UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Rick Lopes de Souza

Modificações no Provedor de Serviços Criptográficos OpenHSM para atender Ambientes de Alta Demanda

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação.

Jeandré Monteiro Sutil
Orientador

Prof. Ricardo Felipe Custódio, Dr.
Co-Orientador

Florianópolis, junho de 2011

Modificações no Provedor de Serviços Criptográficos OpenHSM para atender Ambientes de Alta Demanda

Rick Lopes de Souza

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciências da Computação e aprovada em sua forma final pelo Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

Prof. Vitório Bruno Mazzola, Dr.
Coordenador do Curso

Banca Examinadora

Jeandré Monteiro Sutil

Prof. Ricardo Felipe Custódio, Dr.

Roberto Gallo, M.Sc.

Marcelo Carlomagno Carlos, M.Sc.

Prof. Lau Cheuk Lung, Dr.

"Se os fatos não se encaixam na teoria, modifique os fatos."

Albert Einstein

Ofereço este trabalho aos meus pais que me proporcionaram as oportunidades e me apoiaram até o fim dessa jornada.

Agradecimentos

Gostaria de agradecer aos meus amigos, colegas e todas as pessoas ligadas a mim em todo esse percurso de curso. Sem vocês nada disso seria possível e seria sem graça.

Um agradecimento especial a todos do LabSEC que estiveram do meu lado em todo meu aprendizado e grande parte da minha vida acadêmica. Ao professor Ricardo Custódio e ao meu Orientador Jeandré um agradecimento mais que especial por terem me proporcionado a oportunidade de aprender muito sobre segurança digital e a ser o que sou hoje.

Sumário

Lista de Figuras

Lista de Siglas

AC	Autoridade Certificadora
ASI-HSM	Advanced Security Initiative - Hardware Security Module
ICP	Infra-estrutura de Chaves Públicas
ICP-Brasil	Infra-estrutura de Chaves Públicas Brasileira
ICPEDU	Infra-estrutura de Chaves Públicas para Ensino e Pesquisa
ITI	Instituto de Tecnologia da Informação
LabSEC	Laboratório de Segurança em Computação - UFSC
LEA	Laboratório de Ensaios e Análise
MCT	Manual de Condutas Técnicas
MSC	Módulo de Segurança Criptográfica
NIST	National Institute of Standards and Technology
RNP	Rede Nacional de Ensino e Pesquisa
RSA	Rivest-Shamir-Adleman

Resumo

O ASI-HSM é um Módulo de Segurança Criptográfico desenvolvido em território nacional com parceria de três entidades: Rede Nacional de Ensino e Pesquisa (RNP), Laboratório de Segurança em Computação (LabSEC - UFSC) e Kryptus. Como o projeto foi inicialmente pensado para um ambiente de baixa demanda, os requisitos estavam voltados ao rígido controle do ciclo de vida das chaves e da sua utilização. Este trabalho propõe mudanças no desempenho com vista a adequar o provedor de serviços criptográficos OpenHSM, componente do ASI-HSM, para uso em ambiente Online, levando em consideração novos requisitos, que o adequem para uso em ambientes de alta demanda.

Com a implementação de uma ferramenta de testes foi possível obter valores mais precisos da quantidade de assinaturas por segundo que o ASI-HSM estava fazendo, e com isso, sendo parte fundamental no processo de melhoria da velocidade das assinaturas. Foram encontradas novas ferramentas para testar o funcionamento do software, onde os resultados eram fornecidos com interfaces gráficas, ajudando na interpretação dos resultados dos testes.

Após as mudanças feitas, o ASI-HSM teve um ganho de 500% na velocidade das assinaturas apenas alterando parte da implementação atual do provedor de serviços criptográficos OpenHSM, otimizando protocolos de comunicação e retirando alguns registros de operações. Os registros das operações agora são cíclicos, evitando possíveis perdas de registros ou erros ao alcançar os limites físicos que o ASI-HSM possui. Outro ganho foi em relação a segurança na utilização das chaves, pois a atrelação de um grupo de operadores ao uso possibilitou mais segurança no

processo de assinatura. Um dos principais ganhos é o fato do ASI-HSM agora ser customizável, ou seja, pode ser utilizado em diferentes tipos de demanda, não sendo mais necessário a troca do hardware.

Palavras chave:Módulo de Segurança Criptográfica, Ambientes de Alta Demanda, Otimização, Segurança

Capítulo 1

Introdução

1.1 Contextualização

O ASI-HSM é um Módulo de Segurança Criptográfico (MSC) desenvolvido com tecnologia nacional com a parceria entre a empresa Kryptus, fabricante de hardwares criptográficos, e o Laboratório de Segurança em Computação (Universidade Federal de Santa Catarina). Ambos foram responsáveis pelo projeto da arquitetura de todo o ASI-HSM e do software do projeto. O OpenHSM é o software que gerencia o ciclo de vida das chaves criptográficas no ASI-HSM e foi o alvo de estudo neste trabalho.

O projeto do qual o ASI-HSM faz parte foi lançado no ano de 2007 com o caráter experimental, cuja proposta era de implantar uma Infra Estrutura de Chaves Públicas em instituições de ensino superior [JUN 09]. Foi financiado pela Rede Nacional de Ensino e Pesquisa, fazendo parte do projeto ICPEDU.

Esse MSC foi lançado com o intuito de atender baixas demandas, como os MSCs de Autoridades Certificadoras em ambiente Offline. Não havendo grande preocupação com as otimizações relacionadas à taxa de assinaturas por segundo, bem como a outros requisitos necessários a ambientes de alta demanda.

Atualmente o ASI-HSM é utilizado em algumas instituições de ensino superior, criando certificados e chaves criptográficas para auxiliar transações em aplicações

acadêmicas e de pesquisa sendo também utilizado na Infraestrutura de Chaves Públicas Brasileira, a ICP-Brasil, que é um conjunto de práticas, ténicas e procedimentos elaborados para trabalhar com sistemas criptográficos que tem base em certificados digitais. Desde agosto de 2001, o comitê gestor rege as regras, diretivas e normas para licenciamento das Autoridades Certificadoras e Autoridades de Registro. [Ins 09]

O ASI-HSM passou nos últimos anos por um processo de homologação e foi aprovado pelo processo 00100.000241/2008-96, no âmbito da ICP-Brasil. O modelo AHX2-L3 foi avaliado pelo Laboratório de Ensaios e Auditoria - LEA, com relação aos requisitos técnicos de segurança e interoperabilidade exigidos pelo Manual de Condutas Técnicas nº 7, volume 1, e foi homologado no nível de segurança de homologação 3. [Mau 11] [Inf 09]

Atendendo aos requisitos de um ambiente de alta demanda, os usuários teriam muito ganho, pois não seria necessário adquirir MSCs de outras empresas estrangeiras, já que o ASI-HSM é um módulo desenvolvido totalmente em território nacional, com protocolo aberto e também seria uma questão de soberania nacional.

1.2 Objetivo

O objetivo deste trabalho de conclusão de curso é demonstrar o estudo, pesquisa e modificações feitas em um Módulo de Segurança Criptográfico para atender à altas demandas.

1.3 Objetivos Específicos

O objetivo específico deste trabalho é demonstrar as melhorias efetuadas em um MSC, um produto desenvolvido em território brasileiro, possibilitando a sua utilização em ambientes de alta demanda. Os objetivos são:

- Analisar o desempenho do ASI-HSM
- Identificar possíveis gargalos nos serviços providos pelo MSC

- Identificar possíveis vulnerabilidades advindas da mudança de ambiente (baixa demanda para alta demanda)
- Propostas de modificações com vista a solucionar as questões identificadas
- Análise do impacto da implementação de tais modificações nos serviços do módulo.

1.4 Justificativa

A utilização dos Certificados Digitais trouxe inúmeros benefícios para as grandes organizações, instituições de ensino, bancos entre outras finalidades. Com a inserção da tecnologia da Certificação Digital nos mais variados meios, a demanda por assinaturas digitais teve um aumento considerável. Sendo necessário a adaptação dos equipamentos relacionados a Certificação Digital para essas novas situações, em que a exigência a respeito de velocidade e segurança envolvida nos processos eram superiores a anterior.

Uma das principais finalidades para a adaptação do ASI-HSM foi o aprimorando desde a velocidade das assinaturas até a melhora da segurança na utilização das chaves criptográficas para atender a ambientes de alta demanda, como notas fiscais eletrônicas, bancos, processos jurídicos e Autoridades Certificadoras Online.

1.5 Metodologia

O processo para realizar esse trabalho foi baseado em ferramentas, estudos, conhecimento prévio e testes. O código do ASI-HSM estava disponível, facilitando a implementação do processo.

Para otimizar a realização das assinaturas, foram utilizados os seguintes passos:

- Criação de uma ferramenta para testar a velocidade das assinaturas

- Análise de desempenho com ferramentas já existentes
- Análise baseada na ferramente de testes desenvolvida
- Análise dos pontos críticos relacionados aos métodos de assinatura encontrados no OpenHSMd
- Aplicações das alterações necessárias
- Avaliação do novo código
- Avaliação dos resultados

1.6 Limitações do Trabalho

O LabSEC está envolvido diretamente com a parte do funcionamento do provedor de serviços criptográficos que roda dentro do hardware do ASI-HSM e em consequência disso, os estudos foram feitos apenas em nível lógico deste MSC. Otimizações de hardware estão fora do escopo deste trabalho.

Capítulo 2

Fundamentos Criptográficos

Este capítulo tem como principal objetivo fornecer base teórica sobre uma Infraestrutura de Chaves Públicas (ICP), ressaltando alguns problemas relacionados ao uso em ambientes Offline e Online. Serão abordados os requisitos e soluções para determinadas Autoridades Certificadoras e as principais características de um MSC.

No processo de certificação digital, um algoritmo de chaves públicas é utilizado para garantir a autenticidade e confidencialidade, mas apenas isso não é suficiente para garantir sua segurança das informações e transações. Apesar da segurança envolvida na utilização da criptografia de chaves públicas, esta está sujeita ao ataque do homem no meio, pois não há como garantir a identidade do detentor da chave em situações onde a chave pública não foi assinada, a menos que os dois se apresentem pessoalmente. Sabe-se que o dono daquela chave pública está recebendo a mensagem ou documento cifrado, entretanto não há garantias de que o receptor seja a pessoa ao qual se destina a mensagem. Para isso precisa-se adotar a política de confiança em uma terceira parte, confiando em uma entidade maior e estabelecer uma cadeia de confiança chamada de ICP. [VIE 02]

O principal mecanismo de confiança nas ICPs se baseia nas chaves criptográficas e nos certificados. Os certificados servem para identificar pessoas ou entidades e as chaves criptográficas para assinar e verificar. [VIE 02]

2.1 Criptografia de Chaves Públicas

Em 1976 Diffie e Hellman propuseram um novo sistema de criptografia revolucionário que resolveria os problemas da criptografia simétrica. Também conhecido como Criptografia Assimétrica, esse novo sistema resolvia o grande problema de distribuir as chaves de forma segura. [MAR 05]

Neste sistema proposto, cada utilizador dos softwares criptográficos, possuiria um par de chaves, uma de conhecimento público, conhecida como chave pública e outra que deveria ser guardada em segredo, conhecida como chave privada. Uma das principais propriedades é a de não ser possível obter a chave privada através da chave pública.

O novo mecanismo de estabelecimento de chaves foi concebido para que as mesmas funcionem de forma complementar. Quando uma for utilizada para cifrar, a outra será utilizada para decifrar. Por exemplo, se a chave pública for utilizada para cifrar, apenas a sua respectiva chave privada poderá ser utilizada para decifrar.

Com esse novo algoritmo, não há necessidade de um canal seguro para a troca de chaves. Além disso, diminuiu o número de chaves necessárias que cada usuário deveria ter para trocar mensagens. Com esse novo mecanismo também há a possibilidade de dois usuários que não se conhecem trocar mensagens de maneira segura.

Para a utilização deste algoritmo, deve-se primeiramente gerar um par de chaves criptográficas que poderão ser armazenadas tanto em software quanto em hardware.

Um exemplo de algoritmo utilizado na criptografia de Chaves Públicas é o desenvolvido por Rivest, Shamir e Adelman, mais conhecido por RSA, no ano de 1978, no Massachusetts Institute of Technology, do qual pode ser utilizado tanto para sigilo quanto para autenticidade. [RIV 78], mas existem outros algoritmos assimétricos que podem ser utilizados para ocasiões específicas. Os conceitos de criptografia de chaves públicas com o objetivo criptográfico podem ser exemplificados da seguinte forma:

Sigilo: Um exemplo do uso do algoritmo de chaves públicas para sigilo seria a situação

em que João e Maria desejam trocar uma mensagem secreta. Então João utilizaria a chave pública de Maria para cifrar a mensagem e a enviaria. Maria sabendo que a mensagem foi cifrada com sua chave pública, pode decifrar com sua chave privada e ter certeza de que apenas ela poderá ler essa mensagem. [MAR 10]

Autenticidade: Para provar a autenticidade de uma mensagem que foi enviada, João tenta verificar a mensagem com a chave pública de Maria. Caso tenha sucesso, fica provado que foi Maria quem cifrou a mensagem. [MAR 10]

Não-Repúdio: O não-repúdio é uma propriedade parecida com a da autenticidade. Se conseguirem verificar uma mensagem com a chave pública de João, o mesmo não poderá negar a autoria da mensagem, já que apenas ele tem a posse de sua chave privada, que foi utilizada para cifrar a mensagem. [MAR 10]

2.1.1 Resumo Criptográfico

Normalmente a cifragem de um documento utilizando a criptografia assimétrica é mais lenta do que utilizando a criptografia simétrica[Mic], sendo a função resumo um auxílio nesse processo. A função resumo, ou hash como é normalmente conhecida, é uma função que quando aplicada à um documento, retorna uma sequência de bits de tamanho fixo. Essa função tem como propriedade não ser inversível, ou seja, não deve existir uma maneira que através do resumo criptográfico, consiga chegar ao documento original. [MAR 05]

A função de resumo criptográfico é utilizada para ajudar na garantia da integridade, já que essa função foi feita de maneira que qualquer alteração realizada no documento, o resumo criptográfico resultante deve ser diferente do obtido anteriormente. [MAR 10]

Dentre as funções de resumo criptográfico, destacam-se o Secure Hash Algorithm (SHA) e o MD5. O orgão National Institute for Standards and Technology (NIST), responsável pelas padronizações na área de Tecnologia nos Estados Unidos, publicou um documento para a troca de alguns algoritmos criptográficos, sendo

um deles a troca do algoritmo SHA-1 para a família SHA-2 a partir do ano de 2011. [BAR 10]

2.1.2 Assinatura Digital

Rivest, Shamir e Adelman ,no ano de 1978, propuseram o algoritmo RSA[RIV 78], um algoritmo de chaves públicas, em que era possível garantir três propriedades: Sigilo, Autenticidade e Não-Repúdio, conforme descrito na seção 2.1. A segurança desse algoritmo tem base na dificuldade em fatorar números primos muito grandes. [MAR 05]

Como descrito na seção 2.1, o algoritmo de chaves públicas pode garantir a autenticidade ao decifrar a mensagem de uma segunda pessoa com sua chave pública.

Para funcionar de maneira eficiente e mais segura, as funções de resumo criptográfico, citadas na seção 2.1.1, são de extrema importância para prover a integridade e detectar facilmente qualquer alteração efetuada. A figura 2.1 ilustra o processo de assinatura digital.

O processo de verificação de uma assinatura digital é análogo ao processo de Autenticidade, citado na seção: 2.1. Ao assinar uma mensagem com uma chave privada, o destinatário poderá conferir, ou seja, verificar a assinatura utilizando a chave pública de quem enviou a mensagem. A figura 2.2 ilustra o processo de verificação de uma assinatura digital.

2.2 Módulo de Segurança Criptográfico

Para garantir a segurança física e restrita do uso das chaves criptográficas utilizadas nas operações envolvendo certificados, é necessário o uso de um MSC. Esse dispositivo é um hardware construído e projetado para gerenciar o ciclo de vida das chaves criptográficas e proteger do uso indevido das chaves. Esse hardware é projetado para que as funções críticas de segurança, aconteçam dentro de um perímetro criptográfico, do qual uma vez violado, provoca a destruição das mesmas. Uma vez

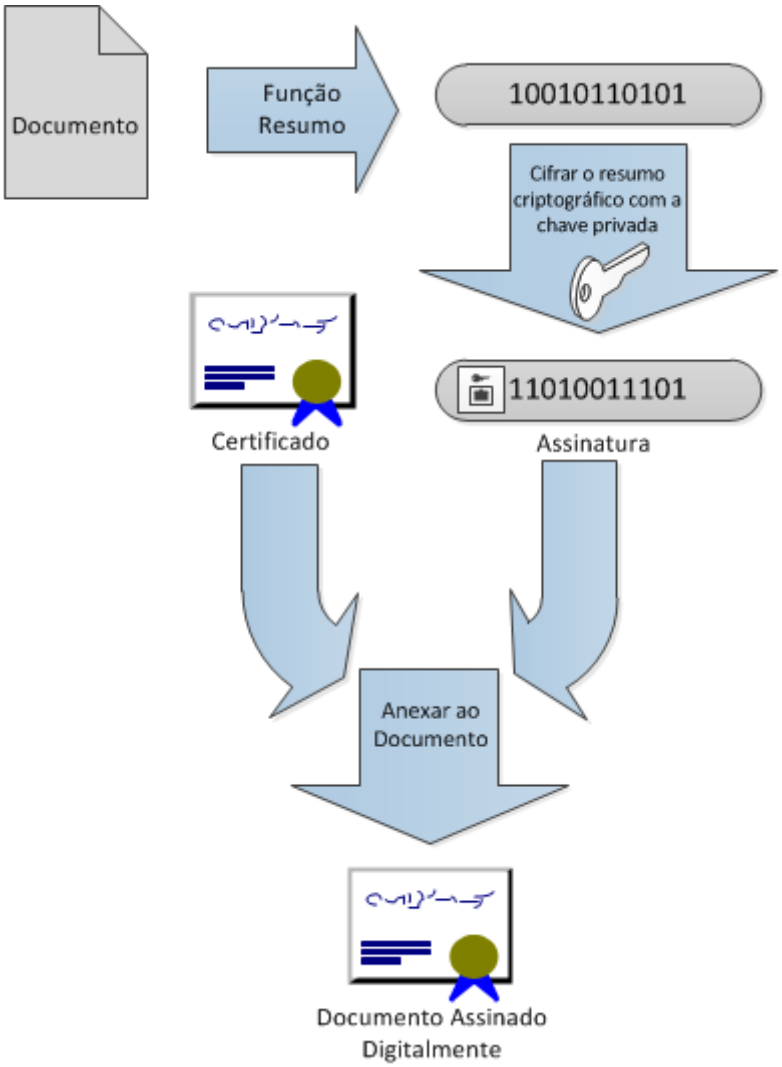

Figura 2.1: Processo Simplificado de Assinatura Digital

que é necessário a utilização das chaves para fazer assinaturas, que é uma das funções exercidas por um MSC, o resumo criptográfico é enviado para o MSC e o mesmo assina, devolvendo os dados assinados para o solicitante. Com isso é possível controlar o uso restrito e evitar que as chaves saiam do perímetro de segurança e tenham contato direto com o exterior, possibilitando o uso indevido. [MAR 05]

A comunicação com os MSCs se dão através de interfaces bem definidas como o PKCS#11 e ENGINE OpenSSL. Hoje existem MSCs projetados especialmente para o uso Online, utilizando Web Services como interface de comunicação, facili-

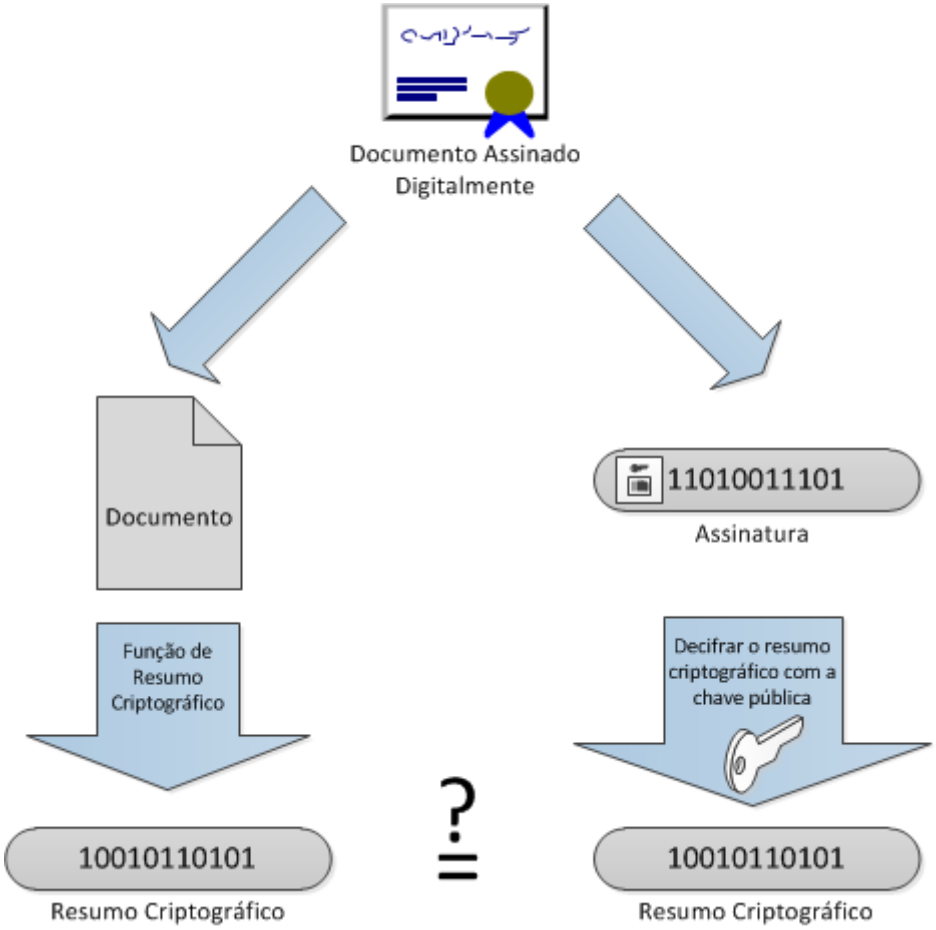

Figura 2.2: Processo Simplificado de Verificação de uma Assinatura Digital

tando a implementação e agilizando a integração com sistemas consumidores de seus serviços. Mas existem os MSCs que foram desenvolvidos na utilização Offline, como é o caso do MSC utilizado nas AC-raizes.

As principais funcionalidades de um MSC são: geração segura das chaves, armazenamento seguro, uso de criptografia e dados críticos e acelerar os servidores, evitando que tenham de prover serviços criptográficos. MSCs conseguem providenciar tanto a proteção física quanto lógica dos dados contra pessoas não autorizadas ou possíveis invasores. [SUT 11]

Dentre as principais funções de um Módulo de Segurança Criptográfico, destacam-se:

- Gerência do ciclo de chaves criptográficas
- Proteção das chaves por mecanismos de hardware e software
- Armazenamento de chaves e demais segredos.
- Assinar e Verificar

Um Módulo de Segurança Criptográfico pode ser projetado para ser um hardware com dispositivos de segurança, sensores, leitores de smartcard entre outras características. Alguns também podem possuir aceleradores criptográficos e com isso conseguem aumentar consideravelmente a quantidade de assinaturas por segundo, assim conseguindo atender a altas demandas.

2.3 Infraestrutura de Chaves Públicas

2.3.1 Certificado Digital

No centro das Infraestruturas de Chaves Públicas estão os certificados, que em suma, ligam uma chave pública com um nome distinto da pessoa ou entidade que a possui. Um certificado pode ser comparado a um passaporte, o qual tem anexo a foto da pessoa deixando o documento mais autêntico. Um passaporte é emitido por uma terceira parte confiável (governo) e contém informações a respeito da pessoa para o qual o passaporte foi emitido e também informações sobre o governo que emitiu esse documento. Um certificado digital age de forma parecida, sendo emitido por uma terceira parte confiável que contém informações sobre quem emitiu e da pessoa que possui o certificado. [HOU 01]

Assim como as marcas d'água em passaportes e outros documentos, o certificado digital contém mecanismos para garantir sua autenticidade, ser verificado e também facilitando na hora de verificar se foi forjado ou não. Assim, como uma carteira de identidade e outros documentos, o certificado digital também tem um período

de validade definido e após esse período, a validade do certificado expira e um novo certificado deve ser emitido, fazendo com que o antigo não seja confiado.

Um certificado é assinado usando a chave privada de entidades conhecidas como Autoridades Certificadoras e que serão explicadas na seção 2.3.3. Também contém informações a respeito do emissor do certificado, o período de validade entre outros dados. [SUT 11]

Assinando um certificado com a chave privada do emissor, qualquer um pode verificar essa assinatura utilizando a chave pública que está contida no certificado. Essa assinatura serve como uma proteção contra possíveis adulterações. Assinando o certificado de uma pessoa, o emissor confirma ter verificado a autenticidade da chave pública e garante ser confiável. Enquanto o emissor for confiável, os certificados assinados por ele também serão. [HOU 01]

2.3.2 Lista de Certificados Revogados

Quando existe um comprometimento da chave privada de um certificado ou existe a perda de alguma credencial, o certificado deve perder validade devendo ser revogado. Para que isso seja controlado e divulgado de maneira mais criteriosa, existem as Listas de Certificados Revogados. Nelas estão contidas os detalhes sobre os certificados revogados, como serial e data de revogação. Essas LCR são emitidas periodicamente por entidades chamadas Autoridades Certificadoras que serão melhor detalhadas na seção 2.3.3 para que todas as outras entidades possam ter acesso e fazer suas verificações e controles se determinados certificados ainda são válidos.

Mesmo que um certificado esteja na LCR, não necessariamente ele perde toda a validade e de todos os documentos que foram assinados por ele. Com essa lista é possível fazer uma verificação se uma assinatura foi efetuada antes de sua data de revogação e com isso, torna-se uma assinatura válida. Caso a assinatura tenha sido efetuada após a data da revogação, a assinatura não será válida. Todos esses dados contidos na LCR devem estar organizados para que as assinaturas possam ou não ser validadas. [MAR 05]

2.3.3 Autoridade Certificadora

Uma autoridade certificadora (AC) é uma organização que emite certificados tendo uma grande responsabilidade para garantir que os certificados que ela emite são legítimos. Assim a AC necessita garantir de todas as maneiras possíveis que cada certificado emitido contém uma chave pública que foi gerada para uma pessoa específica. Deve ter provas suficientes de todos os certificados emitidos sobre demanda, para garantir a confiabilidade da Autoridade Certificadora. [HOU 01]

A AC deve ser confiável e para tal, o seu certificado contendo a chave pública deve ser amplamente difundido. Na maioria das vezes seus certificados são publicados em determinados locais em que qualquer entidade pode ter acesso. Mais comumente os softwares que os utilizam já vem com os certificados embutidos, como os sistemas operacionais, navegadores, entre outras aplicações.

Uma AC, além de emitir certificados, deve supervisionar o ciclo de vida de cada certificado emitido. Caso ocorra algum tipo de problema, comprometimento da chave privada ou necessidade de mudança de algum atributo do certificado, a AC deverá revogar o certificado antigo. Para que os usuários verifiquem a validade de certificados, existem LCRs que são emitidas periodicamente contendo informações sobre as revogações. [SUT 11]

Em uma ICP há diferentes necessidades para o uso das chaves criptográficas das Autoridades Certificadoras, diferentes níveis de segurança e diferentes necessidades em relação ao uso das chaves criptográficas. [Mar 07]

As ICPs possuem diversos tipos de arquiteturas, uma delas e a mais utilizada é a Estrutura Hierárquica, detalhada na seção 2.3.4. Essa estrutura está baseada em uma rede de confiança complexa, nela podemos dividir em dois tipos de ambientes: Offline e Online.

2.3.3.1 Ambiente Offline

Na estrutura hierárquica o usuário confia em um único ponto, que é a Autoridade Certificadora Raiz [MAR 05], que tem seu certificado auto-assinado. Como essa

Autoridade Certificadora é a raiz da confiança, é necessário um alto grau de proteção. Não importando a quantidade de níveis de hierarquia, a AC-Raiz deve possuir acesso restrito a usuários autorizados e para manter toda essa segurança, adota-se a postura de não deixá-la conectada na internet, ou seja, uma AC-Offline. Também existem mais proteções como deixar fisicamente separada em salas com mais níveis de segurança. [Mar 07]

Devido a importância de uma Autoridade Certificadora Raiz, seu uso se restringe normalmente a assinar o certificado de Autoridades Certificadoras Intermediárias e periodicamente emitir a Lista de Certificados Revogados. Uma possível aceleração criptográfica é desnecessária, pois o número de usos é pequeno e quando utilizado, o tempo não é importante.

2.3.3.2 Ambiente Online

Existem as Autoridades Certificadoras finais, que emitem os certificados dos usuários finais, tendo uma maior demanda do que as ACs Offline. Com isso, a preocupação com a velocidade e disponibilidade na utilização das chaves criptográficas é maior. Mesmo assim, essas ACs devem ser protegidas por MSCs e em ambientes seguros.

Mantendo a estrutura hierárquica, quanto mais níveis existem na rede de confiança, menos requisitos de segurança estão envolvidos nos níveis mais baixo e o número de acessos aumentam. Com isso, a preocupação é mais voltada para a demanda, velocidade e maneiras de proteger a comunicação com tantos acessos. E para que esses requisitos sejam atendidos, os MSCs precisam estar preparado para atender tais procedimentos. [Mar 07]

Hoje os MSCs de mercado utilizam principalmente aceleradores criptográficos para aumentar o desempenho nas assinaturas e funções criptográficas. Mas quando não possuem, uma alternativa é fazer uma melhor avaliação a respeito do código, métodos para avaliar possíveis gargalos e análise dos protocolos utilizados.

2.3.4 Arquiteturas de ICPs

2.3.4.1 Arquitetura com Autoridade Certificadora Única

Esta arquitetura apresenta uma estrutura simples, onde existe apenas uma única Autoridade Certificadora que é responsável pela emissão dos certificados de todos os usuários da arquitetura, detalhado na seção 2.3.3.

Ao utilizar essa arquitetura, é necessário existir uma confiança na Autoridade Certificadora. Para validar um certificado digital, é necessário conhecer, ter a posse do certificado e confiar na Autoridade Certificadora.

A figura 2.3 ilustra essa arquitetura.

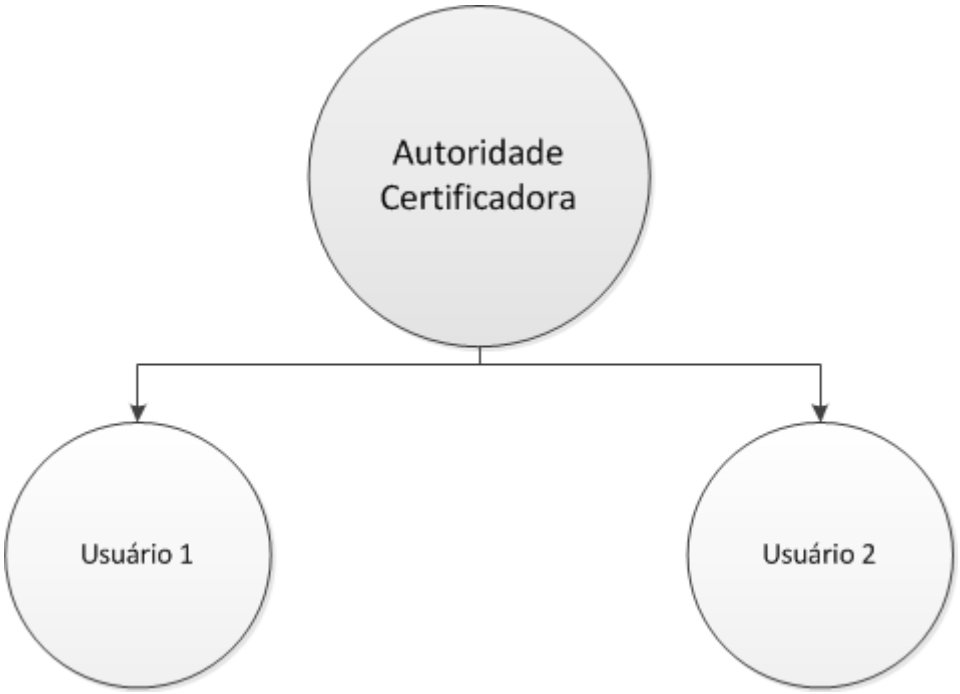

Figura 2.3: Arquitetura com Autoridade Certificadora Única

Um exemplo sobre como confiar no certificado de outra pessoa utilizando essa arquitetura seria da seguinte maneira: Caso Pedro deseje validar o certificado digital do Bruno, então terá de confiar no certificado da Autoridade Certificadora. Com isso, Pedro poderá verificar se o certificado de Bruno foi assinado pela AC confiável.

A implementação dessa estrutura é bem simples, mas existe uma vulnerabilidade grande em relação a chave utilizada da AC. Caso aconteça algum comprometimento, toda a cadeia perderá a validade, sendo necessário avisar a todos os membros que possuem um certificado emitido por essa AC, recriar novas chaves criptográficas e então emitir novos certificados. [RFC 08]

2.3.4.2 Arquitetura Lista de Confiança

Com a grande utilização dos certificados digitais, é comum que passem a existir outras ACs em diferentes setores e caso uma pessoa deseje confiar em diferentes ACs, precisará montar uma lista de confiança. Para o usuário validar um determinado certificado dessa arquitetura, basta conferir o caminho de certificação mais longo até chegar no ponto mais alto de confiança. Nesse processo, deverá passar por todas as ACs intermediárias que existem entre o certificado a ser validade e a AC Raiz.

Esta arquitetura é bastante comum e muito utilizada em organizações, já que representa uma forma real de confiança dentro de uma organização. Essa arquitetura possui um grau de complexidade alto em relação a validação de certificados, pois há a necessidade de validar todo o caminho de certificação que liga o certificado do usuário ao ponto mais alto de confiança.

O comprometimento das chaves criptográficas utilizadas nesta arquitetura são relativas aos pontos diretamente abaixo da AC comprometida. Caso aconteça o comprometimento de alguma AC intermediária, os pontos mais acima e outras AC intermediárias não serão afetadas. Então existe a possibilidade

A validação dos certificados se dará de forma análoga ao da Certificadora Única, em que parte diretamente de um ponto confiável pelo usuário.

A figura 2.4 ilustra a arquitetura de Lista de Confiança;

Essa arquitetura apresenta os mesmos problemas da arquitetura anterior, mas com um agravante, pois não poderá avisar a todos caso suas chaves criptográficas sejam comprometidas, pois não possui um controle de quem confia nela, já que nem todos os certificados existentes serão emitidos por ela. [MAR 05]

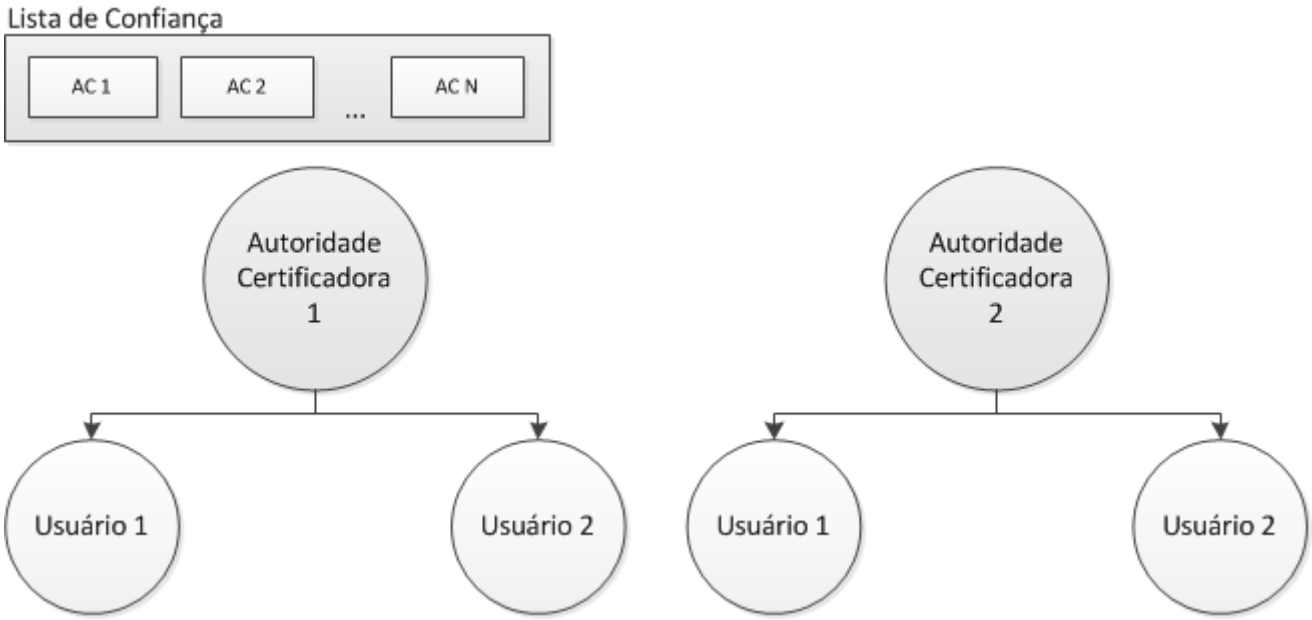

Figura 2.4: Arquitetura Lista de Confiança

2.3.4.3 Arquitetura Hierárquica

Uma evolução natural das outras arquiteturas citadas anteriormente é a arquitetura hierárquica em que se baseia numa rede de confiança mais complexa.

Assim como na arquitetura Certificadora Única, o usuário confiará em apenas um ponto, que é a Autoridade Certificadora Raiz, ponto inicial da confiança da arquitetura hierárquica. Ao confiar nessa AC Raiz, ele passará a confiar em todos os certificados emitidos por ela e pelas ACs intermediárias.

Ao tentar validar um certificado, o usuário precisará validar o caminho de certificação mais longo, até chegar no ponto de confiança. Esse caminho deve incluir todas as ACs intermediárias que existem entre o certificado até a AC Raiz.

A figura 2.5 ilustra a arquitetura Hierárquica de uma ICP.

A estrutura é bem simples e comumente utilizada em organizações, já que é mais fácil representar a real estrutura de confiança dentro de uma empresa ou organização. A segurança em relação as chaves públicas dessa arquitetura tem vanta-

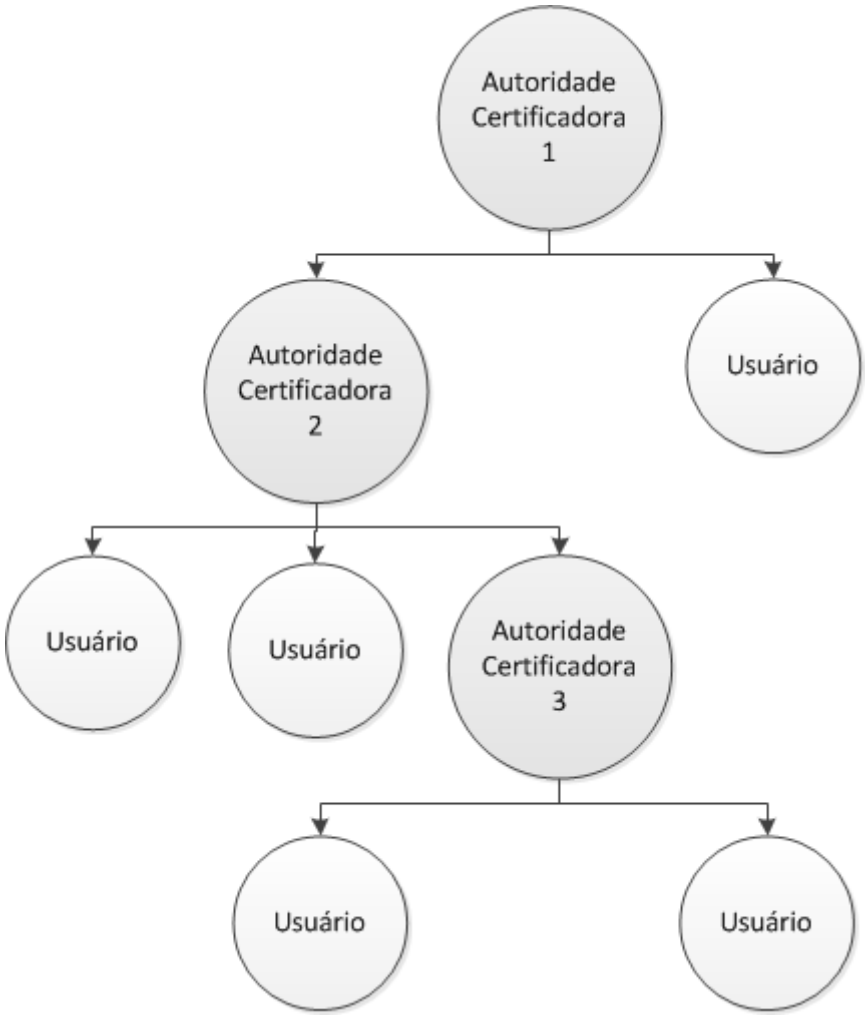

Figura 2.5: Arquitetura Hierárquica

gens sobre as outras. A grande vantagem é caso alguma AC intermediária seja comprometida, apenas os elementos logo abaixo dela que estarão comprometidos, enquanto que o resto da estrutura permanecerá intacto. Neste caso, como todos os usuários estão abaixo do mesmo ponto de confiança, há como notificar todos sobre o comprometimento daquele ponto. Para recuperar essa AC comprometida, basta recriar as chaves criptográficas e emitir novos certificados para ela mesma e todos os usuários abaixo dela. [RFC 08]

2.4 Conclusões

Como visto neste capítulo, existem diferentes necessidades para o uso das chaves criptográficas e por isso, as Autoridades Certificadoras precisam se adaptar e ter equipamentos apropriados para cada caso.

Um módulo criptográfico desenvolvido unicamente para um ambiente de baixa demanda deve ser modificado para ser utilizado em ambientes de alta demanda. Essa visão ajuda a encaminhar a aplicação e implementação dos estudos feitos nessa monografia.

Capítulo 3

ASI HSM

Neste capítulo será descrito o funcionamento do ASI-HSM, com a especificação das tecnologias, como foram implementadas, alguns protocolos e também a abordagem inicial do problema relacionado ao desempenho de assinaturas.

Os MSCs são utilizados em bancos, instituições e empresas e um grande problema é o fato de serem caixas pretas, já que não se sabe direito como foi implementado, como são os protocolos internos e normalmente a parte de auditoria deixa a desejar. Então para uma instituição ou até mesmo governo que deseja utilizar um MSC como proteção para suas chaves criptográficas, seria importante ter mais detalhes e controlar totalmente a utilização e ter o conhecimento de como tudo funciona. Ser implementado de maneira customizada da maneira que melhor convier. Então é aí que entra o projeto ASI-HSM que é um MSC feito em território nacional.

O ASI-HSM foi um projeto feito em parceria entre três entidades. São elas: LabSEC, RNP e Kryptus. O LabSEC e a empresa Kryptus ficaram responsáveis pela modelagem da parte lógica do hardware e os softwares envolvidos na gerência do ciclo de vida das chaves criptográficas. A RNP e a Kryptus foram as financiadoras do projeto e a Kryptus desenvolveu toda a parte de hardware. [JUN 09]

Uma das grandes vantagens de se utilizar o ASI-HSM é de que ao ser projetado, existia uma preocupação muito elevada em relação a segurança, auditoria e possui seus protocolos divulgados para que fique transparente os processos e funções

criptográficas envolvidas. [JUN 09]. Outros MSCs de mercado são como caixas pretas, não tendo seus protocolos divulgados e com isso, deixando os usuários sem mais informações sobre a segurança do produto.

O alto desempenho não era um requisito para o MSC então priorizou-se a segurança nos métodos implementados, sem se preocupar muito com a questão de desempenho. Ao ver que o ASI-HSM também poderia ser utilizado em ambientes de alta demanda, foi necessário fazer um estudo em cima de todo o MSC para investigar se o mesmo estava preparado para ser utilizado neste ambiente.

3.1 Perfis de Gerenciamento

O ASI-HSM possui três tipos de perfis para administração geral de todo o funcionamento. Cada perfil tem suas funções bem definidas e não pode mexer diretamente em outras funções que não são de suas responsabilidades que serão melhores explicados a seguir. Essa divisão possibilita um melhor controle das funções e futura auditoria. Os três perfis são:

- Administrador
- Operador
- Auditor

3.1.1 Administrador

Os administradores possuem a responsabilidade de executar operações relacionadas à configuração do módulo como um todo. No ASI-HSM não pode existir mais de um grupo de Administradores simultaneamente, já que as funções exercidas pelos mesmos são críticas. Este tipo de perfil é muito importante, não possibilitando a criação de outro, ou seja, não será possível tentar criar outro grupo que funcione em paralelo com o grupo atual. Caso seja necessário utilizar outro grupo de administradores, será necessário excluir o grupo antigo e criar um novo.

As funções que podem ser executas pelos administradores são:

- Apagar configurações
- Atualizar firmware
- Desligar
- Alterar data/hora
- Gerar/Recuperar backup
- Gerar par de chaves assimétricas

3.1.2 Operador

O grupo de operadores é encarregado de gerenciar as chaves criptográficas criadas pelos administradores. Quando um grupo de administradores gera um novo par de chaves, as mesmas serão delegadas a algum grupo de operador para que possa ser depois carregada e utilizada. Também fica a cargo dos operadores as políticas de uso das chaves onde podem ter diferentes princípios de utilização, por isso a necessidade de customizar o uso da chave também. Este grupo fica mais restrito a algumas ações relacionadas apenas as suas chaves, então é possível ter mais de um grupo de operadores funcionando em paralelo. Um exemplo disso é caso existam duas chaves criadas no ASI-HSM e o grupo de administrador delegou a chave 1 para o grupo de operadores 1 e a chave 2 para o grupo de operadores 2. Isso possibilita também um melhor controle do uso das chaves.

As funções executadas pelos operadores são:

- Carregar chave para uso
- Definir políticas de utilização da chave
  - Tempo de uso
  - Número de usos

3.1.3 Auditor

O grupo de Auditores tem a responsabilidade de auditar o ASI-HSM. Como os MSCs são de extrema importância e a segurança envolvidas em todas as operações é um requisito principal, todas as operações críticas são registradas em um log e guardadas para posterior consulta. Esses logs só podem ser exportados e visualizados pelos grupos de Auditores. É possível existir mais de um grupo funcionando em paralelo, mas para o funcionamento normal do MSC deve existir pelo menos um.

As principais funções que podem ser executadas pelos auditores são:

- Exportar logs
- Bloqueio do equipamento
- Recuperar backup

3.2 Interfaces de Comunicação

Para que o usuário utilize o ASI-HSM, ele necessita de uma interface para poder se comunicar e executar as funções desejadas. No ASI-HSM existem três tipos de interface: Inteface gráfica Java, Interface texto e Engine OpenSSL

A figura a seguir ilustra a comunicação entre os clientes e o ASI-HSM.

Figura 3.1: Comunicação entre a Interface e o ASI-HSM

3.2.1 Interface Gráfica Java

Existe a interface gráfica desenvolvida em Java para gerenciar todo o funcionamento do ASI-HSM. Ele foi desenvolvido utilizando o Swing Java, que é uma ferramenta para o renderizar/desenhar por conta própria os componentes.

Com esta interface é possível executar as funções de operador, auditor e administrador. Ou seja, todas as funções podem ser executar na mesma máquina, necessitanto apenas da autorização e autenticação de cada grupo.

A grande vantagem de utilizar esse tipo de interface gráfica é a sua portabilidade, pois necessita apenas a instalação da máquina virtual Java no Sistema Operacional desejado e conseguirá utilizar normalmente.

3.2.2 Interface de Texto

A interface de texto pode executar as mesmas funções que a interface gráfica Java. Foi desenvolvido para ser rodado no shell de um sistema operacional UNIX.

Hoje essa interface é mais utilizada como auxílio no desenvolvimento e implementação, pois o fato do software que gerencia o ASI-HSM ser rodado em um sistema UNIX, facilita na aplicação de testes, acelera a execução de algumas funções e com isso, consegue-se agilizar parte do processo de implementação.

3.2.3 Engine OpenSSL

As interface listadas e explicadas até agora servem mais para a gerência do ASI-HSM. A Engine OpenSSL possibilita o uso das funções disponíveis e chaves gerenciadas pelo MSC.

Uma engine é um tipo de interface que permite a um desenvolvedor customizar chamadas do OpenSSL, mas sem poder mudar os parâmetros das funções originais.

As engines podem ser compiladas como estáticas e dinâmicas. No caso do

ASI-HSM a engine é compilada como dinâmica, podendo ser carregada em tempo de execução, deixando assim mais fácil e ágil o processo de desenvolvimento do software que vai utilizar este recurso.

A Engine possui alguns campos para auxiliar no controle interno. Algumas delas são:

- id: identifica unicamente a engine que poderá ser utilizada. Este campo é a principal fonte de referência.

- init: função para inicialização da engine. Aqui podem ser definidas variáveis para controle e auxílio das próximas funções que podem ser executadas.

- ctrl: este campo serve como ponteiro para uma função customizada que pode ser implementada dentro da Engine OpenSSL. Um exemplo para este caso é quando um usuário deseja se conectar no MSC para executar determinadas operações. Nisso é necessário passar o IP e porta do MSC que se deseja conectar.

- finish: finaliza o uso da engine. Essa função auxilia no fechamento de contextos, liberação de memória entre outras operações de finalização.

- load_pubkey e load_privkey: serve para carregar a chave pública e chave privada respectivamente. Essa função tem como retorno uma EVP_KEY que é uma estrutura do OpenSSL que pode ser utilizada para gerenciar as chaves assimétricas carregadas.

Existem outros parâmetros mas que não serão abordados aqui.
No ASI- as operações que estão implementadas na Engine OpenSSL são:

- Operações que permitem o uso das chaves assimétricas RSA e ECDSA.

- Ações relacionadas a números aleatórios. Este tipo de operação utiliza recursos de hardware do ASI-HSM para obter um número aleatório com mais confiança.

3.2.3.1 Especificações da Engine OpenSSL que tem integração com o ASI-HSM

Para se comunicar via Engine OpenSSL com o ASI-HSM é necessário que tenha instalado na máquina o OpenSSL. Então será necessário carregar a engine e então chamar as funções implementadas para o ASI-HSM.

As funções implementadas foram:

- ENGINE_init
- ENGINE_finish
- ENGINE_load
- ENGINE_load_private_key
- ENGINE_load_public_key
- ENGINE_destroy
- RSA_finish
- RSA_verify
- RSA_sign
- RSA_pub_enc
- RSA_priv_dec
- RAND_bytes
- RAND_pseudo_bytes
- RAND_status

A figura a seguir ilustra a comunicação entre a Engine OpenSSL e o ASI-HSM.

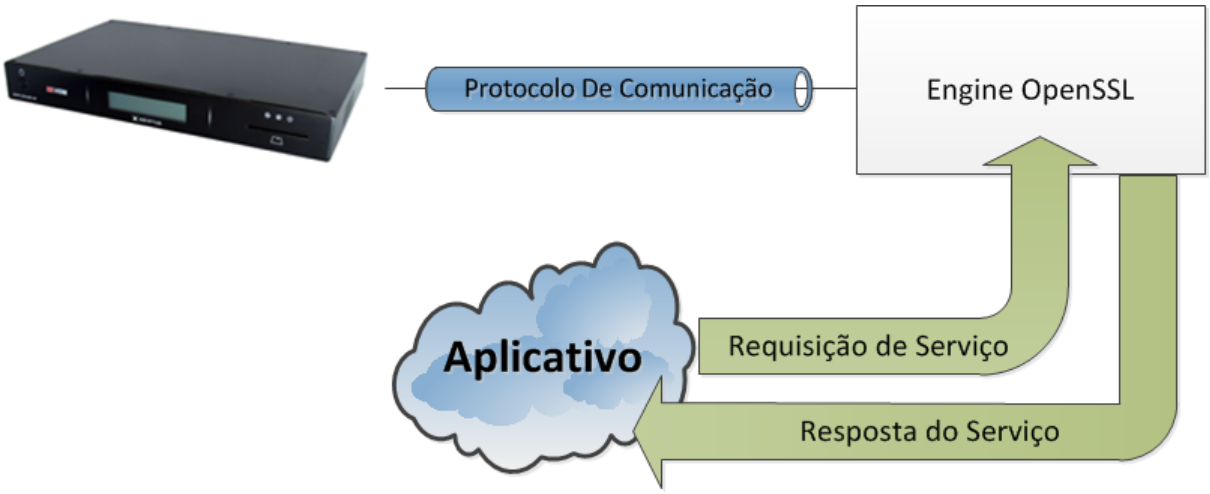

Figura 3.2: Comunicação entre a Engine OpenSSL e o ASI-HSM

3.3 OpenHSMd

3.3.1 Visão Geral

O ASI-HSM é um hardware para gerenciar o ciclo de vida das chaves criptográficas e para fazer isso precisa de um software que cuide e administre todas as operações relacionadas tanto na gerência quanto na utilização. Neste caso, o software que roda no ASI-HSM que tem essa função chama-se OpenHSMd. Este software foi desenvolvido pelo LabSEC e recentemente passou por um processo de homologação seguindo normas do MCT-7 [Inf 09]. O OpenHSMd foi implementado na linguagem C e é compilado para rodar no sistema operacional FreeBSD, que é o sistema operacional que roda dentro do ASI-HSM.

Seu projeto foi feito pensando em ser implantado primeiramente em Instituições de Ensino e ambientes de baixa demanda. Por isso, acabou-se dando prioridade as questões de segurança, auditoria e não tanto ao desempenho.

O OpenHSMd como relatado anteriormente, faz toda a gerência do ciclo de vida e controla o uso das chaves criptográficas. As operações criptográficas são feitas através de chamadas da biblioteca OpenSSL. Nesta sessão serão mostradas algumas

das principais funcionalidades e seus detalhes.

A figura 3.3 é uma ilustração simplificada da estrutura interna do ASI-HSM. Serve para mostrar a localização do software dentro do módulo.

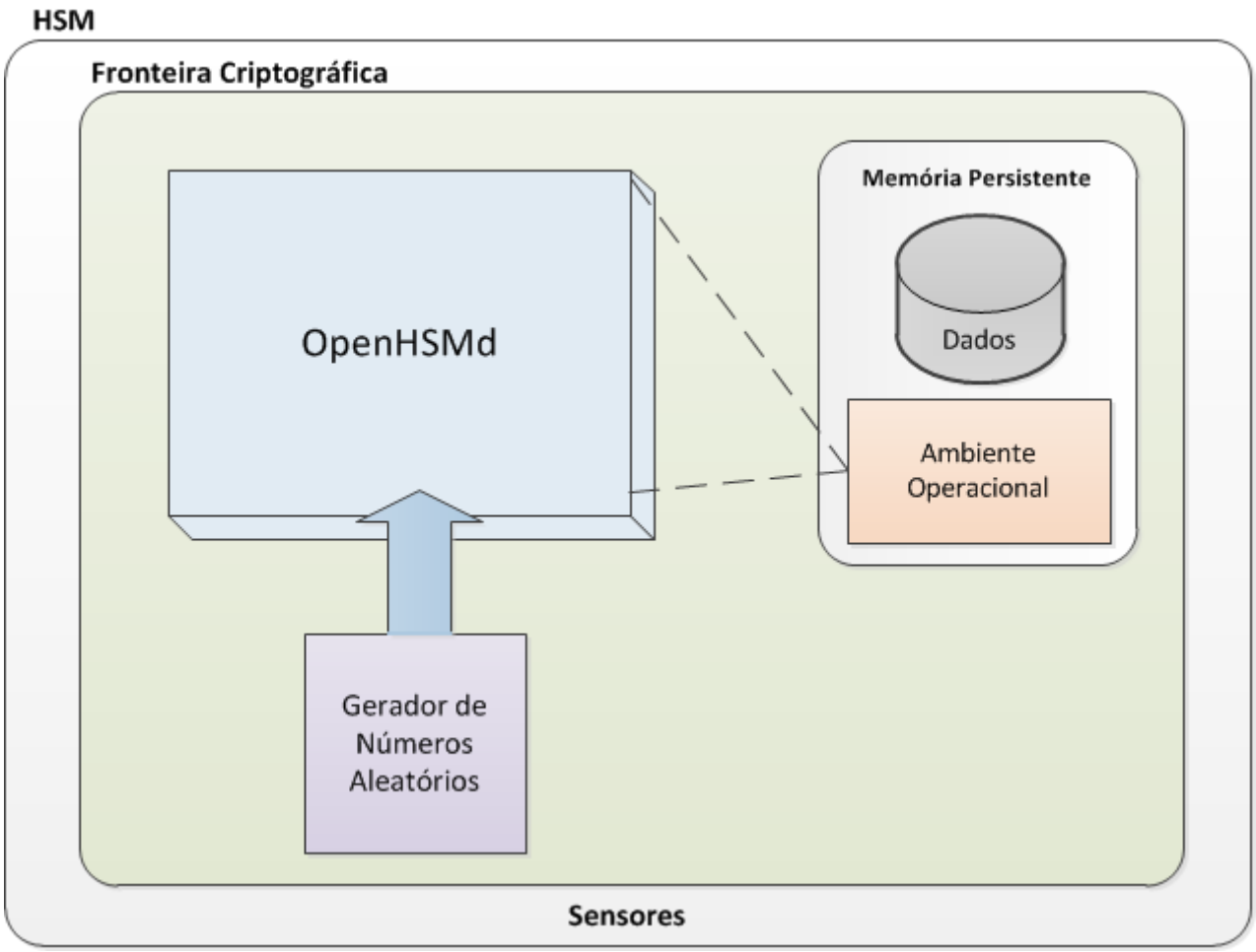

Figura 3.3: Localização do OpenHSMd dentro do Módulo do ASI-HSM.

3.3.2 Gerenciamento

Uma das principais funcionalidades é a parte do gerenciamento. Nessa etapa é onde são feitas todas as configurações iniciais do MSC e pode-se determinar inicialmente se este equipamento vai servir como backup ou não. Caso não tenha sido

configurado como backup, serão feitas as criações de cada grupo.

Nesta fase inicial também são executados os auto-testes para verificar a integridade tanto do software quanto do hardware do ASI-HSM. Caso algum desses auto-testes falhe, o ASI-HSM entrará em um estado de erro impossibilitando a execução de funções críticas referente as chaves criptográficas.

Como já especificado antes, só pode existir um grupo de administradores e esse vai ser o primeiro grupo a ser criado pois através dele que será possível criar os outros grupos. Nesta etapa pode-se criar um ou mais grupos de operadores e auditores. Após essa fase inicial de configurações, o MSC é reiniciado e então fica em estado operacional.

3.3.3 Controle de Estados

Uma das principais implementações feitas no ASI-HSM foi o controle de estados. Antes um usuário ao tentar utilizar o ASI-HSM não conseguiria ver de maneira rápida e prática em que estado o ASI-HSM se apresenta. Não sabendo se ja foi configurado, se está em um estado de erro ou se pode ser utilizado normalmente. [JUN 09]

Com esse controle de estados, pode-se definir mais claramente as funções que poderiam ser executadas em determinadas fases de operação.

Hoje o ASI-HSM possui os seguintes estados:

- Desligado
- Energizado
- Inicializado
- Autotestes
- Aguardando Configuração
- Backup

- Modo de Segurança
- Aguardando Criação dos Administradores
- Aguardando criação dos Auditores
- Aguardando criação dos Operadores
- Operacional
- Autenticação de um Administrador
- Atividade em execução de um Administrador
- Autenticação de um Auditor
- Atividade em execução de um Auditor
- Autenticação de um Operador
- Atividade em execução de um Operador
- Atividade em execução que não necessita de autenticação

3.3.4 Gerenciamento das chaves Criptográficas

Hoje no ASI-HSM os algorítmos assimétricos suportados para a geração de chaves são: RSA e ECDSA. As funções derivadas como assinatura e verificação também são suportadas para esses algorítmos.

Depois de criado os grupos de administradores, auditores e operadores ja é possível gerar chaves e delegar a custódia a determinado grupo de operador para conseguir utilizar futuramente. As principais funções de gerenciamento das chaves são:

- Criar chave e delegar a custódia a um grupo de operadores
- Carregar a chave podendo ter como parâmetros:

- Tempo de uso
- Número de usos

A função de criar chave é executada por um grupo de administradores. Nessa função é possível escolher dentre os algorítmos suportados do ASI-HSM. Depois disso escolhe-se o tamanho da chave ou tipo de curva dependendo do nível de segurança necessária envolvida na ocasião. Então é selecionado o grupo de operadores que terá a custódia da chave. Após esta fase de escolhas a chave é gerada e guardada em disco cifrada para um uso posterior.

Depois de criada, a chave para poder ser utilizada tem de primeiro ser carregada. Esta função é de responsabilidade do grupo de admnistrador custodiante. Ao solicitar o carregamente da chave, é necessário dizer os parâmetros para utilização da mesma. Neste caso, pode-se configurar o uso passando um tempo para utilização ou a quantidade de vezes. Então a chave estará disponível para ser utilizada.

Após os parâmetros serem ultrapassados, as chaves serão descarregadas impossibilitando o uso.

3.3.5 Auditoria

O procedimento de auditoria é muito importante para um MSC, já que ao sinal de qualquer operação irregular ou algum erro de operação, os registros são essenciais para se fazer uma avaliação e constatar possíveis irregularidades. Como o uso das chaves criptográficas administradas por um MSC é extremamente restrito, qualquer operação deve ser registrada. O uso irregular de uma chave pode ocasionar perdas e danos econômicos muito grandes, então o controle do uso é sempre muito rígido.

Os registros são administrados pelos gupos de auditores, onde podem extrair os registros das operações anteriores para fora do MSC. Registros também podem ser apagados, mas necessita-se a autenticação do grupo de auditores responsáveis por esse MSC. Isso possibilita um controle bem rígido, já que os outros grupos apesar de usarem funções críticas, não podem apagar os registros das operações que realizaram.

Além das operações realizadas, a data e o grupo responsável por estas ações são também registradas. Ou seja, existe um controle pleno da utilização do MSC, dficultando muito o uso indevido deste aparelho.

3.3.6 Utilização das chaves criptográficas

Depois que todo o ASI-HSM foi configurado, chaves criptográficas criadas e carregadas, este MSC está pronto para uso. As funções disponíveis para utilização das chaves são:

- Assinatura
- Verificação

Depois que as chaves foram devidamente carregadas, ja é possível fazer assinaturas com o MSC. Para realizar tal operação, o usuário terá que carregar a Engine OpenSSL do ASI-HSM, identificar qual chave quer utilizar e assim fazer uma requisição de assinatura via métodos da biblioteca OpenSSL.

O usuário então poderá utilizar um determinado número de vezes ou por um determinado tempo necessário pré-determinado no carregamento da chave. Assim que for descarregada, não será possível fazer mais nenhuma operação com a chave.

Depois que uma assinatura foi feita, é possível conferir e fazer a verificação com o ASI-HSM. Para isso chama-se a função responsável do OpenSSL para efetuar a verificação da assinatura do resumo criptográfico desejado.

3.3.7 Protocolo de Comunicação

Para que as interfaces conversem com o ASI-HSM e façam seus pedidos de funções, deve existir um padrão de comunicação, que possibilita a conexão e a troca de mensagens. Pode-se dizer que são regras estabelecidas antecipadamente para quando desenvolver uma nota interface ou novas funções saber como é o padrão e implementar seguindo-o.

No ASI-HSM essa comunicação é feita através de uma abstração de sockets do OpenSSL chamada BIO. Essa abstração implementa o protocolo TCP/IP e são aceitas conexões para transferência de dados através de Entrada/Saída. Se a conexão não foi estabelecida, mas a porta foi inserida corretamente, o BIO espera por uma nova conexão. [Ope]

A conexão para administração do ASI-HSM é feita também através de um canal seguro, utilizando SSL também chamando funções da biblioteca OpenSSL. [Ope]

Para distinguir os comandos e executar corretamente cada função, o protocolo espera strings com uma sequência pré-determinada para então dividí-la e encaminhar corretamente para a função correspondente.

Um exemplo de como funciona essa distinção através do conjunto de caracteres: ”adm genadm -n N -m M”. Neste caso, o OpenHSMd fará uma separação através dos espaços em branco entre cada conjunto de caracteres e guardará em um array cada pedaço do pedido. Então fará um encaminhamento através de cada pedaço que está no array. Neste exemplo o primeiro pedaço é ”adm”, então o resto do array será enviado para uma função que cuida das funções de administradores do ASI-HSM. Depois pega o segundo pedaço que é ”genadm”e executa a função encarregada de criar administradores. Com isso, faltará apenas os parâmetros necessários para criar os administradores, que neste caso é o N e M. Então passará pelo resto do array e guardará esses valores para executar corretamente a função. Com isso é possível fazer a comunicação entre a interface de administração e o ASI-HSM.

A Engine OpenSSL também tem o mesmo princípio de comunicação, só que o canal estabelecido não é seguro. Apenas utiliza a abstração BIO do OpenSSL.

O protocolo de comunicação entre a Engine OpenSSL e o ASI-HSM estabelece que será escrito no buffer de comunicação um número máximo de 2048 bits por vez. Então na primeira vez que vai escrever, é avaliado se o conteúdo está abaixo desse limite, caso esteja, a mensagem precisará ser enviada apenas uma vez, não precisando quebrar em partes e refazer a comunicação. E para confirmar a chegada da mensagem,

um sinal de conhecimento é enviado para quem escreveu a mensagem. Com isso é possível estabelecer a comunicação entre as interfaces e o ASI-HSM.

Funções do OpenSSL utilizadas no protocolo:

- BIO_write
- BIO_read
- BIO_new_accept
- BIO_do_accept
- BIO_pop
- SSL_write
- SSL_read
- SSL_new
- SSL_set_bio
- SSL_accept
- SSL_shutdown
- SSL_clear
- SSL_get_shutdown

Capítulo 4

Problemas e Propostas

4.1 Baixo Número de Assinaturas por Segundo

Como explicado na seção 2.3.3, existem diferentes necessidades e requisitos para os diferentes ambientes em que o MSC estará inserido e como o ASI-HSM foi projetado para atender ambientes de baixa demanda, não houve a necessidade de atender requisitos relacionados a otimização das assinaturas. Mas os problemas começaram a surgir na necessidade de atender a altas demandas de assinatura, onde a velocidade no processo é de extrema relevância.

Em ambientes onde podem existir até milhões de requisições de assinatura por dia, era de suma importância que o ASI-HSM pudesse usar de maneira mais eficaz seus recursos para atender de maneira satisfatória a essas demandas. Não existia nenhum teste mais preciso ou detalhado sobre as possibilidades de assinatura em que não levasse em conta apenas o hardware, mas também contando com o provedor de serviços criptográficos OpenHSM.

Uma das possíveis medidas que poderiam ser tomadas era implementar uma ferramenta para testar de maneira mais precisa a taxa de assinaturas por segundo que o ASI-HSM poderia suportar. Após os resultados obtidos, avaliar e fazer possíveis modificações para poder otimizar o processo de assinatura. Existem outras ferramentas que podem identificar a taxa de assinaturas por segundo que um hardware pode efe-

tuar, mas no caso do ASI-HSM, esse teste não seria completamente verdadeiro, pois ao solicitar uma assinatura, quem efetua a parte lógica e de comunicação entre o aplicativo e o ASI-HSM é o provedor de serviços criptográficos chamado OpenHSM, que gerencia o ciclo de vida das chaves criptográficas armazenadas internamente e também é responsável pelas assinaturas.

Após os testes iniciais será possível obter valores mais exatos sobre o estado atual do ASI-HSM e então realizar testes sobre o desempenho que o hardware teria sem influência do OpenHSM, trazendo consigo metas de desempenho a serem atingidas ao final do estudo e implementação.

4.2 Número Elevado de Registros

Em um ambiente onde o número de requisições pode chegar a uma taxa de milhões de assinaturas por dia, há a necessidade de se preocupar com o armazenamento dos registros das operações. O ASI-HSM possui um mecanismo de auditoria, visto na seção 3.3.5, onde todas as funções críticas são registradas para posterior consulta. Mas em ambientes onde o número de funções críticas, como assinar, poderia chegar na casa dos milhares ou até milhões, a preocupação com o armazenamento desses registros torna-se um problema, pois não havia um mecanismo de verificação automática do espaço disponível em disco na partição dos registros, podendo gerar perdas de informações e dados cruciais no momento da auditoria do ASI-HSM.

Uma das possibilidades seria implementar uma lógica onde registros mais novos sobrescrevessem os mais antigos, fazendo com que não extrapolasse o limite físico da partição. Essa proposta está baseada no fato de que os registros precisavam ser exportados e então apagados manualmente pelo grupo de auditores quando se aproximasse do limite do tamanho da partição, mas para que isso ocorresse, era necessário uma verificação constante do espaço disponível em disco para os registros.

Estabelecendo um possível limite de tamanho que ocuparia dentro da partição e calculando uma média de tamanho que os registros ocupam, seria possível calcular

um número máximo de registros. Com isso seria necessário fazer uma verificação a cada vez que fosse inserir um novo registro, em que caso alcançasse o número máximo de registros, deveria começar a sobrescrever os mais antigos. Após essa modificação, seria possível manter os registros das operações sem haver a necessidade de verificar constantemente o espaço disponível, exportar e ter que remover os registros mais antigos para que os novos não fossem perdidos.

4.3 Controle no Uso das Chaves

As chaves criptográficas devem possuir um controle rígido para acesso e utilização, já que são fundamentais no processo de assinatura digital, vide seção 2.1.2, em que a chave privada deve permanecer em um ambiente seguro para que seja dificultado ao máximo a utilização incorreta da mesma. Neste caso normalmente são criadas e mantidas dentro de um MSC, do qual possui ferramentas físicas e lógicas para proteger o uso indevido. Mas após a liberação para uso da chave, existe uma necessidade de se controlar quem está a utilizando, tentando garantir que apenas o perfil desejado tenha acesso e mais nenhum outro. Para isso deve-se tentar atrelar o uso da chave ao grupo responsável utilizando algum mecanismo de autenticação ou comprovação de posse.

O ASI-HSM possui uma pequena vulnerabilidade em que uma vez a chave carregada, qualquer aplicativo dentro da mesma máquina em que executaria um programa autorizado e que souber o nome da chave e tiver acesso a engine compilada, poderia utilizar de maneira indevida a chave criptográfica para efetuar assinaturas. Esse é um procedimento relacionado a um uso local do ASI-HSM, ou seja, em um ambiente supostamente controlado e Offline. Quando há um direcionamento deste MSC para ambientes de alta demanda, como ambientes em que uma Autoridade Certificadora pode ser Online, o problema persiste e com isso, há a necessidade de se buscar uma solução para evitar o uso indevido da chave.

Uma possível solução para este problema seria atrelar uma senha entre a

chave e o grupo de operadores responsável pela mesma. Com isso, ao se fazer uma requisição para uma assinatura, do qual uma chave será necessária, seria necessário repassar essa senha para liberar a utilização. Dessa forma, um aplicativo ou outro usuário que não tivesse a senha, não conseguiria utilizar de forma maliciosa a chave e garantindo o uso seguro da mesma para as assinaturas. Essa senha não poderá ser guardada e apenas fica em memória enquanto a chave estiver carregada, ou seja, em qualquer momento em que a chave for descarregada, seja no desligamento ou passando o período pré-determinado de utilização, a senha não será mais válida para liberar o uso.

Capítulo 5

Testes e Modificações

5.1 Testes Iniciais de Avaliação

5.1.1 Criação da Ferramenta de Testes

Como o ASI-HSM foi projetado para ambientes de baixa demanda, não haviam muitos testes e relatórios sobre a velocidade de assinaturas por segundo. Então a primeira etapa é avaliar o estado em que se encontra esse MSC, pegar o máximo de informações possíveis e traçar objetivos claros. Depois disso, fazer possíveis modificações no código, avaliar os resultados e fazer uma análise comparativa.

Não existia nenhuma ferramenta disponível para avaliar mais precisamente o desempenho do ASI-HSM em relação ao número de assinaturas por segundo, então o primeiro passo foi implementar uma ferramenta de testes customizada. A linguagem escolhida foi C, que é a linguagem do OpenHSMd e que também levou-se em conta devido a implementação da biblioteca OpenSSL, já que grande parte das chamadas dentro desse programa seriam através dessa biblioteca. O programa de testes foi projetado para ser rodado em um console shell em um sistema Unix.

