4,149
	Tempos Médios (seg)	
	9,114125	**4,266475**

Resumo

A tecnologia de redes ativas aplicada ao gerenciamento de rede baseado em política provê uma excelente ferramenta de gerenciamento, pois toda flexibilidade de redes ativas permite modificações dinâmicas no comportamento da rede. Porém, a flexibilidade de redes ativas também introduz problemas de segurança. Motivado por estas questões de segurança, o presente trabalho apresenta uma arquitetura que provê segurança ao gerenciamento de redes ativas baseado em políticas, através da utilização de repositórios seguros utilizados em agentes móveis. Um protótipo da arquitetura foi implementado e um estudo sobre o impacto de processamento extra foi realizado.

PALAVRAS-CHAVE

Redes Ativas, Gerenciamento Baseado em Políticas, Repositório Seguro, Criptografia.

ÁREA/SUB-ÁREA DE CONHECIMENTO

Teleinformática, Telecomunicações

2006

N°: 407

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo