UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Túlio Cícero Salvaro de Souza

Aspectos Técnicos e Teóricos da Gestão do Ciclo de Vida de Chaves Criptográficas no OpenHSM

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Ricardo Felipe Custódio, Dr.
Orientador

Florianópolis, Outubro de 2008

Aspectos Técnicos e Teóricos da Gestão do Ciclo de Vida de Chaves Criptográficas no OpenHSM

Túlio Cícero Salvaro de Souza

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Frank Augusto Siqueira, Dr.
Coordenador do Curso

Banca Examinadora

Prof. Ricardo Felipe Custódio, Dr.
Orientador

Prof. Jeroen Antonius Maria van de Graaf, PhD.

Prof. Joni da Silva Fraga, Dr.

Prof. Michael Anthony Stanton, Dr.

Prof. Olinto José Varella Furtado, Dr.

Prof. Renato da Silveira Martini, PhD.

”Me abafa Dalila ... que eu estou abanado!”

Marco Aurélio Salvaro de Souza 2005.

Ao meu grande avô Silvino.

Agradecimentos

Primeiramente, gostaria de agradecer a toda minha família, principalmente aos meus pais, Mariano e Ilda, e a Giani, que se tornou minha esposa durante a execução deste trabalho. É o apoio de vocês que garante o sucesso de todas minhas escolhas de vida.

Também sou grato ao meu orientador, professor Custódio, que está pronto para ajudar a qualquer momento, mantendo sempre a situação sobre controle. Sua experiência serve de guia para os membros do LabSEC. Membros estes que, mesmo a um oceano de distância, continuam apoiando e ajudando na minha caminhada de descobertas. Com certeza todos ganham muito convivendo neste ambiente de aprendizado.

Não poderia deixar de citar minha gratidão ao professor, tio, padrinho e amigo Olinto. Ele vêm me guiando desde a escolha do curso de graduação ideal. Nunca vou esquecer o dia que estava em sua sala, desabafando sobre o meu descontentamento com o tema do meu trabalho de conclusão de curso. Ele virou pra mim e disse: "por quê você não vai ali conversar com o Professor Custódio, eu acho que você tem o perfil para trabalhar na área de segurança". Obrigado pelas dicas certas nas horas certas.

E finalmente, agradeço a Rede Nacional de Ensino e Pesquisa, entidade que vêm promovendo o uso de tecnologias avançadas nas instituições parceiras, proporcionando aos professores e pesquisadores ferramentas e meios de grande valia na pesquisa e ensino no Brasil.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AC	Autoridade Certificadora
ACT	Autoridade de Carimbo de Tempo
AES	Advanced Encryption Algorithm (Algorítmo de Cifragem Avançada)
ASN	Abstract Syntax Notation (Notação para Sintaxe Abstrata)
ANSI	American National Standards Institute (Instituto Nacional de Padrões Americanos)
API	Application Program Interface (Interface de Programação de Aplicativos)
AR	Autoridade de Registro
BIO	I/O Abstraction (Abstração de Entrada e Saída)
CBC	Cipher Block Chaining (Encadeamento de Blocos Cifrados)
CC	Common Criteria (Critérios Comuns)
CSP	Critical security parameter (Parâmetro Crítico de Segurança)
DER	Distinguished Encoding Rules
DES	Decryption and Encryption Standard (Padrão de Cifragem e Decifragem)
EAL	Evaluation Assurance Level (Nível de Garantia de Avaliação)
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard (Padrão Federal de Processamento de Informações)
GSM	Global System for Mobile communications (Sistema Global para Comunicações Móveis)

GT	Grupo de Trabalho
HSM	Hardware Security Module (Módulo de Segurança Criptográfica)
ICP	Infra-estrutura de Chaves Públicas
ICPEDU	Infra-estrutura de Chaves Públicas Educacional
ICP-Brasil	Infra-estrutura de Chaves Públicas Brasileira
IEEE	Institute of Electrical and Electronics Engineers (Instituto de Engenharia Elétrica e Eletrônica)
ISO	Internal Organization for Standardization (Organização Internacional para Padronização)
ITI	Instituto Nacional de Tecnologia da Informação
JCA	Java Communications API (API de comunicação Java)
JCE	Java Cryptography Extension (Extensão de criptografia JAVA)
LCR	Lista de Certificados Revogados
LEA	Laboratório de Ensaios e Auditoria
MCT	Manual de Condutas Técnicas
NIST	National Institute of Standards and Technology (Instituto Nacional de Padrões e Tecnologia)
NSF	Nível de Segurança Física
NSH	Nível de Segurança de Homologação
OSSI	Open Source Software Institute (Instituto de Software de Código Aberto)
PEM	Privacy Enhanced Mail
PIN	Personal Identification Number (Número de Identificação Pessoal)
PKCS	Public Key Crypto System
PUK	Personal Unlocking Code (Código de Desbloqueio Pessoal)
RNP	Rede Nacional de Ensino e Pesquisa
RoHS	Restriction to the use of Hazardous Substances
RSA	Rivest, Shamir e Adelman
SGC	Sistema de Gerenciamento de Certificados Digitais

SHA	Secure Hash Algorithm (Algoritmo Seguro de Resumo Criptográfico)
SO	Sistema Operacional
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UFSC	Universidade Federal de Santa Catarina
UFMG	Universidade Federal de Minas Gerais
UG	Unidade Gestora
Unicamp	Universidade de Campinas
US	Unidade de Segurança
USB	Universal Serial Bus
WEEE	Waste from Electrical and Electronic Equipament

Lista de Símbolos

$\leq$ Menor ou igual que

$\oplus$ XOR / Ou exclusivo

Resumo

O OpenHSM é um protocolo aberto para gestão do ciclo de vida de chaves criptográficas em módulos de segurança criptográfica, voltado principalmente para implantação de Infra-estruturas de Chaves Públicas. Esta dissertação formaliza e apresenta os vários sub-protocolos que juntos permitem a gestão confiável das chaves criptográficas. Os protocolos foram implementados e embarcados em um hardware criptográfico especialmente desenvolvido para evitar o acesso ao material sensível dos mesmos. Deu-se especial atenção aos aspectos práticos da implementação tal como sua aderência as normas nacionais e internacionais – MCT-7 e FIPS PUB 140-2 – e sua interface de comunicação com a aplicação de gestão de certificados digitais. Também são tratados os processos de criação e recuperação de cópias de segurança do OpenHSM, possibilitando a continuidade do ciclo de vida das chaves gerenciadas mesmo em caso de falhas ou desastres.

Abstract

The OpenHSM is an open protocol that has been developed to manage the life cycle of the cryptographic keys in hardware security modules, mainly addressed to public key infrastructure deployment. This thesis formalises and presents all sub-protocols that together provide reliable management of cryptographic keys. The protocols have been implemented and embedded in a cryptographic hardware especially developed to avoid unauthorized access to its sensitive data. Special attention was given to the practical aspects of the implementation, such as adherence to national and international standards – MCT-7 and FIPS PUB 140-2 – and integration with certificate management systems. The processes of creation and recovery of backup copies are also included in this thesis, permitting the continuity of the management keys' life-cycle even in the event of hardware failures or disasters.

Capítulo 1

Introdução

Hardware ou módulos criptográficos[1] têm sido usados como âncoras de confiança em sistemas de informação e comunicação. São dispositivos especialmente projetados para o armazenamento e o processamento seguro de informações sensíveis, tal como chaves criptográficas. Entre os modelos mais conhecidos estão os cartões inteligentes ou smartcards e os módulos de segurança criptográfica (HSMs)[2]. Ambos são usados para a geração, proteção e processamento de chaves criptográficas. Os smartcards são mais simples que os HSMs em termos de funcionalidades, quantidade e qualidade dos mecanismos de proteção das chaves e registro dos eventos associados ao ciclo de vida das chaves criptográficas, capacidade de processamento e formato físico.

Os smartcards são amplamente usados como dispositivo de autenticação de clientes em aplicações bancárias e cidadãos em sistemas de governo eletrônico (e-CPF). Em alguns países europeus e estados norte-americanos, por exemplo, os smartcards são utilizados como cartão de identificação do cidadão ou licença de motorista.

Os HSMs, por sua vez, são dirigidos a sistemas mais complexos que requerem um maior rigor no controle e auditoria do material sensível armazenado e processado.

O grande mercado dos HSMs são as aplicações militares e bancárias. Nessas aplicações, os HSM são usados para processar informações de autenticação de

[1] Neste trabalho os termos hardware criptográfico e módulo criptográfico são usados indistintamente.
[2] HSM vem do Inglês e significa Hardware Security Module.

usuários, registros de eventos críticos, assinatura de mensagens e execução segura de código.

Um nicho pouco explorado mas importante para o uso de HSMs são as infra-estruturas de chaves públicas (ICP) e suas aplicações, tais como autoridades certificadoras (AC), autoridades de registro (AR) e autoridades de carimbo de tempo (ACT). Neste nicho, não seriam necessárias algumas das funcionalidades e potencialidades existentes nos HSM disponíveis no mercado, ao mesmo tempo em que precisariam de outras características não contempladas, tal como um tratamento mais elaborado da criação e recuperação de cópia de segurança do material sensível. Esta lacuna é exaustivamente explorada neste trabalho.

Para a garantia da qualidade e padronização dos serviços e mensagens processadas pelos hardwares criptográficos, estes podem e devem ser desenvolvidos segundo normas e recomendações de entidades tais como o *National Institute of Standards and Technology* (NIST) e a Organização Internacional para Padronização (ISO). Dentre as várias normas existentes, a FIPS 140-2 promovida pelo NIST trata especificamente de hardware criptográfico, com a criação de um serviço de homologação de tais dispositivos. O fabricante de um hardware criptográfico pode solicitar a homologação de seu equipamento segundo os requisitos e recomendações apostas na norma. Após a avaliação pelo NIST ou por uma entidade credenciada, o hardware criptográfico recebe um atestado de aderência. Este atestado é exigido por inúmeras organizações como requisito para a sua aquisição.

Entretanto, além de preocupações técnicas, deve-se considerar a questão de segurança nacional, principalmente quando o HSM é usado para proteger informações de governo, militares e de empresas brasileiras. Em determinadas aplicações, consideradas de segurança nacional, tais dispositivos devem ser de absoluta confiança. Não é suficiente ter uma declaração do NIST ou de outra entidade internacional atestando que o hardware criptográfico é seguro e respeita os requisitos, recomendações e boas práticas de um hardware a ser usado como âncora de confiança. É necessário que o HSM seja passível de auditoria - deve ser possível avaliar a segurança dos mecanismos criptográficos e, em particular, a geração de chaves criptográficas. Como tais dispositivos precisam ser natu-

ralmente lacrados para evitar que seja possível o acesso a informação sensível, tal auditoria não pode ser realizada. É necessário confiar-se cegamente na entidade certificadora e no fabricante.

Com esta preocupação em mente, o governo brasileiro, sabiamente, através do Instituto Nacional de Tecnologia da Informação (ITI), estabeleceu normas e criou laboratórios de ensaios e análises (LEA) para avaliação de sistemas e equipamentos a serem usados pela Infra-estrutura de Chaves Públicas Brasileira (ICP-Brasil) e suas aplicações. Foi criado uma série de normas de avaliação dos equipamentos e sistemas, com vários níveis de análise. A mais rigorosa requer o depósito do código fonte da aplicação e do sistema operacional embarcado.

Apesar do grande mercado e de questões de segurança nacional, o Brasil ainda está engatinhando em relação ao domínio tecnológico e fabril de dispositivos de hardware criptográficos. Duas louváveis iniciativas merecem destaque quanto ao desenvolvimento de competência nacional na área de hardware criptográfico: o programa João de Barro criado pelo ITI a ordem do Comitê Gestor da ICP-Brasil e a Infra-estrutura de Chaves Públicas para Pesquisa e Ensino (ICPEDU) da Rede Nacional de Ensino e Pesquisa (RNP)[1].

1.1 Infra-estrutura de Chaves Públicas para Pesquisa e Ensino

A RNP tem-se utilizado de grupos de trabalho, constituído por pesquisadores de instituições públicas e privadas escolhidos a partir de editais públicos periodicamente lançados, como forma de absorver e realizar inovação tecnológica, adquirir experiência e desenvolver ferramentas para melhorar e disponibilizar novos serviços para seus clientes, normalmente universidades e centros de pesquisa brasileiros.

Especificamente, em 2003, foi criado o Grupo de Trabalho de Infra-estrutura de Chaves Públicas (GT ICPEDU) com o objetivo de estudar e implantar um serviço piloto para a emissão de certificados digitais para fins educacionais e de pesquisa.

Participaram desse esforço as Universidades Federais de Santa Catarina (UFSC) e de Minas Gerais (UFMG) e a Universidade de Campinas (UNICAMP). O GT ICPEDU teve três edições. A primeira, GT ICPEDU I, teve como resultado o domínio da tecnologia de certificação digital, com o desenvolvimento de um sistema de gerenciamento do ciclo de vida de certificados digitais (SGCI). O GT ICPEDU II teve como objetivo o desenvolvimento de um HSM, enquanto, o GT ICPEDU III, preocupou-se com a guarda da chave privada dos usuários, com o desenvolvimento de smartcards virtuais.

Entre os serviços almejados pela ICPEDU estavam a utilização de certificados digitais para a autenticação dos usuários nos serviços de rede e a realização de assinatura digital de documentos eletrônicos em substituição ao papel, principalmente para viabilizar a utilização de formulários eletrônicos nos processos acadêmicos. É objetivo também, a disseminação da tecnologia de certificação digital, através de cursos, reuniões e participação ativa das entidades na concepção da Infra-estrutura de Chaves Públicas para Pesquisa e Ensino (ICPEDU). Vê-se a ICPEDU como uma versão acadêmico e de pesquisa da ICP-Brasil.

A ICPEDU é hoje condição e insumo básico para a integração e o compartilhamento de recursos das instituições de ensino e pesquisa brasileiros e entre estas, incluindo as instituições afins de outros países. Vários projetos associados a disponibilização de recursos computacionais pela Internet necessitam dos serviços prestados pela ICPEDU. Destaca-se o esforço da RNP na implantação de um sistema de federações de forma a compartilhar os processos de autenticação entre as instituições usuárias.

Um dos grandes desafios a ser superado pelos participantes na implantação da ICPEDU era o alto custo do HSM necessário à criação de ACs e ARs e suas aplicações. O custo era proibitivo para maioria dessas instituições. Além do custo, a tecnologia de fabricação de hardwares criptográficos não estava disponível no Brasil e haviam restrições quando a sua importação, devido ao caráter de "segurança nacional" que os governos dos países detentores dessa tecnologia impunham aos fabricantes. Devido a isso, a RNP aceitou a proposta do GT ICPEDU II para desenvolver, projetar e construir um HSM de baixo custo e que provesse os requisitos de segurança apostos na

FIPS 140-2.

Assim, foram estabelecidos os seguintes requisitos gerais para o HSM:

- baixo custo;
- código aberto para facilitar a auditoria;
- visando atender os requisitos dos componentes de infra-estruturas de chaves públicas (rastreabilidade de chaves e um sistema de auditoria forte);
- capacidade mínima de processamento de 10 assinaturas RSA/segundo;
- compatível com o Sistema de Gerenciamento de Certificados Digitais da ICPEDU (SGCI).

O estudo e projeto do HSM tem sido alvo de várias dissertações de mestrado e trabalhos de conclusão de curso na Universidade Federal de Santa Catarina. Ao mesmo tempo em que estudam e se formam na área de protocolos criptográficos e sistemas criptográficos embarcados, os alunos e professores se dedicam a desenvolver a tecnologia e o produto HSM da ICPEDU (ASI-HSM). O software de gerenciamento do ciclo de vida das chaves criptográficas foi desenvolvido neste seio. O hardware, desenhado pelo GT ICPEDU II, foi projetado pela empresa brasileira Kryptus[3] em parceria com a RNP.

1.2 Contextualização das Contribuições

A implantação de ACs e ARs no contexto de ICPs é bem estabelecida, devendo seguir políticas previamente definidas de forma a nortear todas as atividades envolvidas no seu gerenciamento. A figura 1.1 apresenta a estrutura documental normalmente adotada na implantação de uma infra-estrutura de chaves públicas.

As políticas são descritas na forma de documentos, seguindo preceitos regidos por normas, padrões e recomendações nacionais e internacionais. Parte desses

[3]http://www.kryptus.com.br

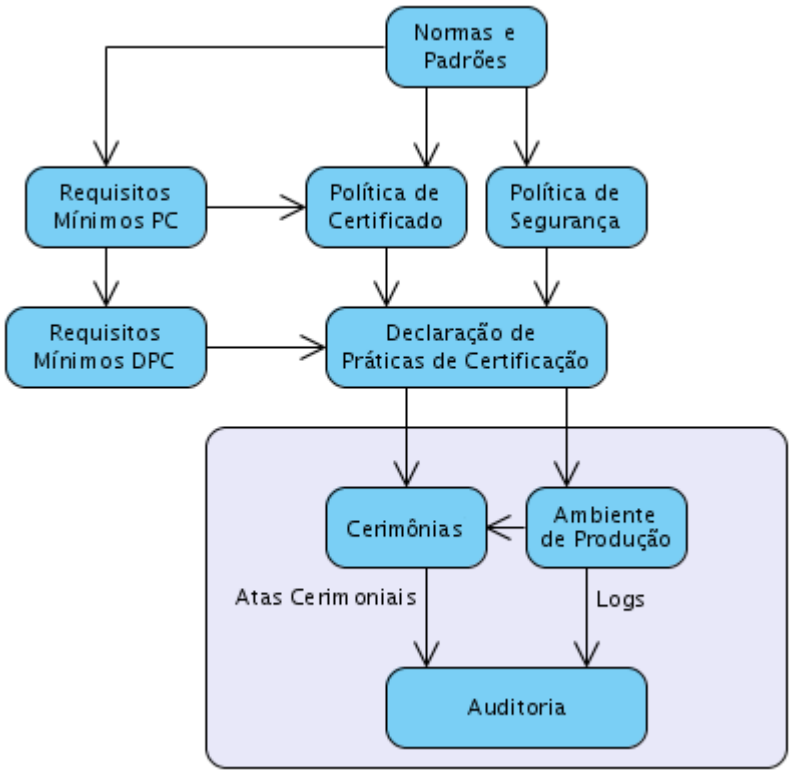

Figura 1.1: Contextualiza a área de atuação do trabalho

documentos orientam como um certificado deve ser emitido para ser aplicável um uma determinada situação, respeitando os requisitos de uma política de segurança, na forma de Políticas de Certificado (PC). Outra parte, o conjunto de requisitos que promovem a proteção adequada dos ativos dos sistemas de informação das entidades da ICP, constitue a política de segurança (PS).

A declaração de práticas de certificação (DPC), por sua vez, descreve como a PC é implementada, levando em consideração as boas práticas de segurança estabelecidas na PS, englobando os requisitos da organização que hospeda a ICP.

Como forma de facilitar a escrita destes documentos, o comitê gestor da ICP define requisitos mínimos das PC e DPC para todas as ACs e ARs existentes na árvore da ICP. Desta forma, todas a entidades participantes da ICP podem facilmente seguir a mesma orientação.

Uma vez consolidado o documento DPC, pode-se implementar o ambiente de produção e elaborar as cerimôniais. O ambiente de produção consiste de um ambiente seguro com sistemas de controle de acesso, ambiente de registro, entre outros. Esses controlam o acesso ao material criptográfico, gerando evidências de seu uso através de registros de evidências (logs). As cerimônias são normalmente escritas na forma de roteiro em ordem cronológica de execução, para todas as atividades previstas no ambiente de produção. O produto das cerimônias são as atas cerimoniais. A partir das atas e registros de atividades dos componentes envolvidos são realizadas as auditorias, que produzirão relatórios, demonstrando a confiabilidade de toda a estrutura para a parte confiante.

Este trabalho contribui no contexto de implementação de ICPs nas áreas de Cerimônias (detalhando todos os passos realizados no gerenciamento do ciclo de vida das chaves privadas), Ambiente de Produção (ASI-HSM com o OpenHSM) e Auditoria (registro de todas as etapas realizadas no interior do HSM com a possibilidade de exportação), conforme a área em destaque na figura 1.1.

Em termos de ambiente de produção, destacam-se os protocolos que são propostos para o estrito controle do ciclo de vida das chaves criptográficas, mesmo considerando várias cópias operacionais destas chaves, através do uso de grupos de operadores (custodiantes de material sensível), grupos de administradores e grupos totalmente independentes de auditores. Todos os passos executados dentro do HSM são registrados em arquivos de registro de auditoria e exportados com a autorização dos auditores.

Os passos de todos os sub-protocolos do HSM são mapeados dentro das cerimônias, detalhando principalmente as atividades diretamente relacionadas ao gerenciamento do ciclo de vida das chaves criptográficas de uma ICP. Esta associação mostra perfeita sintonia entre o roteiro cerimonial e o registro de sua execução nos registros de atividades do HSM. Garante-se com isso maior confiança das testemunhas presentes na cerimônia quanto a corretude do uso das chaves criptográficas.

Por último, sentem-se os auditores capazes de produzir suas análises e relatórios com a certeza de mostrar que tudo foi feito respeitando as políticas preconizadas.

As contribuições deste trabalho são de caráter técnico e teórico-científico. Quando se expõe a contribuição técnica procura-se organizar o material já existente na literatura técnica e documentação das soluções existentes de forma a elucidar e dirigir tal escrito às necessidades dos desenvolvedores de componentes destinados a ICP. Um forte exemplo deste tipo de contribuição é a apresentação didática para desenvolvimento de Engines – padrão utilizado para a comunicação entre aplicações e provedores de serviços criptográficos em ambientes de código aberto suportados pela plataforma OpenSSL.

Por outro lado, este trabalho apresenta diversas contribuições teórico-científicas. A principal delas esta relacionada ao protocolo que permite a gestão confiável de várias instâncias simultâneas do material criptográfico em HSM. Tal protocolo está implementado no HSM do projeto ICP-EDU II, ASI-HSM.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem por objetivo geral descrever, e propor modificações onde necessário, nos sub-protocolos criptográficos de gestão do ciclo de vida de chaves criptográficas que compõem o OpenHSM, com ênfase nos esquemas de cópia e recuperação de chaves e a instanciação de ambientes de produção com cópias das chaves gerenciadas, respeitando as principais recomendações nacionais e internacionais relacionadas à implementação de hardware criptográficos.

1.3.2 Objetivos Específicos

Visando alcançar o objetivo geral, foram definidos alguns objetivos específicos, tais como:

- apresentar os principais dispositivos criptográficos utilizados no contexto de uma infra-estrutura de chaves públicas;

- levantar as normas e recomendações nacionais e internacionais que regem o desenvolvimento de dispositivos criptográficos;
- descrever o OpenSSL e suas funcionalidades, com foco na implementação de engines para integração de hardware criptográficos;
- aprofundar o entendimento dos mecanismos de criação e recuperação de cópias de segurança do OpenHSM, capazes de proporcionar o uso simultâneo de várias cópias das chaves gerenciadas em ambientes paralelos;
- gerar, quando inexistente, documentação sobre os recursos tecnológicos utilizados no produtação do OpenHSM, na forma do ASI-HSM.

1.4 Justificativa e Motivação

A certificação digital tem se tornado uma ferramenta essencial à modernização das instituições públicas e privadas brasileiras. Vê-se o uso desta tecnologia na emissão de notas fiscais eletrônicas, nos sistemas bancários e na gestão de documentos eletrônicos, para citar alguns exemplos.

Um dos elementos chaves desse sucesso é o investimento que o Estado Brasileiro tem feito para o total domínio desta tecnologia no Brasil. O uso acadêmico de certificação digital nas instituições de pesquisa e ensino nacionais formarão os recursos humanos necessários para garantir a adoção em larga escala dos certificados digitais nas mais diversas situações, sem ficarmos reféns de corporações que estão unicamente subordinadas ao arcabouço jurídico de outros países.

Em especial, o domínio no desenvolvimento de dispositivos de hardware irá permitir, não só um maior uso destes nas mais diversas aplicações, como um substancial barateamento da tecnologia. Isso sem falar que tais dispositivos sofrem de controle rigoroso nos países que dominam sua tecnologia, e não são poucos os casos em que a tecnologia é vista como de segurança nacional, impondo restrições a sua exportação.

Foi neste cenário que a RNP criou e mantém um grupo de pesquisadores, de várias universidades e instituições de pesquisa, com o objetivo de implantar uma

infra-estrutura de chaves públicas para pesquisa e ensino.

O HSM, resultado desse esforço, foi concebido para prover aos integrantes da ICPEDU e parceiros de um dispositivo de baixo custo que permitisse a gestão segura de chaves criptográficas.

1.5 Trabalhos Correlatos

Um dos primeiros trabalhos que se tem notícia sobre a proteção de chaves criptográficas no contexto de uma AC é devido a Jeff Schiller[2]. Seu trabalho propõe duas abordagens diferentes para a gestão do ciclo de vida da chave, dependendo de quem tem acesso ao material criptográfico. A primeira, quando o ser humano tem acesso direto as chaves e a segunda, quando este acesso direto não é possível. O trabalho, de caráter mais geral, discute as diferentes preocupações quanto ao controle do ciclo de vida das chaves nestes dois contextos.

Os primeiros estudos sobre o gerenciamento de chaves criptográficas no Laboratório de Segurança em Computação (LabSEC) da Universidade Federal de Santa Catarina foram feitos em 2003. Assim que o GT ICPEDU II foi aprovado pela RNP, iniciou-se o desenvolvimento de um protótipo do HSM. Como resultado acadêmico do GT teve-se dois trabalhos: um trabalho conclusão de curso (TCC) de Graduação em Sistemas de Informação e uma dissertação de mestrado no Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Catarina (UFSC).

No início dos trabalhos foi feita uma revisão da literatura sobre esta área em particular e obteve-se somente referências a produtos comerciais de empresas fabricantes de HSM. Muito pouco se sabe sobre os protocolos que esses produtos implementam, provavelmente devido a proteção de propriedade intelectual e ao segredo industrial.

Souza[3] tratou em seu trabalho o gerenciamento de chaves criptográficas em hardwares embarcados. Martina[4] desenvolveu sua dissertação de mestrado estudando e propondo modificações nos dispositivos de hardware criptográficos de forma a atender melhor as necessidades funcionais de uma autoridade certificadora.

Em 2007, Martina[5] apresentou os protocolos de gestão dos grupos de

administração, operação e auditoria de chaves criptográficas no OpenHSM. E em 2008, Souza[6] publicou um trabalho que aprimorava os protocolos de cópia e recuperação do material criptográfico no OpenHSM.

1.6 Organização deste Trabalho

O Capítulo 2 apresenta os dispositivos mais utilizados para armazenamento de chaves criptográficas. Nesse, são apresentados os smartcards, com ênfase no modelo utilizado no OpenHSM, tokens criptográficos e HSM disponíveis no mercado.

No capítulo 3, as normas nacionais e internacionais de certificação com maior relevância são analisadas. É apresentada para cada norma, seus níveis de certificação, áreas de atuação e requisitos definidos para cada área.

A seguir, no capítulo 4, é apresentado o OpenSSL, poderosa ferramenta criptográfica, na qual o ASI-HSM está baseado. O capítulo descreve a aplicação de linha de comando, suporte a engines para integração de módulos criptográficos e a versão simplificada da biblioteca com certificação FIPS 140-2 nível 1. Justifica-se a inclusão deste material neste texto uma vez que a documentação original é desatualizada e dispersa.

O capítulo 5 apresenta a solução como um todo do HSM, na forma do ASI-HSM, incluindo sua arquitetura, interfaces de gerenciamento e engine para utilização de seus serviços criptográficos.

Na seqüência, o capítulo 6 aborda o protocolo OpenHSM para gerenciamento do ciclo de vida das chaves criptográficas, incluindo os sub-protocolos para criação e uso de chaves criptográficas gerenciadas.

O capítulo 7 descreve todos os sub-protocolos do OpenHSM responsáveis pela criação e recuperação de cópias de segurança do HSM, de forma a garantir a continuidade do ciclo de vida das chaves criptográficas gerenciadas mesmo em caso de falhas de hardware ou desastres.

Finalmente, no capítulo 8, conclui-se o trabalho, apresentando suas contribuições e trabalhos futuros.

Capítulo 2

Hardware Criptográfico

2.1 Introdução

Um hardware criptográfico é um dispositivo que visa atender aos requisitos de segurança encontrados em aplicações que usam serviços criptográficos onde não é possível segregar o material sensível unicamente dentro da estação hospedeira, que embarca a aplicação, ou em situações onde é necessário um alto volume de processamento criptográfico. Um HSM, por exemplo, é tipicamente de 10 a 100 vezes mais rápido que um computador pessoal para as finalidades especializadas.

Algumas aplicações, como servidores seguros de páginas Web, requerem grande capacidade de processamento criptográfico, que pode ser alcançado com o emprego de um HSM que é usado como um co-processador para operações criptográficas, liberando unidade de processamento dos servidores para outras finalidades.

Este capítulo apresenta os dois principais tipos de hardware criptográficos, sendo na seção 2.2 apresentado os smartcards, com especial atenção aos requisitos referentes ao modelo de smartcards utilizados no HSM alvo do OpenHSM. A Seção 2.3 apresenta uma visão geral dos HSMs, hardware que permite um controle rigoroso do ciclo de vida de chaves criptográficas, sendo um dispositivo fundamental para a gestão da chave privada de uma autoridade certificadora raiz de uma ICP. Adicionalmente, alguns HSMs comerciais são destacados.

2.2 Smartcards

Primeiramente empregados por empresas de telefonia da França e Alemanha nos anos 80, os smartcards ganharam mercado com o desenvolvimento da criptografia moderna e evolução da tecnologia de semicondutores, sendo considerados uma geração mais nova, inteligente e segura dos cartões de identificação. Os celulares com tecnologia GSM e cartões de créditos incorporam smartcards, visando principalmente sua efetividade no combate a fraudes[7].

Smartcards contêm um circuito integrado embarcado, capaz de receber, transmitir, armazenar e processar dados. A comunicação pode ser feita com ou sem contato físico, sendo no último caso utilizadas ondas de rádio.

Uma das grandes características dos smartcards é que os dados nele armazenados podem ser protegidos contra acesso e manipulação não autorizados. Seu sistema operacional estabelece uma lógica de segurança que controla a entrada e saída de comandos e dados através de sua interface de comunicação serial. É possível, por exemplo, gerar um par de chaves assimétricas em um smartcard e configurá-lo de tal forma que a chave privada nunca poderá ser exportada. Neste caso, as operações criptográficas para utilizar esta chave podem ser realizadas somente dentro do mesmo.

O controle de acesso ao smartcard é realizado com a utilização de senha ou controle biométrico. A senha é conhecida como número de identificação pessoal (PIN)[1], que deve ser conhecida somente pelo dono do smartcard, e consiste de uma seqüencia numérica normalmente entre 4 e 6 dígitos. Os smartcards podem implementar um modelo de PIN fixo ou modificável. O modelo de PIN fixo não permite que seus usuários alterem o PIN, obrigando sua memorização, enquanto o modelo de PIN modificável atende as necessidades de seus usuários, permitindo a alteração do PIN.

Os smartcards, por questão de segurança, possuem um controle do número de tentativas consecutivas sem êxito na liberação do acesso, bloqueando o PIN quando digitado mais de três vezes incorretamente. Este comportamento evita o ataque de tentativa e erro, também chamada de ataque de força bruta. Em caso de bloqueio, o smart-

[1] do inglês Personal Identification Number

card precisa ser submetido a um processo de desbloqueio, utilizando uma senha chamada de chave de desbloqueio pessoal (PUK)[2], normalmente entre 6 e 8 dígitos numéricos. Ela reinicia o contador de tentativas sem sucesso e permite a configuração de um novo PIN. O PUK, além de não ser alterável, também é bloqueado se digitado incorretamente 10 vezes, inutilizando o smartcard.

O uso de smartcards se dá através de leitores especialmente desenvolvidos para este fim. Eles são o meio pelo qual um computador pode interagir com seu conteúdo, provendo energia para o sistema operacional.

Os conceitos de smartcard também são empregados em tokens criptográficos, no formato de dispositivos USB[8], não necessitando de leitora específica para sua utilização.

2.2.1 Smartcards para HSMs

O objetivo do emprego de smartcards no projeto piloto do HSM é a segura identificação dos membros de seus grupos. Cada membro possui um smartcard, no qual será armazenado um par de chaves criptográficas RSA[9] e um certificado. A utilização deste smartcard com o PIN correto garante a autenticação do membro.

Apesar de existirem inúmeros modelos comerciais de smartcards atualmente no mercado, com diferentes características e capacidades, o GT ICP-EDU II definiu o seguinte conjunto de requisitos na escolha do smartcard apropriado:

- ser utilizável a partir de contato físico;
- suportar a geração de chaves assimétricas RSA[9] de do mínimo 1024 bits;
- permitir a importação de pares de chaves RSA gerados fora do smartcard;
- suportar criptografia de dados com chaves assimétricas RSA de até de 1024 bits;
- ter capacidade de geração e verificação de assinaturas digitais utilizando chaves RSA;

[2] do inglês Personal Unblocking key

- suportar armazenamento de certificados X.509v3[10];
- ser compatível com a norma ISO 7816 1/2/3/4[11];
- implementar uma estrutura de objetos e arquivos compatível com PKCS#15[12];
- possuir no mínimo 32 kilobytes de capacidade de armazenamento.

2.3 Módulo de Segurança Criptográfica (HSM)

Módulos de segurança criptográfica são empregados em soluções onde o controle do ciclo de vida de suas chaves criptográficas é crucial para seu funcionamento. Esses módulos devem permitir que as chaves criptográficas sejam criadas, armazenadas, utilizadas e apagadas internamente, sem nunca serem visíveis ao mundo externo.

Os HSMs possuem mais capacidade de processamento que os smartcards, servindo na sua maioria como aceleradores criptográficos, sendo capazes de gerenciar o ciclo de vida de várias chaves ao mesmo tempo.

2.3.1 Ciclo de vida de chaves criptográficas

O ciclo de vida de uma chave criptográfica consiste da seqüencia de estados por ela assumida, desde sua geração até a sua destruição[13], com o módulo criptográfico garantindo seu correto manuseio em cada estado assumido.

A continuidade do ciclo de vida das chaves deve ser possível mesmo em caso de falha de hardware ou acontecimento de um desastre, através da utilização de mecanismos de criação e recuperação de cópias de segurança. Entretanto, no caso de destruição de uma chave, deve-se ter garantia que todas suas instâncias foram destruídas, inclusive as cópias de segurança.

A seguir, apresenta-se uma lista dos principais estados que uma chave criptográfica pode assumir durante seu ciclo de vida em um HSM:

- geração: o ponto inicial do ciclo de vida de uma chave criptográfica;

- armazenamento: a chave está armazenada de forma segura no interior do perímetro criptográfico;
- uso: utilização da chave para realização de operações criptográficas;
- backup: referente as cópias de segurança da chave, que deve estar armazenado em lugar independente, possibilitando a recuperação em caso de falha de hardware ou desastre;
- recuperação: refere-se a recuperação de uma cópia de segurança da chave, utilizado quando a chave foi perdida de alguma forma, sem seu comprometimento;
- destruição: a chave não é mais necessária, finalizando seu ciclo de vida.

Os estados ainda podem ser categorizados quanto a disponibilidade de uso de uma chave criptográfica[14], como é descrito na tabela 2.1.

Tabela 2.1: Categorias dos estados do ciclo de vida de chaves criptográficas quanto a sua disponibilidade de uso.

Categoria	**Descrição**
pré-operacional	período desde a criação do par de chaves até sua liberação para realização de operações criptográficas;
operacional	a chave está disponível para uso
pós-operacional	a chave não pode ser mais utilizada ativamente para realização de operações criptográficas. Sua utilidade é prover garantia das operações realizadas no passado.
obsoleta	todas as instâncias da chave criptográfica são apagadas

2.3.2 Módulos de Segurança Criptográfica de Mercado

Existem diversos fabricantes de HSMs no mundo e alguns de seus produtos são descritos nas tabelas 2.2, 2.3 e 2.4, salientando suas características e capacidades. Os principais fabricantes são: nCipher, SafeNet e AEP Networks.

O nShield F3 2000 da nCipher, além das características apresentadas na tabela 2.2, suporta execução segura de código, o que permite ao seu usuário importar um

trecho de código para ser executado dentro do HSM, considerado um ambiente seguro. Além disso, conta com relógio interno de alta precisão, permitindo sua utilização em sistemas de carimbo do tempo.

Tabela 2.2: Características do HSM nShield F3 2000 da nCipher

Item	**Características**
Certificação	FIPS 140-2 nível 3
Capacidade de Processamento	2000 assinaturas RSA de 1024 bits por segundo
Conectividade	Interface PCI
API integração	OpenSSL, PKCS#11, Microsoft CryptoAPI
Algoritmos criptográficos (Assimétricos)	DSA, ECDSA, RSA
Algoritmos criptográficos (Simétricos)	AES, Triple-DES
Algoritmos criptográficos (Funções de Resumo)	Triple-DES MAC, SHS, HMAC
Backup	Proprietário e para HSMs do mesmo fabricante
Uso	Propósito geral

Da mesma forma que o HSM da nCipher, o Luna PCI 3000 da SafeNet apresenta características adicionais ao gerenciamento do ciclo de vida de chaves criptográficas, como a utilização de um teclado diretamente conectado ao HSM para autenticar seus usuários, provendo uma caminho confiável para a digitação do PIN. Suas características gerais podem ser vistas na tabela 2.3.

O HSM Keyper PCI da AEP Networks possui uma funcionalidade interessante para ambientes que demandem alto poder de processamento, suportando balanceamento de carga entre vários HSMs. Em caso de uma falha em um dos HSMs, os restantes automaticamente assumem o controle. As características gerais do KeyPer PCI podem ser encontradas na tabela 2.4.

2.4 Conclusão

A utilização de hardwares criptográficos é crucial em ambientes com algum tipo de valor agregado, já que, idealmente, a segurança dos algoritmos criptográficos

Tabela 2.3: Características do HSM Luna PCI 3000 da SafeNet

Item	**Características**
Certificação	FIPS 140-2 nível 2 e 3
Capacidade de Processamento	3000 assinaturas RSA de 1024 bits por segundo
Conectividade	Interface PCI
API integração	OpenSSL, PKCS#11, Microsoft CryptoAPI 2.0, Java JCA/JCE
Algoritmos criptográficos (Assimétricos)	RSA, DSA, Diffie Hellman
Algoritmos criptográficos (Simétricos)	DES/3DES, AES, RC2, RC4, RC5
Algoritmos criptográficos (Funções de Resumo)	SHA-1, SHA-256, SHA-384, SHA-512, MD2, MD5
Backup	Proprietário e para HSMs do mesmo fabricante
Uso	Propósito geral

está diretamente relacionada ao correto manuseio de suas chaves.

Os dois principais exemplares de hardware criptográficos são o smartcard e o HSM. O smartcard tem o formato de uma cartão e é utilizado para proteger a chave privada de pessoas ou de sistemas, onde não há a necessidade de uma taxa elevada de processamento criptográfico. Já os HSMs, como visto neste capítulo, tem um controle maior do ciclo de vida de chaves criptográficas e alto poder de processamento.

Pode-se observar que os HSMs de mercado são normalmente destinados a aplicações gerais e não são projetados para a instanciação de autoridades certificadoras.

Uma das principais deficiências dos HSMs comerciais é a falta de um padrão aceitável para realizar a cópia e a restauração do seu material crítico, tal como as chaves criptográficas. Os fabricantes alegam que possuem tal funcionalidade, mas esta é limitada a produtos do próprio fabricante. Alegam que isso é necessário para manter a proteção das chaves. Acontece que num ambiente de produção, por exemplo, uma AC Raiz, normalmente a chave é gerada e deve ser mantida no módulo por período de tempo superior a 10 anos. E os produtos comerciais são descontinuados antes de vencer a validade da chave.

Como será visto nos próximos capítulos, esta dissertação de mestrado trata desta funcionalidade com especial atenção. Ou seja, como podem ser criadas cópias

Tabela 2.4: Características do HSM Keyper PCI da AEP Networks

Item	**Características**
Certificação	FIPS 140-1 nível 3
Capacidade de Processamento	não informada
Conectividade	Interface PCI mas utilizável através da rede
API integração	OpenSSL, PKCS#11, Microsoft CryptoAPI 2.0, Java JCA/JCE
Algoritmos criptográficos (Assimétricos)	RSA, DSA, Diffie Hellman
Algoritmos criptográficos (Simétricos)	DES/3DES
Algoritmos criptográficos (Funções de Resumo)	SHA-1, MD5
Backup	Proprietário e para HSMs do mesmo fabricante
Uso	Propósito geral

de segurança, que permitem posterior recuperação em dispositivos adicionais, sem perder a rastreabilidade de todas as instâncias das chaves criptográficas em operação.

Capítulo 3

Normas

3.1 Introdução

A falta de interoperabilidade entre hardware criptográficos torna sua utilização custosa, pois cada vez que é necessário mudar um hardware, deve-se adaptar a aplicação que o utiliza. Neste sentido, governos e empresas se uniram e propuseram padrões de interoperabilidade que devem ser seguidos no projeto e fabricação de tais dispositivos.

Desta forma, este capítulo apresenta as principais normas nacionais e internacionais que devem ser consideradas no contexto de hardware criptográficos. Dentre eles estão a FIPS PUB 140-2 (seção 3.2) e o Manual de Condutas Técnicas 7 do LEA-ITI (seção 3.3). Ambas estabelecem requisitos na construção de HSMs.

Os critérios comuns[15], que podem ser aplicados na construção de qualquer dispositivo eletrônico, também pode ser utilizado na construção de HSMs. Entretanto, por ser mais genérico, não será abordado aqui, permanecendo o foco nas normas FIPS PUB 140-2 e no MCT-7.

3.2 FIPS PUB 140

A norma FIPS PUB 140-2[16], publicada pelo Instituto Nacional de Padrões e Tecnologia[1] do governo norte-americano, define um conjunto de requisitos de segurança para hardware criptográficos empregados na proteção de informações sensíveis em sistemas computacionais e de telecomunicação. Esta norma é a evolução da FIPS PUB 140-1[17].

Os requisitos de segurança cobrem diversas áreas relacionadas com o projeto e implementação de um módulo criptográfico, devendo o módulo satisfazer todos os requisitos do nível de certificação desejado. A FIPS 140-2 prevê quatro níveis: 1, 2, 3 ou 4. Nos níveis em que os requisitos se diferem, aplica-se o comportamento acumulativo, ou seja, as certificações em níveis mais altos (4) devem também satisfazer os requisitos dos níveis mais baixos (1).

Em um processo de avaliação de um módulo criptográfico, atribui-se o nível de cada área analisada individualmente. Posteriormente, a área com o nível mais baixo estabelecerá o nível de certificação do módulo criptográfico.

A FIPS 140-2 utiliza o termo ”parâmetros críticos de segurança (CSP)”para informações que, se divulgadas ou modificadas, podem comprometer a segurança do módulo criptográfico, tais como segredos e chaves privadas criptográficas e dados de autenticação (PINs e senhas).

A norma ainda inclui outros quatro documentos, complementando o documento principal, chamados de anexo A, B, C e D, respectivamente a família de algoritmos criptográficos aprovados, perfis de proteção, geradores de números aleatórios aprovados e técnicas de estabelecimento de chaves[2] simétricas e assimétricas.

As sub-seções seguintes sumarizam cada uma das áreas cobertas pela norma, com uma ênfase nas principais diferenças entre os níveis de segurança.

[1] do inglês National Institute of Standards and Technology - NIST
[2] do inglês key establishment

3.2.1 Especificação do Módulo Criptográfico

Esta área contém as definições de módulo e fronteira criptográficos que devem ser utilizadas na norma como um todo, além de sumarizar uma lista com toda a documentação de hardware, software e firmware, diretamente relacionada à segurança do módulo, que é necessária no processo de certificação. Entre os documentos requeridos estão:

- especificação física do módulo;
- portas físicas e interfaces lógicas empregadas;
- especificação dos controles físicos e lógicos do módulo;
- lista de todas as funções de segurança, aprovadas ou não, empregadas pelo módulo, especificando todos os modos de operação suportados, aprovados ou não;

Adicionalmente, requer-se que os módulos criptográficos possuam uma política de segurança[3], contendo regras derivadas dos requisitos da norma FIPS 140-2 e quaisquer outras regras derivadas dos requisitos impostos pelo fabricante do módulo.

Por fim, define-se que a documentação deve conter um diagrama de blocos, detalhando os principais componentes de hardware e suas interconexões, tais como microprocessadores, buffers e memórias. Além disso, é necessário prover documentação do projeto dos componentes de hardware, software e firmware.

3.2.2 Portas Físicas e Interfaces Lógicas

A FIPS 140-2 define que o fluxo de dados e acesso físico de um módulo criptográfico devem estar limitados as suas portas físicas e interfaces lógicas, identificando assim, todos os pontos de entrada e saída existentes.

As interfaces lógicas são divididas em quatro tipos. Apesar de cada tipo poder compartilhar da mesma porta física, elas devem ser logicamente independentes umas das outras. Os tipos de interfaces são:

[3] do inglês security policy

- entrada de dados: todos os dados que são direcionados ao módulo criptográfico para serem processados devem utilizar este tipo de interface;
- saída de dados: todos os dados, com exceção de dados de estado, devem sair do módulo através deste tipo de interface;
- entrada de controle: os comandos, sinais e dados de controle para operacionalização do módulo criptográfico devem utilizar este tipo de interface;
- saída de estado: todos os dados utilizados para indicar estado do módulo, incluindo leds e displays, devem utilizar este tipo de interface.

Os níveis 1 e 2 somente requerem que exista uma interface lógica de cada um dos tipos descritos anteriormente. Nos níveis 3 e 4, além do requisito anterior, a norma requer que a porta de entrada e saída de CSP do módulo criptográfico - parâmetros de chaves criptográficas em texto claro e dados de autenticação - sejam fisicamente separadas de outras portas ou utilizem uma interface lógica de caminho confiável[4]. Adicionalmente, a entrada dos dados deve ser de forma direta, através de um cabo diretamente conectado ao módulo, ou também, com a utilização de um caminho confiável de dados. O uso de caminho confiável visa cumprir os dois requisitos dos níveis 3 e 4.

3.2.3 Papéis, Serviços e Mecanismos de Autenticação

O módulo criptográfico deve suportar papéis de usuários, onde cada papel é associado a um conjunto definido de serviços, porém, é opcional para o módulo, o emprego ou não de mecanismos para autenticação dos mesmos. Os serviços que não modifiquem, substituam ou revelam chaves criptográficas e CSP do interior do módulo, podem ser executados sem que os usuários sejam autenticados.

Entre os papéis de usuários que devem ser suportados pelo módulo estão:

- operador: executa serviços gerais de segurança, incluindo operações criptográficas;

[4]do inglês trusted path.

- administrador (crypto officer): responsável pela inicialização e gerenciamento do módulo criptográfico, incluindo a geração de chaves criptográficas e auditoria;

- manutenção: papel opcional, utilizado durante reparos físicos ou lógicos no módulo criptográfico. Todos os segredos e chave privadas em aberto e CSPs desprotegidos devem ser apagados utilizando o circuito zerador[5] sempre que este papel é autenticado.

É opcional para o módulo criptográfico a definição de papéis ou subpapéis adicionais aos citados acima.

Existem alguns serviços que o módulo deve obrigatoriamente prover, tais como mostrar o estado do módulo, executar auto-testes e suportar a execução de pelo menos uma operação criptográfica em um algoritmo aprovado. Além destes, muitos outros serviços podem existir, até operações criptográficas de algoritmos não aprovados, operando em modo também não aprovado.

Os mecanismos de autenticação de um módulo criptográfico, quando implementado, devem ser classificados em uma das seguintes categorias:

- baseado em papel: a autenticação é realizada no papel selecionado (implícita ou explicitamente). O módulo não autentica a identidade individual do usuário.

- baseado em identidade: o usuário é individualmente identificado e verifica-se se ele pode assumir o papel selecionado. O papel pode ser selecionado direta ou indiretamente.

É opcional para o módulo a possibilidade de executar vários serviços uma vez que um papel está autenticado ou solicitar uma autenticação para cada serviço. Entretanto, os resultados das autenticações depois de desligar e ligar o módulo não podem ser armazenados, necessitando re-autenticação de qualquer um dos papéis existentes.

O nível 1 da norma FIPS 140-2 não requer nenhum mecanismo de autenticação dos operadores. Já o nível 2, requer a autenticação baseada em papel, e os níveis 3 e 4 baseada em identidade.

[5]do inglês zeroized

3.2.4 Modelo de estados finitos

Todos os níveis de certificação da FIPS 140-2 requerem que a operação do módulo criptográfico seja especificada em um modelo de estados finitos, representado por um diagrama ou tabela de transição de estados, incluindo:

- todos os estados operacionais e de erro;
- as transições de um estado para outro;
- os eventos de entrada que causam uma transição;
- os eventos de saída após a transição para outro estado.

Alguns exemplos de estados operacionais obrigatórios são a execução dos auto-testes, autenticação do papel administrador, estado de manutenção, entre outros.

3.2.5 Segurança Física

O FIPS 140-2 define que um módulo criptográfico deve empregar mecanismos de segurança para restringir acesso físico não autorizado ao conteúdo do módulo, prevenindo sua utilização e modificação.

O nível 1 de certificação não possui nenhum requisito para módulos criptográficos de chip único, como um smartcard, porém, define que o módulo deve estar embalado por um invólucro de metal ou plástico duro, podendo incluir portas ou capas removíveis.

O nível 2 requer a utilização de travas e a evidência de qualquer tentativa de invasão do perímetro criptográfico do módulo. No nível 3, além de ficar evidente a tentativa de invasão, deve existir alta probabilidade de ocorrer danos irreparáveis ao módulo, por exemplo, envolvendo o módulo em uma pasta para criação de um bloco único, rígido e opaco a luminosidade.

O último nível, requer ainda que o módulo responda ativamente a tentativa de invasão, ativando o circuito zerador que apaga segredos, chaves privadas e PCSs em texto claro.

3.2.6 Ambiente Operacional

O ambiente operacional de um módulo criptográfico se refere ao gerenciamento de software, firmware e hardware necessários para sua operação, sendo o sistema operacional (SO) um dos principais componentes. O ambiente operacional de um módulo deve se enquadrar em uma das seguintes categorias:

- ambiente operacional de propósito geral: se refere ao uso de um sistema operacional de propósito geral e comercialmente disponível.
- ambiente operacional limitado: ambiente operacional estático não modificável, que não requer um sistema de propósito geral para seu suporte.
- ambiente operacional modificável: se refere a ambiente operacional que pode ser reconfigurado, adicionando, deletando ou modificando funcionalidades. Sistemas operacionais são considerados ambientes operacionais modificáveis se componentes de software/firmware, que não foram incluídos no processo de certificação, podem ser importados e executados no módulo por seus usuários.

Os requisitos desta área se aplicam apenas para ambientes operacionais modificáveis, tendo abrangências diferentes para cada nível de certificação.

O nível básico requer que apenas um usuário possa utilizar o módulo criptográfico por vez, prevenindo acesso de outros processos a segredos, chaves privadas e CSPs armazenados em aberto, sendo que o requisito anterior é aplicável somente para o nível 1 da certificação. O software/firmware do módulo deve estar instalado de uma tal forma que bloqueie modificações ou acesso de usuários não autorizados, utilizando alguma técnica de integridade aprovada pela família de algoritmos FIPS.

Os níveis 2, 3 e 4, requerem, respectivamente, um sistema operacional de nível EAL2, EAL3 e EAL4 do Critérios Comuns[15], ou uma avaliação equivalente. Um mecanismo de auditoria é requisito do nível 2, incluindo uma lista de eventos que precisam obrigatoriamente ser registrados. O nível 3 inclui outros eventos e requer a utilização de caminho confiável[6] e modelo de política de segurança[7].

[6] do inglês trusted path (FTP_TRP.1 dos critérios comuns).
[7] do inglês Security Policy Model (ADV_SPM.1 do critério comum).

3.2.7 Gerenciamento de Chaves Criptográficas

Esta área apresenta requisitos de segurança que visam proteger o gerenciamento de todo o ciclo de vida de chaves criptográficas, componentes de chaves criptográficas e CSPs empregados pelo módulo.

Os requisitos são agrupados em várias atividades do processo de gerenciamento de chaves criptográficas, tais como:

- geração de números aleatórios: o gerador utilizado para geração de chaves criptográficas deve obrigatoriamente ser aprovado pela família de padrões FIPS. Entretanto, se o módulo criptográfico emprega geradores não aprovados, os dados gerados devem apenas ser utilizados para gerar sementes para um gerador aprovado ou para gerar vetores de inicialização para algoritmos simétricos;

- geração de chaves criptográficas: se o módulo suportar essa funcionalidade, os métodos de geração devem ser aprovados pela família de padrões FIPS [16, anexo A];

- estabelecimento de chaves: o processo de estabelecer uma chave simétrica para a comunicação de duas ou mais partes, podendo ser realizado através de método automático, manual ou uma combinação de etapas manuais e automáticas;

- importação e exportação de chaves criptográficas: o módulo pode suportar importação e/ou exportação de chaves, que podem ser realizados de forma manual, através do teclado, ou automática, utilizando smartcards;

- armazenamento de chaves criptográficas: as chaves criptográficas podem ser armazenadas cifradas ou em claro. Segredo e chaves armazenados em claro não podem ser acessíveis a usuários não autorizados;

- circuito zerador de chaves criptográficas: o módulo deve prover um método para apagar segredos, chaves privadas e CSPs armazenados em claro dentro do módulo.

Além disso, define-se que chaves criptográficas e CSPs cifrados com algoritmos não aprovados são considerados em formato de texto claro.

3.2.8 Interferência e Compatibilidade Eletromagnética

O módulo criptográfico deve estar em conformidade em relação a interferência[8] e compatibilidade[9] eletromagnética definida pela comissão de comunicação federal do governo americano (FCC)[18]. Os níveis 1 e 2 da certificação devem seguir os requisitos para uso comercial, enquanto os níveis 3, 4 para uso doméstico.

3.2.9 Auto-testes

O módulo criptográfico deve realizar auto-testes que visam garantir seu funcionamento apropriado, devendo entrar em um estado de erro se qualquer auto-teste falhar. O módulo não poderá realizar nenhuma operação criptográfica enquanto estiver neste estado, devendo a interface de saída de dados ser fechada. Existem duas categorias de auto-teste: de energização e condicional.

Os auto-testes de energização são automaticamente realizados quando o módulo é energizado, sem intervenção dos usuários. Entre os testes necessários estão: testes de algoritmos criptográficos, testes de integridade de software e firmware, testes de funções críticas para o funcionamento seguro do módulo. Define-se que todos os algoritmos criptográficos devem ser submetidos a testes de resposta conhecida.

Os auto-testes condicionais são realizados quando determinadas operações ocorrerem no módulo:

- geração de par de chaves: testes de consistências do par;
- carregamento de software/firmware externo: testes de validação de software/firmware carregados para dentro do módulo;
- quando chaves ou componentes de chaves são entradas manualmente no módulo: mecanismos de verificação de corretude do seu conteúdo;
- utilização de um gerador de números aleatórios: executar testes contínuos para detectar geração de valores constantes.

[8] do inglês electromagnetic interference - EMI
[9] do inglês electromagnetic compatibility - EMC

- quando o módulo troca de modo de operação normal para de desvio[10]: testes para confirmar o comportamento correto de funções que operam com material criptográfico no momento de troca de modo de operação.

3.2.10 Garantia de Projeto

A garantia de projeto se refere ao uso de boas práticas do fabricante do módulo criptográfico durante seu projeto, distribuição e operação. O fabricante deve fornecer garantias que o módulo é devidamente testado, configurado, entregue, instalado e desenvolvido, contendo guias de manuseio adequado para seus usuários (administradores e operadores).

Esta área requer a utilização de gerência de configuração para todo o ciclo de desenvolvimento do módulo, incluindo a definição dos requisitos de sua entrega e operação. Além disso, para auxiliar a compreensão do funcionamento interno, requer-se um conjunto de documentos referentes a sua especificação e desenvolvimento.

Adicionalmente, um conjunto de requisitos ligados ao desenvolvimento do módulo criptográfico é definido, variando dependendo do nível de certificação. Estes requisitos visam prover garantia que a implementação do módulo corresponde com sua política de segurança e especificação funcional.

3.2.11 Mitigação de Outros Ataques

Esta área de requisitos visa abranger ataques que não eram conhecidos durante a escrita da norma FIPS 140-2 ou que ficaram de fora do escopo da mesma. Esses ataques que o módulo não é suscetível devem estar descritos na sua política de segurança.

A maioria dos ataques a um módulo criptográfico tentam determinar alguma informação referente ao material criptográfico armazenado dentro do módulo, tais como: análise da tensão utilizada durante a geração de chaves e/ou operações criptográficas, análise de tempo de processamento, indução para ocorrência de erros, entre outros.

[10] do inglês bypass

3.3 Manual de Condutas Técnicas 7 (LEA/ITI)

O manual de condutas técnicas 7 (MCT-7)[19], de responsabilidade do Instituto Nacional de Tecnologia da Informação (ITI)[20] em conjunto com os Laboratórios de Ensaios e Auditoria[11] (LEA), descreve os requisitos técnicos a serem validados no processo de homologação de um módulo de segurança criptográfica no âmbito da Infra-estrutura de Chaves Públicas Brasileira, ICP-Brasil[21].

Por HSM entende-se um servidor ou placa auxiliar criptográfica fisicamente segura, resistente à violação que fornece funcionalidades criptográficas com capacidade de geração e armazenamento de chaves simétricas e assimétricas voltados essencialmente para utilização em uma Infra-estrutura de Chaves Públicas (ICP).

O padrão MCT-7 define três Níveis de Segurança de Homologação (NSH) e dois Níveis de Segurança Física (NSF). O NSH é baseado na disponibilização do código fonte do HSM, onde o NSH 1 não requer depósito e análise do código fonte do HSM, o NSH 2 requer depósito e análise do código fonte de componentes específicos associados ao dispositivo em homologação, enquanto o NSH 3 requer o depósito e análise do código fonte completo associado ao dispositivo em homologação.

O NSF 1 requer que o módulo tenha mecanismos de segurança física que evidenciam e resistem à violação, enquanto o NSF 2, requer que os mecanismos de segurança física do HSM detectem e respondam às tentativas de violação. O fabricante do módulo, referenciado pela norma como parte interessada (PI), deve especificar o Nível de Segurança Física desejada para homologação.

O processo de homologação visa a interoperabilidade e operação segura dos serviços criptográficos providos por um HSM para uma ICP, analisando a aderência aos requisitos. O escopo da homologação não visa avaliar outros serviços presentes no HSM que não sejam úteis para uso em ICPs, desde que não exista risco desta coexistência.

O MCT-7 aborda as 11 áreas de atuação relacionadas ao projeto e implementação do módulo criptográfico utilizadas pela FIPS 140-2 (vide seção 3.2), além

[11] Entidades formalmente vinculadas ao ITI, aptas a realizar os ensaios exigidos nas avaliações de conformidade.

de incluir outros quatro temas: algoritmos criptográficos obrigatórios, gerenciamento, interoperabilidade e restrição de substâncias nocivas.

Apesar da utilização das mesmas áreas de atuação, algumas delas sofreram alterações (adição, remoção ou modificação de alguns requisitos) e serão abordados a seguir, com foco nestas diferenças.

3.3.1 Requisitos de Especificação

Esta área permanece praticamente a mesma existente no padrão FIPS 140-2, diferenciando-se apenas em um dos requisitos. Enquanto a FIPS requer que o HSM implemente no mínimo um algoritmo criptográfico aprovado, o MCT-7 apresenta uma lista de algoritmos obrigatórias, que inclui:

- DES[22] nos modos de operação ECB e CBC. Apenas para uso legado;
- Triple-DES nos modos de operação ECB e CBC;
- AES[23] nos modos de operação ECB e CBC com tamanho de chaves de 128 bits;
- Assinatura RSA[9, 24] com chaves de 1024 bits no mínimo;
- Resumos criptográficos: SHA-1[25] e SHA-256, sendo o primeiro apenas para uso legado.

Além dos obrigatórios, o MCT-7 recomenda o suporte a outros algoritmos criptográficos.

3.3.2 Portas Físicas e Interfaces Lógicas

O MCT-7 define que as portas físicas e interfaces lógicas para a entrada e saída de componentes de chaves criptográficas, dados de autenticação e PCSs devem ser fisica e logicamente separados de outras portas e interfaces do HSM, se enquadrando nos requisitos de nível 3 e 4 do FIPS 140-2.

3.3.3 Papéis, Serviços e Mecanismos de Autenticação

Esta área difere em alguns pontos com relação aos requisitos definidos na FIPS 140-2. Primeiramente, define-se que o mecanismo de autenticação deve ser baseado em papel (referente ao nível 2 da FIPS 140-2) no NSH 1 e baseado em identidade nos NSH 2 e NSH 3 (referente aos níveis 3 e 4 da FIPS 140-2).

Além disso, apesar de suportar o mesmo conjunto de papéis, somente o papel administrador é obrigatório, devendo receber a responsabilidade dos papéis não existentes. A seguir, apresenta-se o conjunto de serviços obrigatórios para cada um dos papéis:

- papel administrador: inicialização do HSM, geração de chaves RSA, sobrescrita de chaves criptográficas com zero, finalização do módulo, execução de auto-testes e requisição do estado do HSM;
- papel operador: manipulação de chaves criptográficas e PCS no HSM, acesso a algoritmos de resumo e autenticação, requisição do estado do HSM e geração de chaves RSA;
- papel de manutenção: backup e recuperação de chaves, configuração de operadores e configuração e controle dos registros do sistema (o processo de auditoria é realizado pelo papel de manutenção).

3.3.4 Modelo de Estado Finito

O modelo de estado finito segue os requisitos especificados no padrão FIPS 140-2 com a exclusão do estado de desvio, que não é aceito pelo MCT-7.

3.3.5 Segurança Física

O MCT-7 não faz distinção do número de componentes eletrônicos contidos no HSM como definido na FIPS 140-2, sendo que o modelo de segurança física deve se enquadrar em uma das seguintes categorias:

- proteção que evidencia violação: referente ao nível 2 do padrão FIPS 140-2 para módulos criptográficos $multi - chipstandalone$

- proteção que resiste a violação: referente ao nível 3 do padrão FIPS 140-2 para módulos criptográficos $multi - chipstandalone$

- proteção que detecta e responde à violação: referente ao nível 4 do padrão FIPS 140-2 para módulos criptográficos $multi - chipstandalone$

3.3.6 Ambiente Operacional

O MCT-7 classifica os ambientes operacionais nas três categorias definidas pela FIPS 140-2, utilizando os requisitos referentes ao nível 2, sem a obrigatoriedade de um sistema operacional com nível de garantia de avaliação 2 (EAL2) do Critério Comum.

3.3.7 Gerenciamento de Chaves Criptográficas

Além de herdar todos os requisitos existentes no padrão FIPS 140-2, o MCT-7 incorpora outros para se adequar as características da ICP-Brasil. Primeiramente, requer que a documentação especifique os métodos utilizados para armazenar chaves secretas, chaves privadas e PCSs que evitem divulgação, modificação e substituição não autorizada. O mesmo se aplica para chaves públicas, mas com o objetivo de proteger contra modificação e substituição não autorizadas.

O MCT-7 também requer que as chaves criptográficas geradas internamente ao HSM possam ser configuradas como exportável ou não exportável. Este requisito visa criar compatibilidade com os certificados digitais do tipo S3, S4, A3 e A4 da ICP-Brasil. Uma vez configurada como não exportável, não deve ser possível alterá-la para exportável.

3.3.8 Interferência e Compatibilidade Eletromagnética

Esta área de atuação contém os mesmo requisitos existentes na FIPS 140-2.

3.3.9 Auto-testes

O MCT-7 utiliza o mesmo conjunto de auto-testes definidos no padrão FIPS 140-2, com a restrição de que se o auto-teste de inicialização falhar, o HSM está comprometido e não pode mais ser considerado confiável.

3.3.10 Garantia de Projeto

Os requisitos de garantia de projeto são os mesmos encontrados no padrão FIPS 140-2, mudando apenas a relação de alguns dos requisitos com o nível de segurança de homologação (NSH) desejado.

O NSH 1 requer documentos guias para administradores e operadores, enquanto o NSH 2, adicionalmente, requer o código fonte de todos os componentes de software e firmware, com comentários que esclareçam a correspondência com os componentes do módulo criptográfico.

3.3.11 Mitigação de Outros Ataques

Abordado da mesma forma que o padrão FIPS 140-2.