Um dos primeiros passos desse programa é carregar dinamicamente, ou seja, em tempo de execução a engine do ASI-HSM através de chamadas da biblioteca OpenSSL. É através desta engine, melhor detalhada na seção: 3.2.3, que o aplicativo

irá se comunicar com o OpenHSMd e fazer as requisições das assinaturas. Os métodos da biblioteca responsável por essa carga são:

- ENGINE_load_dynamic();
- ENGINE_ctrl_cmd_string(e, ”SO_PATH”, ”caminho da engine compilada”, 0);
- ENGINE_ctrl_cmd_string(e, ”ID”, openhsmd, 0));
- ENGINE_ctrl_cmd_string(e, ”LIST_ADD”, ”1”, 0));
- ENGINE_ctrl_cmd_string(e, ”LOAD”, NULL, 0));
- ENGINE_ctrl_cmd_string(e, ”IP”, ”ip do hsm”, 0));
- ENGINE_ctrl_cmd_string(e, ”PORT”, ”porta do hsm”, 0));

Este programa de teste foi desenvolvido de maneira que os parâmetros sejam passados já no momento da execução, deixando assim os testes mais automatizados. No final de toda a execução é impresso no console o resultado em termos de assinaturas por segundo. Dependendo de como foram passados os parâmetros, é mostrado quantas assinaturas foram feitas ou por quanto tempo foi executado. Os parâmetros necessários para a realização dos testes são:

- Número de vezes que deseja assinar ou tempo passado em segundos
- Verificar ou não a assinatura a cada iteração
- Carregar ou não a engine em cada iteração
- Caminho para a Engine OpenSSL compilada
- Ip do MSC
- Porta do MSC

5.1.2 Configurando o MSC para os Testes

Para executar este programa, o ASI-HSM já deve estar previamente configurado, com os grupos criados, com as chaves criadas e devidamente carregadas. Esse MSC estava configurado com apenas um grupo de administrador, auditor e operador. Todos esses grupos foram criados com apenas um membro. Como parte inicial da sequência de investigação em cima da possível otimização da velocidade das assinaturas em cima do ASI-HSM, o MSC foi configurado normalmente e sem qualquer alteração em seu código.

Uma chave utilizando o algoritmo RSA de tamanho 1024 bits foi criada e carregada com o tempo e número de vezes de utilização infinito. Essas características das chaves foram inseridas para facilitar os testes, já que essas as mesmas seriam muito utilizadas.

5.1.3 Primeiros Testes

O primeiro passo que o programa de testes irá realizar é carregar a engine do ASI-HSM, verificar se os dados passados no momento da chamada do aplicativo estão corretos, carregar a chave desejada e então fazer as requisições de assinatura. O número de requisições será conforme os dados explicitados na chamada do aplicativo, assim como a maneira em que executará os testes, tanto por número de assinaturas quanto por um determinado tempo.

Os testes iniciais foram feitos com o resumo criptográfico já pronto, que foi obtido utilizando o algoritmo SHA1, evitando que algum tempo, mesmo que pouco, fosse gasto com os algoritmos de resumo, já que não fazem parte do escopo dos testes pois não envolvem diretamente o processo de assinatura.

Este teste foi executado várias vezes, utilizando diferentes parâmetros como tempo e número de vezes de assinatura e os resultados foram praticamente iguais. A tabela 5.1 ilustra essa situação.

Também foram realizados testes utilizando o OpenSSL diretamente do sistema operacional dentro do ASI-HSM. O OpenSSL possui uma função que testa e

Tabela 5.1: Testes Iniciais

Assinaturas	Tempo (segundos)	Assinaturas por Segundo
1	0,17	5,88
10	1,72	5,81
100	17,4	5,74
500	85,3	5,86
1000	170,7	5,85

avalia a performance da máquina com determinados algoritmos e neste caso, solicitou-se testes com chave RSA 1024 bits . O número de assinaturas por segundo foi superior aos testes anteriores e isso gerou desconfiança de alguns pontos de gargalos do ASI-HSM e gerando idéias sobre possíveis soluções.

O comando utilizado foi: ”openssl speed RSA 1024 -elapsed”

```
Doing 1024 bit private rsa's for 10s: 333 1024 bit private
RSA's in 10.03s
Doing 1024 bit public rsa's for 10s: 7072 1024 bit public
RSA's in 10.01s
OpenSSL 0.9.8e 23 Feb 2007

sign/s 33.2
verify/s 706.7
```

Outras velocidades também foram obtidas com o OpenSSL afim de se obter mais detalhes sobre a velocidade das assinaturas com chaves de diferentes tamanhos.

Com chaves RSA de tamanho 2048 bits.

```
Doing 2048 bit private rsa's for 10s: 58 2048 bit private
```

```
 RSA's in 10.04s
Doing 2048 bit public rsa's for 10s: 2203 2048 bit public
RSA's in 10.01s
OpenSSL 0.9.8e 23 Feb 2007

sign/s 5.8
verify/s 220.2
```

Também com chaves RSA de tamanho 4096 bits.

```
Doing 4096 bit private rsa's for 10s: 10 4096 bit private
RSA's in 10.87s
Doing 4096 bit public rsa's for 10s: 628 4096 bit public
RSA's in 10.01s
OpenSSL 0.9.8e 23 Feb 2007
built on: Sun Feb 24 16:11:39 UTC 2008


sign/s 0.9
verify/s 62.7
```

Neste caso pode-se interpretar que o Hardware do MSC, sem influências do software OpenHSMd teria como limite em torno de 33 assinaturas por segundo. Com esse resultado obtido através do OpenSSL e dos resultados anteriores com o programa de testes, pode-se concluir que existe uma disparidade grande entre os números de assinatura por segundo. A tabela 5.2 ilustra esses testes.

5.1.4 Testes com Ferramentas de Performance

Ferramentas de performance prontas e disponíveis na internet facilitam na avaliação do código em procura de pontos em que se pode aprimorar e evitar vazamentos de memória. Servem também para investigar o comportamento do programa

Tabela 5.2: Testes OpenSSL

	Tempo de Assinatura	Tempo da Verificação	Ass/Segundo	Ver/Segundo
RSA 1024 bits	0,03026s	0,001415s	33,2	706,7
RSA 2048 bits	0,173101s	0,004542s	5,8	220,2
RSA 4096 bits	1,087179s	0,015939s	0,9	62,7

em determinadas situações específicas que se queira testar. Normalmente são usados para avaliar a quantidade de vezes e o tempo que cada método foi utilizado e com isso, consegue-se estabelecer um perfil inicial para avaliação e então planejar melhorias para o programa.

No caso do ASI-HSM foram utilizados dois ”profilers”disponíveis gratuitamente na internet. Um deles foi o CPU profiler utilizado no Google com o nome de: google-pertools e outro chamado valgrind.

5.1.4.1 Performance com Google-PerfTools

Esta ferramenta da empresa Google tem como principal objetivo o uso de um novo método de alocar memória, mais rápido do que o da implementação padrão em C. Esta ferramenta vem em conjunto com outra que mostra detalhes sobre a execução do aplicativo, mostrando trechos de código e quais métodos foram chamados. Neste caso, é necessário colocar uma flag na hora de compilar o programa e adicionar uma linha de código no início e outra no final, especificando onde quer que comece e termine a observação.[Goo] Esses códigos são os seguintes:

```
ProfilerStart ( "Nome do arquivo que vai conter os dados
de avaliação ");
ProfilerStop();
```

E a flag de compilação é:

```
-lprofiler
```

Um exemplo de como compilar um programa para funcionar com o google-perftools seria:

```
gcc [...] -o meuprograma -lprofiler
```

Depois de adicionar as linhas e compilar, é só rodar o programa normalmente. Ao fim da execução do programa, a ferramenta google-perftools irá gerar um grafo direcionado apontando a sequência dos métodos que foram executados até o final do programa. A figura 5.1 ilustra um dos grafos gerados nos testes.

Avaliando esse grafo, os tempos e porcentagem em que ficou em cada nó, não foi nada significativo e pela experiência já adquirida no desenvolvimento do código, sabe-se não existe nenhum algoritmo muito complexo que fosse despender tanto tempo executando. Mas apesar de não haver nenhum algoritmo muito complexo, existia um número considerável de chamadas para o registro das operações de assinatura. Então a suspeita recai para um caso que ainda não foi abordado, que é a escrita dos registros no Banco de Dados.

5.1.4.2 Performance com Valgrind

A ferramenta Valgrind também é utilizada para avaliar possíveis vazamentos de memória, como está sendo usada a memória cache e faz a avaliação gráfica de todo o programa que está sendo executado. [Val]

Foi utilizado uma distribuição chamada Callgrind que além de informar os dados a respeito da cache, também mostra graficamente as chamadas de todos os métodos utilizados no aplicativo.

Para visualizar os gráficos é necessário fazer o download do aplicativo: kcachegrind, disponível no repositório padrão do Linux. Esse aplicativo tem uma interface bem detalhada e possibilita uma boa visão e avaliação do programa em questão.

Como esperado, os resultados foram muito parecidos com o perfil colhido através do Google-perftools. Então a próxima etapa seria realmente a retirada do registro de assinaturas.

5.1.5 Conclusões através dos testes iniciais

Com esse pequeno número de assinaturas por segundo e os testes efetuados com o OpenSSL Speed, constatou-se que o ASI-HSM deveria possuir gargalos com influência direta nas assinaturas e que deveriam ser revistos para possibilitar o aumento do número de assinaturas por segundo.

Também surgiram questões sobre a utilização da Engine em um ambiente de alta demanda, uma delas foi que após o carregamento da chave, qualquer usuário que carregar a Engine OpenSSL e souber o nome da chave, teria acesso e conseguiria utilizar a chave carregada, mesmo que não fosse o grupo responsável pela chave.

Outra situação em que poderia gerar problemas era o excesso de registros das operações, pois com a alta demanda, muitos registros seriam guardados no banco de dados, podendo alcançar o limite da partição e ocasionar perda de informações. Para evitar qualquer perda, era necessário pensar em alguma maneira de controlar logicamente a entrada de novos registros e não depender do grupo de auditores para liberar mais espaço na partição.

5.2 Implementação

5.2.1 Eliminando os Registros das Operações de Assinatura

Depois de avaliar os resultados das ferramentas de performance, uma suspeita era a escrita dos registros na operação de assinatura. Essa suspeita foi baseada no fato de que a escrita em memória acaba sendo demorada e acarretaria em um tempo de espera maior do que o necessário.

A cada assinatura é registrado no banco de dados uma nova linha para que depois seja auditado. Mas como a intenção é tentar deixar as assinaturas mais rápidas e não baixar muito o nível de segurança, uma das soluções abordadas foi retirar apenas os registros das assinaturas , pois os outros registros, como os do processo de configuração, não influenciam na velocidade das assinaturas.

Então após a retirada dos registros na operação de assinatura, foram executados novos testes e os resultados estão disponíveis na tabela 5.3

Tabela 5.3: Testes Sem Registros da Assinatura

Assinaturas	Tempo (segundos)	Assinaturas por Segundo
1	0,14	7,14
10	1,32	7,57
100	12,8	7,81
500	65	7,69
1000	130,4	7,66

Mesmo com esses resultados, um novo teste com os Profilers foi efetuado para melhores detalhes em relação aos métodos chamados.

Um novo teste foi efetuado com Profiler Google-perftool. Após configurar conforme explicado na seção 5.1.4.1, um novo grafo foi gerado e estudado. A figura 5.2 nos mostra o novo perfil obtido.

Uma análise inicial desse novo teste mostrou que as chamadas para registrar as operações de assinatura realmente diminuíram conforme o esperado e o número de assinaturas por segundo aumentou, mas ainda não estava muito claro onde estavam os outros possíveis gargalos.

Após esses testes, pode-se verificar que houve uma melhora mas não muito significativa e longe do obtido com o teste do OpenSSL Speed. Então uma nova suspeita foi estabelecida e recaia sobre a comunicação em rede entre o computador que estava fazendo a requisição da assinatura e o MSC. Neste caso, era necessário fazer um estudo sobre o protocolo de comunicação entre o OpenHSMd e a Engine OpenSSL e verificar se algum ponto do protocolo poderia ser modificado.

5.2.2 Modificando o Protocolo de Comunicação entre a Engine Openssl e OpenHSMd

Um dos pontos ainda não avaliado era o protocolo de comunicação e o tráfego em rede de dados entre o OpenHSMd e a Engine OpenSSL. Os testes de performance com as ferramentas não abrangiam essa área e mostravam um resultado que não é totalmente verdadeiro, pois nos testes feitos, não haviam gargalos onde poderiam estar se perdendo tanto tempo.

A partir deste momento a avaliação foi feita depurando o código e estudando o protocolo de comunicação, onde foi melhor detalhado na seção: 3.3.7.

Toda a comunicação feita entre o ASI-HSM e o aplicativo que vai fazer requisições de assinatura, é feita através da biblioteca OpenSSL, mas para isso acontecer, todo o protocolo tinha de ser respeitado. Então tanto no lado da Engine OpenSSL quanto do OpenHSMd tinham que saber se comunicar e ter algumas garantias de segurança. Era necessário também definir um tamanho de buffer para que fossem passadas as mensagens adiante e a comunicação fosse feita. Então o buffer foi delimitado tendo o tamanho de 2048 bits, mas algumas mensagens que trafegavam na rede poderiam ser maior e teriam que ser quebradas. Para isso, foi necessário fazer um algoritmo no caso em que a mensagem fosse maior que o tamanho do buffer, quebrasse do tamanho máximo, enviasse esse pedaço e tentaria enviar o restante em seguida, adotando o mesmo cuidado com o tamanho da mensagem enviada. Pseudo Algoritmo:

if $tamanhoTotalDaMensagem \leq 2048$ **then**
  envia mensagem
  espera sinal de recebimento da mensagem da engine
else
  while $2048 \leq tamanhoTotalDaMensagem$ **do**
    envia os 2048 primeiros bits da mensagem
    espera sinal de recebimento da mensagem da engine
    tamanhoTotalDaMensagem = tamanhoTotalDaMensagem - 2048

end while
if $tamanhoTotalDaMensagem \neq 0$ **then**
envia os bits restantes da mensagem
espera sinal de recebimento da mensagem da engine
end if
end if

Analisando este algoritmo pode-se notar que se a mensagem for maior que 2048 bits, terá que reenviar a mensagem várias vezes, desprendendo um precioso tempo do processo de assinatura. Outra análise feita é que o protocolo fica sempre esperando um sinal de recebimento da mensagem, ou seja, precisa enviar e depois esperar uma resposta. Todos esses aspectos comentados acabam influenciando na velocidade das assinaturas. Uma das possíveis soluções seria aumentar o buffer para 4096 bits para garantir que nenhuma mensagem tenha que ser retransmitida. Com isso seria possível eliminar grande parte dos problemas relacionados a transmitir a mensagem por partes e então uma verificação poderá ser feita para avaliar se a velocidade das assinaturas vai aumentar ou não.

Novamente os testes com o programa auxiliar foram efetuados. O algoritmo ficaria da seguinte maneira sem precisar retransmitir a mensagem:

if $tamanhoTotalDaMensagem \leq 4096$ **then**
envia mensagem
espera sinal de recebimento da mensagem da engine
end if

Essa mudança não influenciou na quantia de assinaturas realizadas por segundo. Pode-se concluir que o tamanho das mensagens nos testes não estava influenciando na velocidade das assinaturas por segundo. Então a suspeita passou para o sinal de recebimento da mensagem.

O próximo procedimento foi modificar novamente o algoritmo para não enviar e ficar esperando o sinal de recebimento da mensagem. O algoritmo ficou da seguinte maneira:

if $tamanhoTotalDaMensagem \leq 4096$ **then**

envia mensagem

end if

Dessa maneira o algoritmo não estaria tentando reenviar a mensagem e nem esperaria o sinal de recebimento da mensagem. Isso poderia ocasionar perda de dados, mas a abstração BIO do OpenSSL utiliza sockets para comunicação com o protocolo TCP, em que há a verificação se os dados foram enviados de forma correta, com a sequência apropriada e sem erros. Caso a mensagem não vá completa ou aconteça alguma alteração inesperada no conteúdo, o OpenHSMd não vai concluir o processo de assinatura e retornará um erro, garantindo que não funcione da maneira incorreta. [RFC 81] [BIO]

Após essa mudança foram realizados novos testes e o resultado foi muito positivo. A tabela 5.4 mostra o resultado das assinaturas por segundo:

Tabela 5.4: Testes Sem Registro da Assinatura e com o Protocolo Alterado

Assinaturas	Tempo (segundos)	Assinaturas por Segundo
1	0,035	28,57
10	0,33	30,3
100	3,6	27,77
500	17	29,41
1000	34,6	28,90

Após esses testes pode-se verificar claramente que o protocolo de comunicação entre o OpenHSMd e a Engine OpenSSL estava sendo o gargalo no número de assinaturas por segundo.

5.2.3 Sobrescrita dos Registros de Operação

O registro das operações em banco de dados é muito importante para a auditoria de um MSC. Caso algum procedimento tenha saído errado, os auditores podem depois conferir que grupo e quais ações foram executadas.

Cada operação crítica do MSC é registrada e guardada em um banco de dados interno que só pode ser visualizado e exportado pelo grupo de auditor. Mas quando o MSC parte para uma nova etapa, em que muitas requisições, muitas chaves e muitas operações serão realizadas, começa a surgir um problema em relação a partição que armazena os registros das operações.

No ASI-HSM existe uma partição, com tamanho de 55 megabytes [JUN 09], só para armazenar os registros das operações. O controle sobre essa partição fica sobre responsabilidade do grupo de auditores, que de tempos em tempos, deverá avaliar o espaço disponível e caso se aproxime do limite, o grupo necessita exportar e depois apagar todos os registros. Como o ASI-HSM foi idealizado para suportar um ambiente de baixa demanda, esse procedimento quase nunca seria realizado. Mas com com um alto número da demanda de assinaturas, poderia alcançar esses limites físicos da partição e, consequentemente, poderia gerar erros e perda de registros.

Uma das possíveis soluções seria implementar um registro das operações circular, ou seja, não existiria um limite de registros. O que existiria seria um limite lógico para então começar a sobrescrever os registros mais antigos. Dessa forma, não haveria a necessidade de excluir os registros, diminuiria a possibilidade de erros e tiraria um pouco a carga em cima dos grupos de auditores em verificar sempre o tamanho disponível na partição dos registros.

Para fazer essa mudança, primeiro levou-se em conta o tamanho total da partição, que é de 55 megabytes. Com isso, foi colocado uma margem de segurança de que o espaço ocupado pelos registros alcançariam no máximo um valor de 46.000.000 bits. É quase o valor de 44 megabytes, dando uma margem de aproximadamente 10 megabytes.

O próximo passo foi calcular quantos bytes cada registro ocupava em memória.

Tomando o tamanho original do banco de dados, foi feito uma inserção de um registro e calculou-se o tamanho novamente. Esse procedimento foi repetido várias vezes para pegar uma média, pois nem todos os registros possuem o mesmo tamanho. Ao final desses testes, obteve-se uma média de 200 bytes por registro.

Depois de obtido o tamanho de cada registro, pode-se obter a quantidade de registros máximo que se pode ter no banco de dados. Para isso, foi dividido o número máximo de espaço que os registros poderão ocupar pelo tamanho médio de cada registro. Com isso obteve-se o número de 230.000 registros, ou seja, o arquivo de registros do ASI-HSM poderá ter no máximo 230.000 registros.

Não havia a necessidade de se fazer verificações em relação ao número máximo de registros, mas com essa nova modificação será necessário identificar qual posição e identificador cada entrada deverá possuir, pois deverá apenas sobrescrever os registros mais antigos.

O algoritmo a seguir descreve como ficou o novo procedimento:

NUMERO_MAXIMO_DE_REGISTROS = 230.000
ultimoRegistro = buscar o identificador do último registro
if $ultimoRegistro \leq NUMERO\_MAXIMO\_DE\_REGISTROS - 1$ **then**
  inserir registro normalmente no Banco de Dados
else
  posicaoDoRegistro = ultimoRegistro - NUMERO_MAXIMO_DE_REGISTROS - 1
  inserir registro no Banco de Dados na posição: posicaoDoRegistro
end if

5.2.4 Segurança nos Métodos de Assinatura

Um ponto muito importante a ser abordado é a segurança envolvida no processo de assinatura, já que apenas otimizá-lo não irá deixar o ASI-HSM preparado

para um ambiente de alta demanda.

Para que o ASI-HSM passasse no processo de homologação em que esteve envolvido, precisou atender a vários requisitos de segurança. Um deles era o de garantir a segurança no uso das chaves através do requisito III.7.28 [Inf 09], descrito abaixo:

”REQUISITO III.7.28: [FIPS 140-2, 4.7.5] O módulo criptográfico deve associar a cada chave (simétrica ou assimétrica) armazenada o seu respectivo operador (pessoa, grupo, processo, servidor, etc).”

Esse requisito motivou a melhoria na utilização das chaves no processo de assinatura, pois além de deixá-lo mais robusto, também era de interesse das organizações envolvidas com o ASI-HSM que ele fosse homologado.

Um dos pontos críticos que já era de conhecimento dos desenvolvedores do software do OpenHSMd era a insegurança na utilização das chaves no processo de assinatura. Depois de carregada, sabendo-se o nome da chave, pode-se utilizá-la por mais de um aplicativo, sem que tenha um controle mais preciso. Logo as chaves poderiam ser utilizadas com um fim malicioso.

Para solucionar esse problema existiu a necessidade de atrelar uma senha entre o grupo de operador e a chave que era de sua responsabilidade. Com isso, ao solicitar uma assinatura utilizando essa chave, seria necessário passar a senha para que fosse liberada a sua utilização, assim garantiríamos um uso das chaves mais controlado.

O primeiro passo foi modificar a estrutura das chaves gerenciadas internamente pelo software OpenHSMd. Foi necessário adicionar uma nova propriedade na estrutura das chaves, pois as existentes não poderiam ser utilizadas para essa nova finalidade. Então a propriedade senha foi criada na estrutura e a partir de então, cada uma estaria atrelada a um número inteiro.

A seguir uma ilustração exemplificada de como ficou a estrutura já modificada com a senha atrelada a chave criptográfica.

```
estrutura chave{
nome
```

```
numero máximo de uso
quantas vezes já foi utilizada
tempo limite
tempo em que foi carregada
tempo em que deve ser fechada
senha
}
```

Para atrelar a senha a chave, o método de carregar chaves criptográficas também foi modificado. Ao solicitar o carregamento de uma chave específica, o grupo de operadores responsáveis deverá informar uma senha de no mínimo seis dígitos para ser atrelada a chave. A partir desse momento, a estrutura da chave já deverá conter a senha responsável pela sua liberação.

Foi necessário também implementar os mecanismos de autenticação no método de assinar, no software do OpenHSMd. Essa mudança foi feita com base na mensagem que virá através da comunicação entre o aplicativo que deseja assinar e o MSC, respeitando o protocolo de comunicação comentado na seção 3.3.7. A mensagem recebida deveria conter a senha para liberar o uso da chave. Essa liberação aconteceria da seguinte forma:

- Busca pela chave solicitada para a assinatura
- Obter a senha na estrutura da chave
- Comparar a senha recebida através da mensagem com a da estrutura da chave
- Caso as senhas sejam iguais, é liberado o uso da chave. Do contrário, um erro será retornado impossibilitando a utilização da chave para a assinatura

Outro ponto a se abordar é o lado da Engine OpenSSL que também deveria ser modificada para enviar a senha referente a chave. Como explicado na seção 3.2.3 , as assinaturas dos métodos implementados pelo OpenSSL não podem ser modificadas,

portanto, houve a necessidade de fazer uma abordagem parecida com a utilizada para passar o ip do MSC, com a qual a engine se comunica.

Para suportar essa nova funcionalidade foi necessário acrescentar um método que recebesse a senha através da chamada da engine. Com isso, essa senha era guardada e só repassada para o OpenHSMd na chamada dos métodos de assinatura. A limitação dessa abordagem é que a senha precisa ser enviada para a engine apenas uma vez, impossibilitando a mudança da mesma e a utilização de outras chaves durante a assinatura. Para realizar uma assinatura com outra chave, é necessário recarregar a engine e enviar uma nova senha.

Para o aplicativo que for utilizar esta Engine modificada, precisará fazer uma chamada através da biblioteca OpenSSL que se assemelha a maneira de passar o ip e porta do ASI-HSM.

A linha a seguir exemplifica como ficaria esse comando:

```
ENGINE_ctrl_cmd_string(e, "PWD", "senha para uso da chave", 0);
```

Esse comando deverá ser utilizado já nos primeiros métodos de carga da engine, assim, deixando a senha guardada em memória para ser utilizada no momento da assinatura.

Com essa mudança, seria garantido parcialmente a segurança na utilização das chaves para fazer as assinaturas. Em ambientes de alta demanda é de extrema importância esse controle e atrelamento ao grupo de operador para evitar o uso indevido das chaves criptográficas.

5.3 Resultados e Conclusões

5.3.1 Resultado da Otimização das Assinaturas

Após a suspeita de que o registro de operações estaria deixando a operação de assinar mais lenta, a retirada dos registros das assinaturas elevou um pouco o número de assinaturas por segundo , mas não a quantidade esperada.

A seguir uma tabela comparando a taxa de assinaturas por segundo antes e depois de retirar os registros das operações de assinatura:

Tabela 5.5: Tabela comparativa entre as implementações com e sem os Registros das Assinaturas. Os dados estão representados em Assinaturas por Segundo

Assinaturas	Com Registros	Sem Registros
1	5,88	7,14
10	5,81	7,57
100	5,74	7,81
500	5,86	7,69
1000	5,85	7,66

Pode-se perceber que houve um ganho, mas ainda longe das média encontrada na seção 5.1.3 utilizando o OpenSSL Speed, que era de aproximadamente 33 assinaturas por segundo utilizando chaves RSA de tamanho 1024 bits.

Depois de perceber o pouco ganho obtido conforme a seção 5.2.1, vários testes levaram a crer que uma possível causa para o pouco desempenho fosse o protocolo de comunicação entre a Engine OpenSSL e o OpenHSMd.

Os resultados após essa alterações foram bem próximos da expectativa conforme a seção 5.1.3. A tabela 5.6 mostra a comparação depois que o registro de operações foi retirado e o protocolo de comunicação entre a Engine OpenSSL e o OpenHSMd foi alterado:

Ao final dos testes, pode-se perceber nitidamente que o desempenho das assinaturas estava sendo prejudicado principalmente pelo tipo de protocolo utilizado para comunicação entre a Engine e o OpenHSMd. Com isso, bons resultados foram obtidos e agora a taxa de assinaturas por segundo está bem próxima do obtido com o OpenSSL Speed, ou seja, próximo do limite do hardware.

A tabela 5.7 faz uma comparação das taxas de assinaturas por segundo depois que as modificações foram sendo feitas, até chegar a um número próximo do ideal. Com todas essas mudanças nos protocolos e retirada dos registros no processo

Tabela 5.6: Tabela comparativa entre as implementações com a forma original e modificada do protocolo de comunicação entre a Engine OpenSSL e o OpenHSMd, ambas sem os Registros das Assinaturas. Os dados estão representados em Assinaturas por Segundo

Assinaturas	Protocolo Original	Protocolo Modificado
1	7,14	28,57
10	7,57	30,3
100	7,81	27,77
500	7,69	29,41
1000	7,66	28,90

de assinatura, o ASI-HSM teve um aumento de 500% na velocidade das assinaturas realizadas, podendo assim ser melhor implantado em ambientes de alta demanda.

Tabela 5.7: Tabela comparativa entre todas as modificações. Os dados estão representados em Assinaturas por Segundo.

Assinaturas	Com Registros	Sem Registros	Sem Reg. e Protocolo Alterado
1	5,88	7,14	28,57
10	5,81	7,57	30,3
100	5,74	7,81	27,77
500	5,86	7,69	29,41
1000	5,85	7,66	28,90

5.3.2 Melhoria na Segurança do Uso das Chaves Criptográficas

Assim como foi comentado na seção 5.2.4, era de extrema importância a garantia de associar uma chave a um determinado operador. Os principais dados que eram passados no método de assinar eram relacionados ao nome da chave, o resumo criptográfico e algoritmo usado no resumo. Mas agora será necessário enviar também

a senha associada a chave que foi carregada. Hoje esse recurso é opcional e pode ser colocado no momento da configuração inicial do OpenHSMd. Este recurso além de melhorar a segurança e deixar o ASI-HSM mais apropriado para o uso em ambientes de alta demanda, ajudou a cumprir o requisito da homologação, processo em que se encontrava enquanto as pesquisas descritas neste trabalho de conclusão estavam sendo desenvolvidas.

5.3.3 Mudança nos Registros das Operações

Para garantir um bom uso em ambientes de alta demanda, o ASI-HSM deveria estar preparado para um alto número de registros das operações. A mudança na maneira como os registros eram guardados no Banco de Dados foi essencial, diminuindo a possibilidade de problemas e a preocupação dos grupos de Auditores, pois não há mais a necessidade de verificar constantemente o espaço disponível para registros. Outro ponto positivo era a possibilidade de deixar o ASI-HSM com o registro de operações mesmo em ambientes de alta demanda, pois o limite físico dos registros não seria mais um problema, já que os registros mais antigos seriam sobrescritos.

5.3.4 Conclusões dos Resultados

Após todas essas mudanças, o ASI-HSM hoje está preparado para ter um bom desempenho em ambientes de alta demanda e não deixando de lado a segurança envolvida no processo de assinatura.

Hoje o ASI-HSM é customizável e isso ajuda muito em sua distribuição, pois se fossem limitados a um só tipo de MSC, teriam que ser configurados individualmente, dificultando na manutenção do software, hardware e do suporte técnico. Com essa nova maneira de configuração, a instituição que deseja utilizar o MSC estudado nesse trabalho, poderá adquirir um só tipo, sem se preocupar se atenderá ou não a sua necessidade. Basta configurar conforme necessário e utilizar normalmente. Assim, todo MSC pode ser configurado para ser utilizado tanto em ambientes de alta demanda como de baixa demanda. Além disso, será também opcional o uso dos registros das

operações de assinatura e senha na utilização das chaves.