3.3.12 Requisitos de Gerenciamento

Esta área de atuação apresenta uma lista de funcionalidades que devem estar disponíveis aos usuários do HSM, para executar operações de controle do hardware, do módulo criptográfico e das chaves gerenciadas. Cada um dos itens anteriores se ramificam em vários requisitos.

O gerenciamento do hardware deve conter procedimentos de geração de cópias de segurança e sua posterior recuperação, proteção contra falta de energia e

falhas de comunicação, atualização e validação de firmware e controle de ativação de mecanismos de segredo compartilhado.

O HSM deve dispor de uma interface gráfica com idioma em português ou inglês para seu gerenciamento, incluindo a possibilidade de importar e exportar chaves simétricas e assimétricas, apagar dados contidos no módulo (inclusive chaves criptográficas), entre outros.

3.3.13 Requisitos de Interoperabilidade

Esta área avalia o módulo do ponto de vista de interoperabilidade, visando garantir um conjunto mínimo de funcionalidades para integração com outras aplicações. O MCT-7 requer que pelo menos uma das seguintes API esteja disponível para o HSM em avaliação:

- Microsoft CryptoAPI[26]
- PKCS#11 v.2.11[27]
- JCE[12]/JCA[13]
- interface própria[14]
- engine OpenSSL (vide seção 4.5)

Para cada uma das API, define-se um conjunto mínimo de funções que deve estar presente, devendo, quando possível, ser compatível com os sistemas operacionais Microsoft Windows 2000, Linux Kernel 2.4 ou suas versões mais recentes.

3.3.14 Requisitos para restrição de substâncias nocivas

O MCT-7 recomenda que os HSM estejam em conformidade com diretivas estabelecidas quanto a restrição da utilização de substâncias nocivas, visando à

[12] Extensão de criptografia JAVA, do inglês Java Cryptography Extension

[13] API de comunicação Java, do inglês Java Communications API

[14] conjunto de funções disponibilizadas pelo fabricante do módulo, sem seguir um padrão conhecido, que permitirá a utilização das chaves criptográficas gerenciadas.

preservação da saúde humana e do meio ambiente. Como referência cita-se o RoHS[15][28] e o WEEE[16][29], ambas diretivas da União Européia, sendo que a última define o correto tratamento, recuperação e reciclagem de resíduos de materiais eletroeletrônicos.

3.4 Conclusão

As normas de avaliação apresentadas nesse capítulo diferenciam-se principalmente pelo grau de abrangência, onde a FIPS 140-2 certifica qualquer hardware criptográfico, enquanto o MCT-7 restringe-se a HSM voltados para ICPs.

Além disso, o MCT-7, nos níveis 2 e 3 de segurança da homologação, requer o depósito de código fonte, requisito muitas vezes não aprovado pelos fabricantes dos HSM, por ameaçar o segredo industrial.

O MCT-7 se encaixa perfeitamente com os objetivos do HSM do GT ICPEDU II, já que este último teve seu desenvolvimento direcionado para ICPs e possui plataforma totalmente aberta.

A ausência de elementos que viabilizem um processo de auditoria afetam tanto a FIPS 140-2 quanto o MCT-7. Além disso, como o foco deste último são as aplicações de ICPs, esse comportamento torna o ambiente deficiente, já que um sistema de auditoria forte é requisito fundamental para a utilização de cerimônias que visam garantir a correta operacionalização de HSM no âmbito de ICPs.

Outro ponto a se considerar é a falta de requisitos para o estabelecimento de um controle mais restrito no uso de chaves criptográficas, como também na criação e recuperação de cópias de segurança dos HSM. A rastreabilidade das mesmas no âmbito de ICPs é fundamental, já que suas ACs e ARs definem o ponto de confiança de toda uma hierarquia.

[15] do inglês Restriction to the use of Hazardous Substances
[16] do inglês Waste from Electrical and Electronic Equipament

Capítulo 4

OpenSSL

4.1 Introdução

O OpenSSL[30] é uma poderosa e robusta ferramenta para soluções envolvendo criptografia. Desenvolvido na linguagem C sob esforço colaborativo de código aberto, o OpenSSL possui uma aplicação de linha de comando para execução de operações criptográficas e uma rica biblioteca, que inclui implementação dos dois principais protocolos de comunicação segura: SSL (Secure Socket Layer v3)[31] e TLS (Transport Layer Security v1)[32].

Atualmente (julho de 2008) na versão 0.9.8h, o OpenSSL foi baseado na biblioteca SSLeay desenvolvida por Eric A. Young e Tim J. Hudson e é distribuído sob licença estilo apache, que basicamente significa que qualquer um é livre para usá-la, comercialmente ou não.

O OpenSSL é largamente empregado em aplicações criptográficas comerciais, como no módulo de conexão segura (mod-ssl) do servidor Web mais utilizado no mundo, o Apache[33]. Adicionalmente, o OpenSSL possui uma versão certificada FIPS 140-2 nível 1 (vide seção 4.6), que mostra a abrangência do projeto.

As próximas seções descreverão as principais funcionalidades e recursos implementados no OpenSSL, incluindo sua aplicação de linha de comando para executar operações criptográficas, a biblioteca de auxílio ao desenvolvimento de aplicações

e o suporte a engines, que permitem a utilização de outras implementações dos algoritmos suportados pelo OpenSSL, possibilitando, por exemplo, sua integração a hardware criptográficos.

4.2 Aplicação de Linha de Comando

O pacote do OpenSSL inclui uma aplicação de linha de comando que possibilita a utilização de todas as funções criptográficas implementadas na biblioteca. Todas essas funções podem ser agrupadas em:

- operações de criptografia simétrica;
- criação e manipulação de chaves e parâmetros criptográficos de algoritmos assimétricos;
- operações de criptografia assimétrica;
- cálculo de mensagens de resumo (hash);
- criação de certificados X.509[10], requisições de certificados (REQ)[34] e lista de certificados revogados (LCR)[10];
- testes de servidores e clientes que implementam protocolos SSL/TLS;
- codificação de e-mails assinados e/ou cifrados;
- requisição, geração e verificação de carimbos de tempo (implementado na futura versão 0.9.9 do OpenSSL).

As funções disponíveis na aplicação de linha de comando são implementadas visando a maior abrangência possível. Por exemplo, no caso de geração de um certificado, seu conteúdo pode ser gravado em formato PEM (Base 64), DER (Binário) ou NET (um formato obsoleto da Netscape).

O poder de customização das operações possibilita o desenvolvimento de aplicações sem utilização da biblioteca e sim chamadas da linha de comando, que possui seu poder de ação aumentado com o suporte a arquivos de configuração (seção 4.3.1), automatizando e padronizando diversas operações. Além disso, existem comandos que foram criados especialmente para integrar aplicações com o OpenSSL linha de comando, tais como as funções que informam se um determinado algoritmo é suportado.

4.3 Infra-estrutura de Suporte

Além de algoritmos criptográficos, o OpenSSL implementa inúmeras funções e funcionalidades auxiliares que facilitam o desenvolvimento de aplicações reais.

Algumas das facilidades, entre elas a operação de números grandes, tratamento de erros e abstração de entrada e saída, serão tratadas separadamente em subseções neste capítulo, apontando suas características e utilidade. As facilidades implementadas podem ser úteis para a aplicação de linha de comando e/ou aplicações utilizando a biblioteca OpenSSL.

4.3.1 Arquivos de Configuração

O OpenSSL inclui a capacidade de ler e interpretar arquivos de configuração em tempo de execução. Com estrutura interna baseada em seções, esses arquivos podem ser utilizados através da aplicação de linha de comando ou pela biblioteca OpenSSL. O item $openssl\_conf$ é utilizado pelo OpenSSL para mapear a seção principal (como pode ser visto na tabela 4.1).

Os arquivos de configuração são essenciais no gerenciamento de uma autoridade certificadora através da aplicação de linha de comando do OpenSSL, definindo o conteúdo dos certificados e lista de certificados revogados. Adicionalmente, todas as extensões suportadas ou não pelo OpenSSL podem ser definidas com a utilização de arquivos de configuração. Para extensões não suportadas é necessário definir sua seqüência ASN.1[35] correspondente.

Um processo de carga e utilização de engines (vide seção 4.5) também pode ser automatizado utilizando arquivos de configuração, suportando até eventuais parâmetros que podem ser necessários em alguns casos. O arquivo de configuração necessário para carga da engine do HSM do GT ICPEDU II pode ser visto na tabela 4.1.

Tabela 4.1: Arquivo de configuração do OpenSSL para carga da engine do ASI-HSM.

```
1.  openssl_conf = openssl_init
2.  [ openssl_init ]
3.  engines = engine_section
4.  [ engine_section ]
5.  openhsmd = openhsmd_section
6.  [ openhsmd_section ]
7.  engine_id = openhsmd
8.  dynamic_path = ../engines/engine_openhsmd.so
9.  ADDRESS_CONN = 192.168.1.1
10.  PORT_CONN = 5001
```

A API dos arquivos de configuração do OpenSSL pode ser extendida e utilizada em outras aplicações, evitando a implementação de um módulo de software com a mesma finalidade, economizando tempo e minimizando as chances de ocorrência de erros.

4.3.2 Funções de Callback

O objetivo dessas funções é proporcionar flexibilidade e extensão das funcionalidades de uma biblioteca, permitindo aos desenvolvedores implementar o comportamento de acordo com a necessidade de sua aplicação.

As funções de callback não são executadas diretamente pela aplicação, ao invés disso, seu ponteiro é passado para outras funções, de onde elas serão chamadas. Para isso, as funções de callback devem seguir o formato previamente definido pela biblioteca que as utilizam, isto é, a quantidade e tipos dos argumentos e o tipo da variável de retorno.

A figura 4.1 demonstra o funcionamento das funções de callback, salientando a indiferença da biblioteca quanto a operação que está sendo executada. A

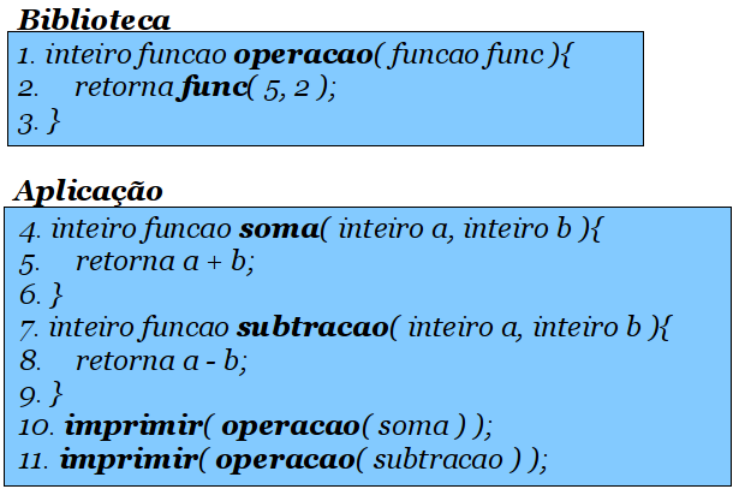

Figura 4.1: Funcionamento das funções de callback, empregadas no OpenSSL.

biblioteca está passando os inteiro 5 e 2 como parâmetro de uma funções implementada pela aplicações. A execução das linhas 10 e 11 resulta, respectivamente, nos valores 7 e 3.

O OpenSSL emprega funções de callback em várias de suas funcionalidades, tal como no processo de geração de números primos, que, com sua utilização, pode-se acompanhar cada tentativa do processo de encontrar o número primo apropriado.

Adicionalmente, para algumas funcionalidades da biblioteca OpenSSL, o emprego dessas funções é fundamental, como no suporte a execução multi-thread, que é o assunto da próxima sub-seção.

4.3.3 Suporte Multi-thread

A biblioteca OpenSSL emprega controle de acesso concorrente as estruturas consideradas críticas quando executado em modo multi-thread, protegendo-os com uso de semáforos.

Entretanto, o efetivo uso desse controle fica a cargo da aplicação que está utilizando a biblioteca OpenSSL, pois esta deve implementar a função que será responsável por checar e bloquear o acesso aos recursos. Essa função de callback deve ser configurada no OpenSSL no início da execução da aplicação.

O controle de acesso a recursos na biblioteca requer obrigatoriamente a definição da função de callback que irá implementar as operações de bloqueio e liberação dos semáforos. Nas chamadas a função de callback solicitando acesso a um recurso compartilhado, existe a distinção de operações de somente-leitura das operações leitura-escrita. Fica a cargo da aplicação o nível de controle desejado. A aplicação de linha de comando do OpenSSL implementa uma função de callback que trata indiferentemente operação de leitura e escrita, garantindo acesso exclusivo a qualquer um dos casos.

Na seção 4.5 será abordado com detalhes o controle de acesso simultâneo realizado pela biblioteca para gerenciar o suporte as engines.

4.3.4 Matemática de Precisão Arbitrária

A biblioteca OpenSSL possui incorporado um conjunto de funções capaz de manipular e executar operações matemáticas sobre números grandes, isto é, que não podem ser representados nas variáveis do tipo inteiro de 32 ou 64 bits encontrado nas plataformas computacionais convencionais.

A manipulação de números grandes é essencial para a implementação, entre outra, do algoritmo de chaves públicas RSA, considerando que chaves deste algoritmo devem ter um tamanho mínimo de segurança de 1024 bits, como definido na FIPS 140-2.