Hoje as possibilidades de configuração do ASI-HSM são:

- Com ou Sem Registro das Assinaturas
- Utilizando senha ou não no uso das chaves
- Protocolo Otimizado ou Protocolo Original

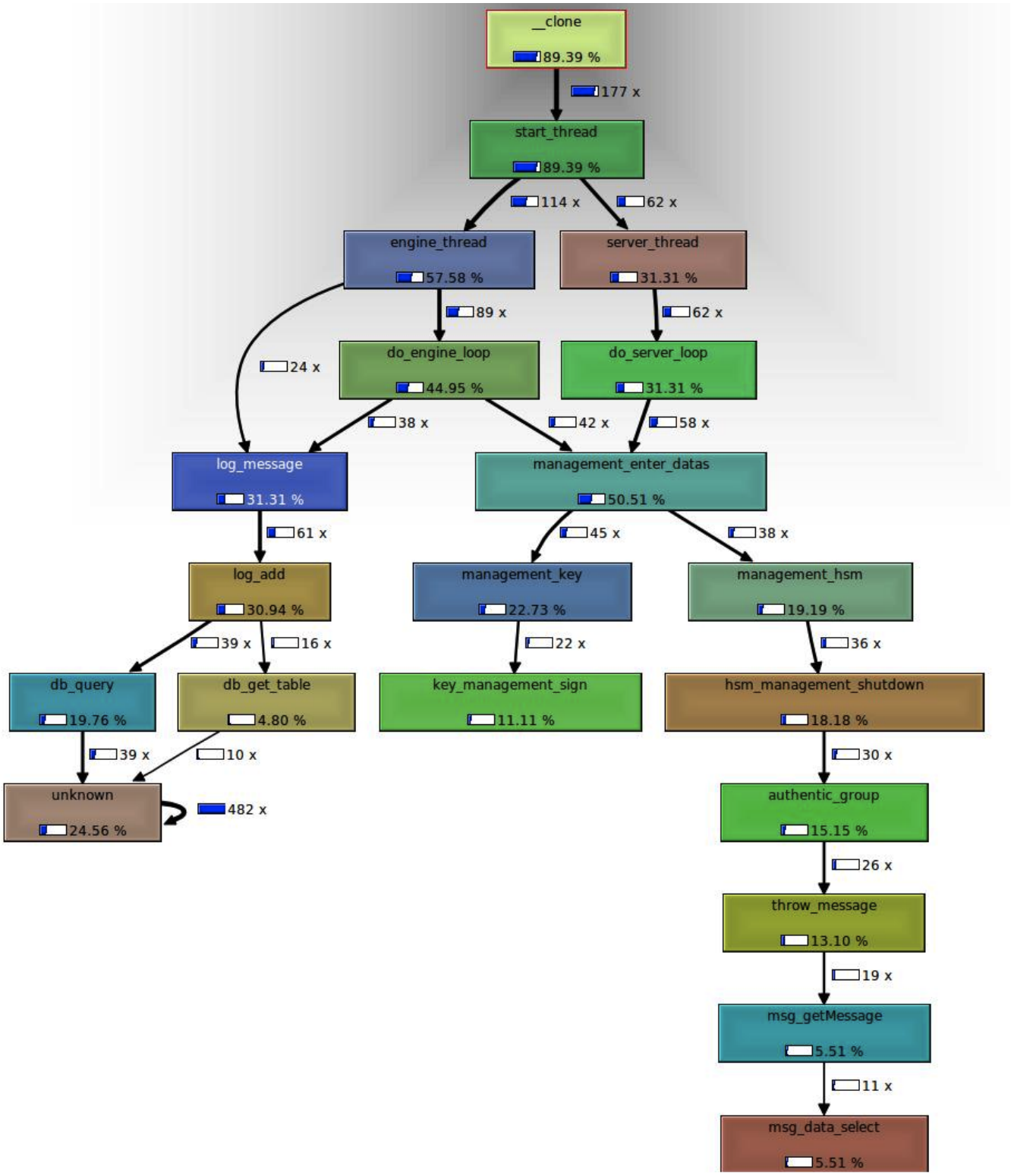

Figura 5.1: Perfil do OpenHSMd com Log e Protocolo Original

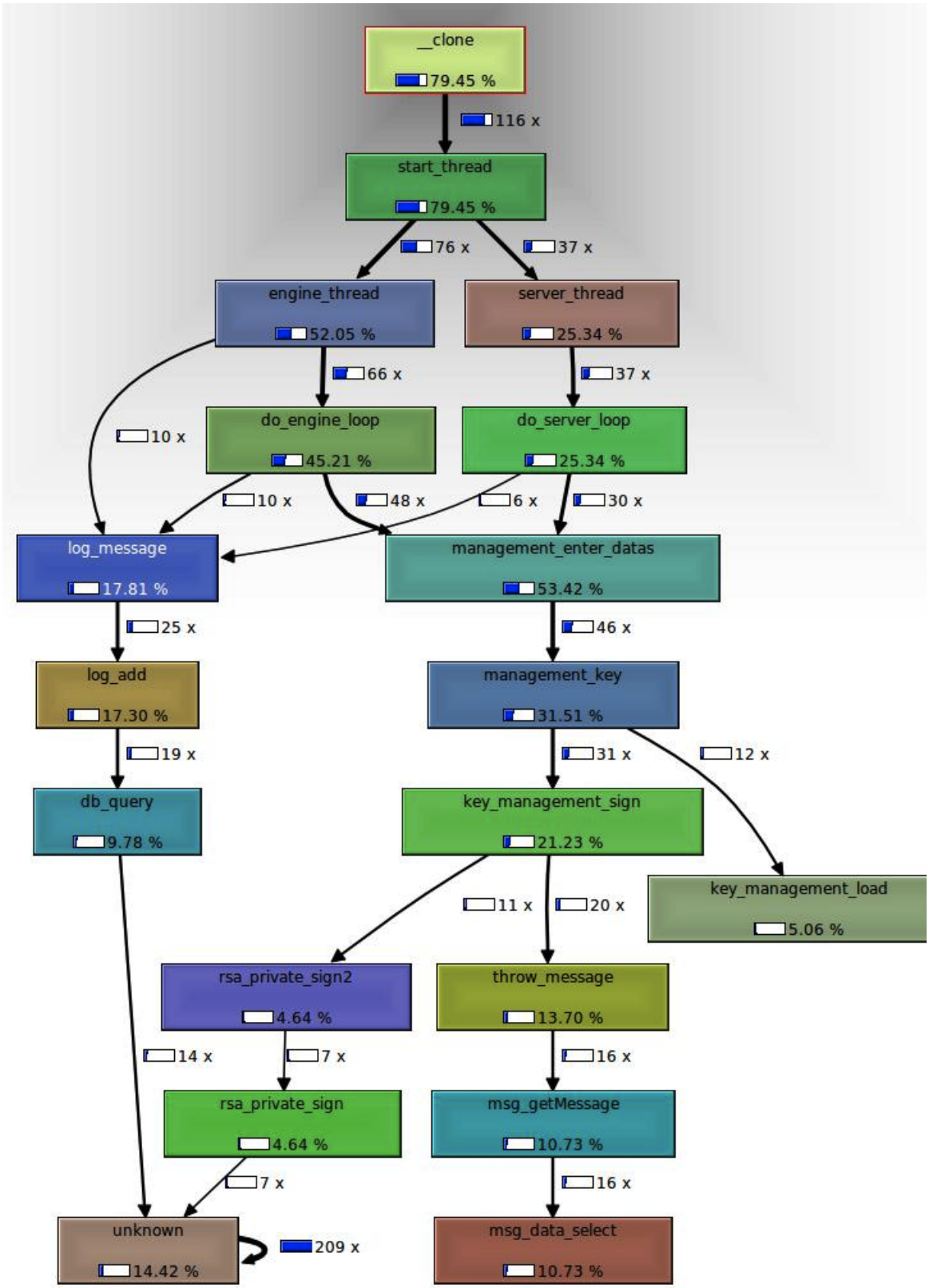

Figura 5.2: Perfil do OpenHSMd sem Registros das Assinaturas e Protocolo Original

Capítulo 6

Considerações Finais

O envolvimendo do autor desse trabalho de conclusão culminou em várias melhorias e novas implementações no ASI-HSM. Várias delas comentadas aqui neste trabalho que ajudaram hoje a deixá-lo customizável e adaptado para ambientes de alta demanda.

Hoje o ASI-HSM vem sendo utilizado em várias instituições e também na ICP-Brasil e, por isso, todas as melhorias em decorrência desse trabalho serão aproveitadas e utilizadas para melhorar e facilitar a vida de muitas pessoas envolvidas nesses projetos.

Entre os pontos positivos desse trabalho, pode-se considerar os novos meios em que o ASI-HSM poderá ser utilizado. Antes mais voltado para ambientes de baixa demanda, como por exemplo as Autoridades Certificadoras Offline, hoje poderá ser utilizado para meios de alta demanda, como por exemplo as Autoridades Certificadoras Online. Poderá também auxiliar nos processos judiciários em que necessitam de alta demanda e que utilizem o ASI-HSM, entre outras utilidades.

Outro ponto importante a se comentar é o processo de Homologação que o ASI-HSM passou. Nos últimos anos o autor desse trabalho de conclusão esteve envolvido diretamente com esse processo e além das funcionalidades implementadas e comentadas nesse trabalho, muitas outras foram colocadas em prática para deixar o ASI-HSM em conformidade com os requisitos necessários. Hoje o ASI-HSM está

homologado em nível 3 e é o único MSC nacional com essa conquista.

Esse trabalho de conclusão servirá também como consulta a futuros integrantes do projeto para melhor entender algumas funcionalidades e também para futuros trabalhos de conclusão de curso.

Entre todos os ganhos comentados, todo o conhecimento obtido para fazer todo esse estudo e modificações para o autor desse trabalho é também de grande importância.

6.1 Trabalhos Futuros

Hoje o ASI-HSM está limitado no número de assinaturas por segundo devido ao hardware utilizado e ao suporte limitado da utilização multi-thread da Engine OpenSSL. Então um possível trabalho futuro é a melhoria no hardware do ASI-HSM.

Possíveis soluções:

- Processador mais rápido
- Acelerador Criptográfico

Outra melhoria em relação ao já comentado acima, seria aprimorar a implementação da Engine OpenSSL do ASI-HSM para que possa suportar melhor os aplicativos multi-threads.

Como o ASI-HSM poderá ser utilizado para ambientes de alta demanda, um possível trabalho futuro é a de balanceamento de cargas, já que apenas um MSC poderá não ser suficiente para a demanda ou caso a utilização de maneira mais distribuída seja necessária.

Melhorar a ferramenta de teste implementada para este trabalho e descobrir novas ferramentas para avaliar mais profundamente os métodos do ASI-HSM.

Estudar as threads hoje utilizadas no OpenHSMd e avaliar se podem ser modificadas afim de melhorar o desempenho e a segurança em geral do ASI-HSM.

Referências

[BAR 10] BARKER, E.; ROGINSKY, A. **The Transitioning of Cryptographic Algorithms and Key Sizes**. 2010. Paper.

[BIO] BIO Socket. **OpenSSL: Documents, BIO_s_accept**. Disponível em <http://www.openssl.org/docs/crypto/BIO_s_accept.html>. Acesso em: 20 de maio de 2011.

[Goo] Google. **Google perftools**. Disponível em <http://code.google.com/p/google-perftools/>. Acesso em: 20 de maio de 2011.

[HOU 01] HOUSLEY, R.; POLK, T. **Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure**. Wiley, 2001.

[ICP 09] ICPEDU. **ICPEDU**. https://www.icp.edu.br.

[Inf 09] Infra-estrutura de Chaves Públicas Brasileira. **Manual de Condutas Técnicas 7 - Volume I: Requisitos, Materiais e Documentos Técnicos para Homologação de Módulos de Segurança Criptográfica (MSC) no Âmbito da ICP-Brasil**. Disponível em <http://www.iti.gov.br/twiki/pub/Homologacao/Documentos/MCT7_-_Vol.I.pdf>. Acesso em: 20 de maio de 2011.

[Ins 09] Instituto de Tecnologia da Informação. **ICP-Brasil**. Disponível em <http://www.iti.gov.br/twiki/bin/view/Certificacao/WebHome>. Acesso em: 20 de maio de 2011.

[JUN 09] JUNIOR, A. B. Aprimoramento de um hsm para homologação na icp-brasil. Universidade Federal de Santa Catarina, 2009. Trabalho de conclusão de curso (graduação em ciências da computação).

[Kry 09] Kryptus. **KRYPTUS - Engenharia Criptográfica**. Disponível em <http://www.kryptus.com>. Acesso em: 20 de maio de 2011.

[Lab 09a] LabSEC. **ASI-HSM**. https://projetos.labsec.ufsc.br/openhsmd.

[Lab 09b] LabSEC. **Laboratório de Segurança em Computação**. Disponível em <http://www.labsec.ufsc.br>. Acesso em: 20 de maio de 2011.

[MAR 05] MARTINA, J. E. **Projeto de um Provedor de Serviços Criptográficos Embarcado para Infra-estrutura de Chaves Públicas e suas Aplicações**. Universidade Federal de Santa Catarina, 2005. Dissertação de Mestrado.

[Mar 07] Martin Kiaer. **A Microsoft PKI Quick Guide**. Disponível em <http://www.windowsecurity.com/articles/Microsoft-PKI-Quick-Guide-Part2-Design.html>. Acesso em: 20 de maio de 2011.

[MAR 10] MARTINS, L. G.; WERLANG, F. C. Sistema de gerenciamento de certificados offline. Universidade Federal de Santa Catarina, 2010. Trabalho de conclusão de curso (graduação em ciências da computação).

[Mau 11] Maurício Augusto Coelho. **Diário Oficial da União**. Disponível em <http://www.jusbrasil.com.br/diarios/26438725/dou-secao-1-03-05-2011-pg-2>. Acesso em: 20 de maio de 2011.

[Mic] Microsoft. **Descrição de criptografia simétrica e assimétrica**. Disponível em <http://support.microsoft.com/kb/246071/pt-br>. Acesso em: 20 de maio de 2011.

[Ope] OpenSSL. **BIO - OpenSSL**. Disponível em <http://www.openssl.org/docs/crypto/bio.html>. Acesso em: 20 de maio de 2011.

[RFC 81] RFC 793. **TRANSMISSION CONTROL PROTOCOL**. Disponível em <http://www.ietf.org/rfc/rfc793.txt>. Acesso em: 20 de maio de 2011.

[RFC 08] RFC 5280. **Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**. Disponível em <http://www.ietf.org/rfc/rfc5280.txt>. Acesso em: 20 de maio de 2011.

[RIV 78] RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. M. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. 1978. Paper.

[RNP 09] RNP. **Rede Nacional de Ensino e Pesquisa**. Disponível em <http://www.rnp.br>. Acesso em: 20 de maio de 2011.

[SUT 11] SUTIL, J. M. **Gestão Segura de Múltiplas Instâncias de uma mesma Chave de Assinatura em Autoridades Certificadoras**. Universidade Federal de Santa Catarina, 2011. Mestrado em programa de pós-graduação em ciência da computação.

[Val] Valgrind. **Callgrind**. Disponível em <http://valgrind.org/docs/manual/cl-manual.html>. Acesso em: 20 de maio de 2011.

[VIE 02] VIEGA, J.; MESSIER, M.; CHANDRA, P. **Network Security with OpenSSL**. O'Reilly Media, 2002.