O conjunto de funções implementadas inclui todas operações normalmente disponíveis sobre variáveis do tipo inteiro em linguagens de programação em geral, incluindo operações modulares, utilizadas em cifragens e decifragens RSA, além da geração de números primos, que incorpora os testes conhecidos de primalidade.

Esta funcionalidade é outro recurso que pode ser usufruido por aplicações utilizando a biblioteca OpenSSL, diminuindo a complexidade de aplicações com necessidades semelhantes.

4.3.5 Tratamento de Erros

O OpenSSL possui um sistema centralizado para tratar erros durante sua execução, atribuindo identificadores únicos para cada parte/módulo de sua biblioteca. Esses identificadores são utilizados para separar os identificadores de funções e erros de cada módulo.

Quando um erro acontece, 5 informações referentes ao erro são registradas:

- o identificador do módulo;
- o identificador da função;
- identificador do erro;
- nome do arquivo (automaticamente obtido com a macro $\_\_FILE\_\_$ da linguagem C);
- linha na qual o erro foi disparado (automaticamente obtido com a macro $\_\_LINE\_\_$ da linguagem C);

Cada módulo é responsável por carregar no sistema o texto correspondente a cada identificador, tanto de função quanto de erro, permitindo a conversão dos erros ocorridos durante a execução do sistema em texto legível.

O OpenSSL disponibiliza a função $ERR\_load\_crypto\_strings$ para carregar todos os identificadores e seus respectivos textos de todos seus módulos nativos de uma só vez. Normalmente, esse é um dos primeiros passos dentro de uma aplicação utilizando a biblioteca OpenSSL.

Na aplicação de linha de comando, os erros são impressos ao final da execução de qualquer operação que venha a falhar. Já no caso de aplicações utilizando a biblioteca OpenSSL, as funções de manipulação dos erros precisam ser explicitamente acionadas.

O sistema de erros do OpenSSL suporta a adição de novos módulos de erros, permitindo que bibliotecas e aplicativos o utilizem para gerenciamento de seus

erros, deixando o sistema uniforme. O OpenSSL inclui scripts que vasculham o código fonte e geram os arquivos necessário para essa integração.

4.3.6 Abstração de Entrada e Saída

A abstração de entrada e saída (BIO) do OpenSSL permite a comunicação entre dois pontos de forma transparente, isto é, sem a manipulação de características inerentes a fonte/destino da informação. Além disso, camadas podem ser adicionadas nesses canais de comunicação, provendo funcionaliades extras. No OpenSSL, os BIOs são agrupados em dois tipos primitivos: comunicação e filtro.

Os BIOs de comunicação são utilizados na troca de informação entre duas entidades, podendo essas serem arquivos, sockets, memória ou simplesmente um outro BIO para comunicação dentro da própria aplicação. O uso de algumas dessas entidades incorporam um outro sub-tipo de BIOs, chamado de $descriptor$. Este sub-tipo se aplica a entidades que utilizam descritores de arquivos, como no caso de sockets.

Um exemplo da utilização dos BIOs de comunicação seria o processo de escrita de dados em um arquivo no disco rígido, através de um BIO arquivo. O mesmo processo vale para leitura de arquivos.

Os BIOs do tipo filtro são recursos adicionais aos BIOs de comunicação, capazes de processar os dados transferidos, agindo como uma camada extra na comunicação entre dois pontos. O processamento realizado pelo filtro depende da sua finalidade, que pode ser:

- conversor base64 (conversão e vice-versa);
- calculo de funções de resumo;
- criptografia simétrica (cifrar e decifrar);
- protocolo SSL;

No caso do exemplo anterior, se os dados a serem escritos no arquivo precisam ser convertidos em base64, acopla-se o BIO filtro específico em frente ao BIO

arquivo, de forma que os dados serão escritos no BIO base64 e terminarão no arquivo de destino, com os dados convertidos em base64. Não existe limite da quantidade de filtros que podem ser conectados.

O emprego de filtros deixa o processo simples e uniforme, independente da quantidade de camadas existentes, onde a saída de uma camada é a entrada da próxima, até atingir o nó final. Cada BIO do tipo filtro realiza uma operação em um sentido que a informação é transferida e a operação inversa no sentido oposto, ou seja, voltando no exemplo do arquivo em base64, tudo que for escrito será convertido em base64, e o que for lido retornará ao formato original. Vale destacar que o processo inverso não se aplica para a funções de resumo, devido sua propriedade de irreversibilidade.

A abstração provida pelos BIOs de comunicação já justificaria a existência dos BIOs, entretanto, com a adição dos tipos filtros, sua utilidade multiplica-se de tamanho, agregando inúmeras possibilidades. É possível por exemplo criar um socket usando BIO e posteriormente adicionar um BIO ssl nas pontas, de forma a estabelecer um canal seguro de comunicação.

A utilização de BIOs está difundida em todo código do OpenSSL, englobando todas as funções de leitura e escrita de arquivos do sistema, tais como o manuseio de chaves públicas e privadas.

A abstração de Entrada e Saída apresentada nesta seção é voltada para desenvolvedores utilizando a biblioteca OpenSSL, já que na aplicação de linha de comando, apesar de utilizá-los intensamente, esse recurso passa despercebido.

4.4 Documentação

Documentação é um dos pontos mais importantes para que uma ferramenta, que é necessariamente voltada para usuários em geral, seja facilmente assimilada e compreendida na sua forma de utilização e operação. O OpenSSL, apesar de toda sua genialidade explicada aqui, peca neste quesito. A falta de documentação, principalmente na biblioteca que permite integração com outros aplicativos, dificulta muito sua utilização.

As constantes atualizações e geração de novas versões acabam aconte-

cendo sem os devidos cuidados com os materiais auxiliares de aprendizado, fazendo com que a pouca documentação existente fique desatualizada e muitas vezes piorando ainda mais sua compreensão.

O código fonte acaba se tornando o principal meio de obter conhecimento detalhado dos objetivos das funções e procedimentos, resultando muitas vezes em mais tempo gasto entendendo a biblioteca do que realmente pensando na solução do problema. A utilização de outras fontes de documentação como livros[36], artigos e lista de discussões são alternativas.

4.5 Engine

O OpenSSL implementa cada um dos algoritmos criptográficos por ele suportado, possibilitando que sejam operacionalizados por outras aplicações. Entretanto, em algumas situações, é necessário a utilização de outras implementações, como no caso dos aceleradores criptográficos, que implementam os algoritmos em hardware e suportam grande poder de processamento. Nestes casos, a saída é a utilização de engines OpenSSL.

Uma engine OpenSSL é um módulo de software que permite ao seu desenvolvedor substituir as implementações padrões dos algoritmos existentes. As engines podem implementar um ou mais algoritmos e são classificadas de estáticas ou dinâmicas.

As engines estáticas são integradas e compiladas com o código fonte do OpenSSL, tornando-as parte do pacote criptográfico. Elas são automaticamente carregadas sempre que o OpenSSL é referenciado. Por outro lado, as engines dinâmicas podem ser carregadas em tempo de execução, sem necessitar nenhuma alteração na versão original do OpenSSL. Uma engine dinâmica é gerada da mesma forma que uma biblioteca dinâmica, sendo normalmente identificada pela extensão $.so$ no Unix[1] e $.dll$ no Microsoft Windows[2].

Internamente, o OpenSSL define uma engine como uma estrutura chamada $ENGINE$, que contem os seguintes campos:

[1]Sistema operacional de software livre.
[2]Sistema operacional da Microsoft Corporation.

- id: identifica unicamente uma engine no OpenSSL. Tanto a aplicação de linha de comando quanto a biblioteca utilizando este campo como referência;

- $name$: tem somente papel descritivo;

- rsa_meth, dsa_meth, dh_method, $ecdsa\_meth$, $ecdh\_meth$ e $rand\_meth$: estrutura definindo os ponteiros das funções responsáveis pelo manuseio do algoritmo em questão (RSA, DSA, DH, ECDSA, ECDH ou funções de aleatoriedade). Por exemplo, o campo $rand\_meth$ contém os ponteiros para funções capazes de gerar dados randômicos e pseudorandômicos, adicionar entropia, carregar o estado do gerador, entre outros;

- $ciphers$ e $digests$: diferentemente dos algoritmos anteriores, onde cada algoritmo possuia uma estrutura própria, os algoritmo simétricos e funções de resumo possuem o ponteiro para apenas uma função cada;

- $init$ e $finish$: funções de inicialização e fim da execução da engine. Servem para a engine criar e preparar as estruturas internas necessárias para seu funcionamento. Por exemplo o estabelecimento da conexão com um HSM.

- $destroy$: função chamada antes de remover a engine da lista de engines do OpenSSL. Deve ser utilizada para liberar qualquer tipo de memória alocada, como por exemplo a descarga de suas mensagens de erros do sistema de erro do OpenSSL;

- $ctrl$: ponteiro para uma função capaz de interpretar os parâmetros necessários pela engine. Por exemplo o endereço IP de um HSM conectado na rede;

- $load\_privkey$ e $load\_pubkey$: função para carregar chaves públicas e privadas gerenciadas através da engine. O retorno desta função é uma estrutura EVP_PKEY, que pode ser utilizada para encapsular todas as chaves assimétricas suportadas pelo OpenSSL;

- cmd_defns: define os comandos aceitos e interpretados por $ctrl$. Cada comando contém um identificador (que deve iniciar com o valor da constante

$ENGINE\_CMD\_BASE$, seguindo com $ENGINE\_CMD\_BASE + 1$, ...), nome do commando, sua descrição e o tipo do valor de entrada esperado. Os possíveis tipos de respostas são: numérico ou texto. Os comandos ainda podem não necessitar nenhuma entrada ou até mesmo ser marcado para uso interno;

- $struct\_ref$ e $funct\_ref$: utilizado pelo OpenSSL para controle do número de referências à estrutura e ao conjunto de funções de uma engine. Garante a utilização da engine em várias partes do aplicação, evitando, por exemplo, a inicialização de uma engine já inicializada;

- outros: parâmetros utilizados para controle interno, como $prev$ e $next$, que mantêm uma lista duplamente encadeada das engines disponíveis, ex_data, utilizada para armazenamento de dados referente a engine e $flags$, que define suas características.

Algumas peculiaridades devem ser levadas em conta no desenvolvimento de uma engine OpenSSL para a utilização de chaves gerenciadas por um HSM. Primeiramente, o que fazer com a função $loadprivkey$, já que não se quer carregar a chave privada na memória da máquina hospedeira, não faria o mínimo sentido utilizar uma HSM para proteger o ciclo de vida de chaves criptográficas e deixá-la em aberto desta forma. Uma abordagem comum para resolver esse problema é retornar a chave pública nas duas funções $loadprivkey$ e $loadpubkey$. Em um primeiro momento pode parecer estranho, mas isso é possível porque além de o OpenSSL utilizar uma única estrutura para representar chaves públicas e privadas de um mesmo algoritmo, será a engine que irá executar as operações sobre essa chave. Portanto, toda vez que uma operação com a chave privada for realizada, a própria engine será chamada e a chave privada correta pode ser identificada baseado na chave pública, utilizando seus valores públicos (números primos n e e) ou através do identificador da chave explicitamente armazenado na variável $exdata$, que como na engine, existe também nas estruturas de chaves.

Adicionalmente, a estrutura de chaves criptográficas da biblioteca possui um atributo chamado $engine$, que aponta para a engine responsável por realizar as operações criptográficas da chave. Portanto, após a carga da chave atráves das funções $load\_privkey$ e $load\_pubkey$, deve-se substituir o valor deste atributo pela engine atual.

Conceitualmente falando, a diferença entre uma cifragem assimétrica e uma assinatura está na chave que é utilizada na operação, chave pública no primeiro caso e chave privada no segundo, e também na informação sobre a qual a ação é realizada, no último caso o resultado de uma função de resumo. Sabendo disso, o último ponto relevante na implementação de uma engine está na estrutura de operações RSA (RSA_METHOD). Ela possui basicamente 6 funções:

- rsa_pub_enc: cifragem utilizando uma chave pública;
- rsa_priv_dec: decifragem utilizando uma chave privada;
- rsa_priv_enc: cifragem utilizando uma chave privada;
- rsa_pub_dec: decifragem utilizando uma chave pública;
- rsa_sign: geração de assinaturas;
- rsa_verify: verificação de assinaturas;

Como pode ser observado, as funções rsa_priv_enc e rsa_sign possuem a mesma finalidade de geração de assinaturas, o que acontence com rsa_pub_dec e rsa_verify também. Uma engine não deve implementar estas quatro funções, e sim escolher o conjunto que preferir. O ponto chave aqui é utilizar o campo $flags$ da estrutura das chaves RSA para marcar qual o conjunto de funções que deve ser acionado. Se a flag $RSA\_FLAG\_SIGN\_VER$ estiver configurada, as funções rsa_sign e rsa_verify serão utilizadas. As quatro funções ainda existem por uma questão de compatibilidade com versões anteriores do OpenSSL, sendo por padrão utilizado $rsa\_privenc$ e rsa_pub_dec, apesar de rsa_sign e rsa_verify serem mais novas.

4.6 OpenSSL FIPS 140-2 nível 1

O OSSI[37] (Open Source Software Institute[3]), entidade sem fins lucrativos, em conjunto com o núcleo de desenvolvimento do OpenSSL preparou e subme-

[3]Instituto de Software de Código Aberto em tradução literal

teu a biblioteca do OpenSSL para avaliação nos requisitos da FIPS 140-2, alcançando a certificação da versão 1.0 do OpenSSL FIPS em março de 2006[38].

O OpenSSL FIPS 1.0 submetido a certificação contém um conjunto limitado de funções do OpenSSL 0.9.7 e sua utilização por um aplicação difere-se um pouco do usual. O código do OpenSSL FIPS deve ser compilado para gerar o código objeto e esse código objeto deve ser estaticamente ligado[4] no processo de compilação do código fonte da aplicação. A biblioteca criptográfica do OpenSSL também deve ser ligada na aplicação, podendo esta ser estática ou dinâmica.

Os desenvolvedores do OpenSSL FIPS implementaram várias técnicas para verificação de integridade nas várias etapas que o módulo pode assumir, com checagens automáticas do código fonte durante a compilação, no código objeto durante a ligação e no código executável quando o modo FIPS do módulo é ativado. Durante a ativação do modo FIPS, através da função $FIPS\_mode\_set$, os auto-testes são iniciados e todos devem obrigatoriamente passar. Uma vez em modo FIPS, somente operações de algoritmos aprovados podem ser executadas. A tabela 4.2 contém os algoritmos criptográficos aprovados na versão FIPS do OpenSSL.

Tabela 4.2: Algoritmos Criptográficos Aprovados no OpenSSL FIPS versão 1.1.2

Tipo de Algoritmo	**Algoritmos**
Assimétrico	RSA
Simétrico	3DES (nos modos: CBC, CFB8, CFB64, ECB e OFB) AES de 128, 192 or 256 bits (nos modos CBC, CFB8, CFB128, ECB e OFB)
HMAC	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384 e HMAC-SHA-512
Funções de resumo	SHA-1, SHA-224, SHA-256, SHA-384 e SHA-512
RNG	ANSI X9.31[39]

Em junho de 2006, o certificado da versão 1.0 foi suspenso, porque identificou-se que funções críticas não estavam implementadas dentro do perímetro de segurança. Após as alterações necessárias, submeteu-se uma nova versão, 1.1, mas a

[4]processo de associação dos vários códigos objetos com o objetivo de resolver pendências externas, gerando o código executável

reativação do certificado nunca ocorreu. A versão foi alterada de 1.1 para 1.1.1 e submetida a um novo processo de certificação.

A versão 1.1.1[40], aprovada em 06/02/2007, foi revogada em 30/11/2007 por dois problemas no gerador de números aleatórios: um bug no auto-teste contínuo e uma vulnerabilidade no processo de inclusão de sementes. Uma nova versão do código, 1.1.2, com essas deficiências sanadas, foi novamente submetida a certificação, porém, no meio do processo, os requisitos dos auto-testes para operações DSA mudou de 512 bits para a utilização de chaves de 1024 bits. Para evitar mais atrasos, o algoritmo DSA foi retirado do processo de certificação, diminuindo o números de algoritmos aprovados. A alteração do auto-teste do DSA de 512 para 1024 bits era realmente simples, mas significaria o reinício do processo de certificação, uma vez que o código fonte iria sofrer alterações.

A versão 1.1.2 recebeu a certificação em 6 de fevereiro de 2008[41]. Uma nova versão, 1.2, já está em processo de desenvolvimento e será baseada na versão 0.9.8 do OpenSSL.

4.7 Conclusão

O OpenSSL é uma ferramenta de criptografia com suporte a um grande número de algoritmos criptográficos. Seus vários anos de desenvolvimento e testes realizados por muitos usuários ao redor do mundo passam credibilidade quanto a qualidade dos resultados obtidos. Além disso, alguns de seus algoritmos criptográficos fazem parte do OpenSSL FIPS, que possui certificação FIPS 140-2 nível 1.

Por outro lado, sua documentação é um tanto deficiente, deixando desenvolvedores sem muitas alternativas de auxílio. As principais fontes de informações acabam sendo seu código fonte e suas listas de discussão, que estão disponíveis na página do projeto.

As engines OpenSSL permitem a integração de aplicativos com módulos criptográficos, tornando possível a utilização de chaves criptográficas gerenciadas em um HSM. É nessa plataforma que o desenvolvimento do ASI-HSM se baseou,

suprindo o requisito de ser compatível com o sistema de gerenciamento de certificado digitais da ICPEDU.

Capítulo 5

ASI-HSM

5.1 Introdução

O projeto ICPEDU II surgiu da necessidade de se criar um módulo de segurança criptográfica com tecnologia brasileira, de código aberto e de baixo custo, voltado principalmente para a implantação da Infra-estrutura de Chaves Públicas para Ensino e Pesquisa (ICPEDU). O HSM fruto deste projeto é o ASI-HSM.

O ASI-HSM será descrito neste capítulo, com apresentação de sua arquitetura e os componentes necessários para sua operacionalização. O protocolo implementado internamente para a gestão do ciclo de vida de chaves criptográficas é conhecida como OpenHSM e é abordado nos capítulos 6 e 7. Os protocolos e algoritmos do OpenHSM foram concebidos para serem embarcados em hardware criptográfico que provesse os mecanismos de proteção conforme estabelece os requisitos apresentados no Capítulo 3. A arquitetura do ASI-HSM é descrita na Seção 5.2.

O HSM pode ser visto como um sistema provedor de serviços criptográficos. Para poder utilizá-lo a partir de uma aplicação numa máquina hospedeira, é necessário uma interface de comunicação. A interface escolhida para o ASI-HSM foi a ”engine”do OpenSSL, descrito no Capítulo 4. A interface do protótipo é descrita na Seção 5.4.

Além deste canal de comunicação para acesso aos serviços, foi proje-

tado um sistema de gestão remota do HSM, que é executado na máquina hospedeira, para realizar a configuração, administração, operação e auditoria do material criptográfico protegido pelo HSM e gerenciado pelo OpenHSM. Este sistema é apresentado na Seção 5.3.

5.2 Arquitetura

O ASI-HSM é composto por uma série de componentes de hardware e software, que cooperam entre si com o intuito de proteger, gerenciar e monitorar os recursos e serviços disponíveis no mesmo. A figura 5.1 apresenta uma visão geral da arquitetura do módulo[1].

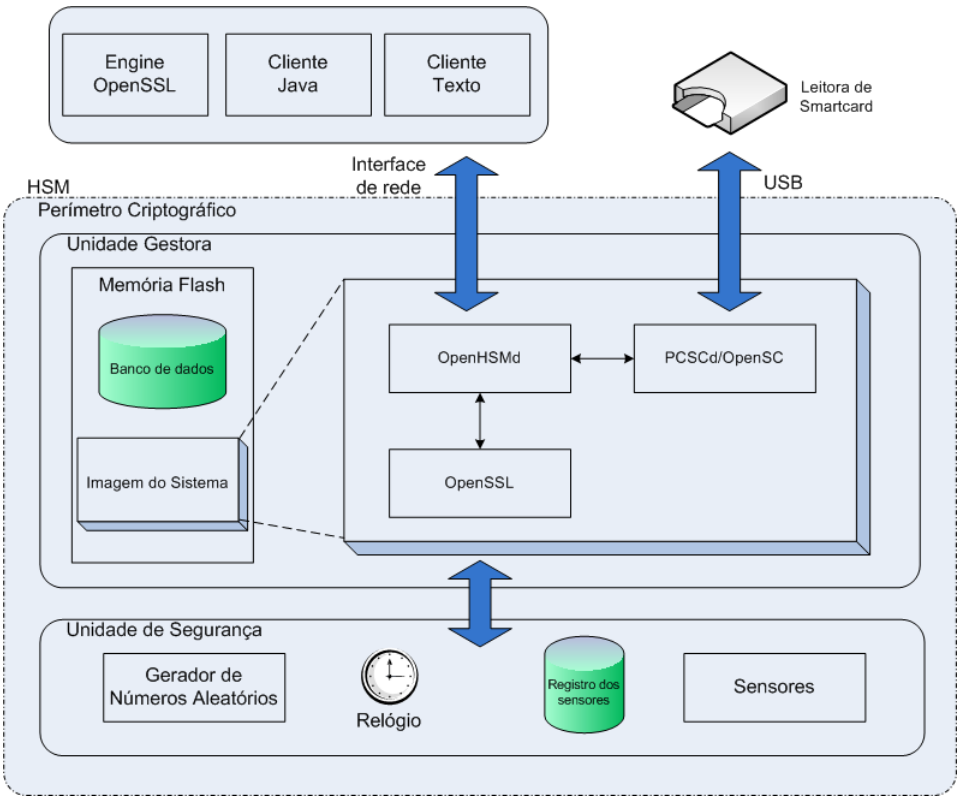

Figura 5.1: Visão geral da arquitetura do HSM da GT ICPEDU II

Na parte interna ao perímetro criptográfico, o módulo conta com dois componentes principais: a Unidade de Segurança e a Unidade Gestora.

[1] créditos pela figura para Juliano Romani

A Unidade de Segurança (US), dispositivo desenvolvido especificamente para monitorar o funcionamento do módulo, conta com uma série de sensores que visam a detecção de qualquer tentativa de violação ou risco de comprometimento da segurança por influência de fatores externos, sendo qualquer comportamento inesperado registrado no sistema de registros interno à US. O módulo emprega sensores de tensão, temperatura, luminosidade e detecção de intrusão física, sendo o último provido através de uma malha de circuitos que envolve o perímetro criptográfico. Além disso, é dentro dessa unidade que ficam o gerador de números aleatórios (PNG) e o relógio de alta-estabilidade, usado no processo de geração dos pares de chaves gerenciados pelo módulo e controle de tempo interno, respectivamente.

A Unidade Gestora (UG) consiste de um conjunto de software/hardware responsável por hospedar as aplicações e dados referentes à execução do módulo, como por exemplo, as ferramentas e bibliotecas que compõem o OpenSSL, bem como as chaves gerenciadas pelo módulo.

Operando sobre a UG estão as aplicações envolvidas na gerência do ciclo de vida das chaves privadas do HSM (OpenHSMd), bem como a memória persistente do módulo, uma Compact Flash (CF), responsável por armazenar os dados relacionados à configuração do HSM, as chaves simétricas e assimétricas, certificados, entre outras informações úteis ao funcionamento do HSM. Ainda na memória flash, encontra-se a versão customizada do sistema operacional FreeBSD embarcada para gerenciamento da solução como um todo.

A comunicação com o mundo externo está restrito a duas portas físicas, uma porta de rede e uma porta USB. A porta de rede é utilizada para gerenciamento do HSM, através dos aplicativos de administração remota (seção 5.3), e para uso das chaves gerenciadas, através de uma engine OpenSSL (seção 5.4). Na porta USB conecta-se o leitor de smartcards utilizado nos processos de criação e autenticação dos usuários do módulo.

5.3 Aplicativos de Administração Remota

O gerenciamento do HSM é realizado através de dois aplicativos de administração remota: interface texto (linha de comando) e interface gráfica. Ambas tem o mesmo conjunto de funções e comandos e se conectam ao HSM através de um canal seguro de comunicação, túnel SSL.

A interface texto, uma vez conectada ao HSM, provê um ambiente no mesmo estilo linha de comando do OpenSSL, aonde comandos podem ser executados com seus respectivos parâmetros e opções, sendo que todas as funções possuem sua documentação de fácil acesso na própria interface. É uma forma eficiente e eficaz de obter a informação requerida. A figura 5.2 apresenta um exemplo da interface texto conectada ao HSM.

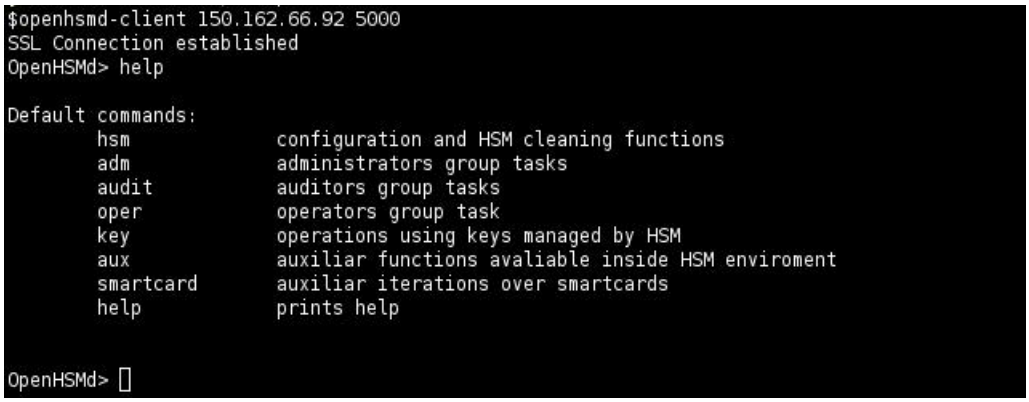

Figura 5.2: Interface texto para administração remota do HSM do GT ICPEDU II

Por outro lado, a interface gráfica, desenvolvida em Java[2], é rica em detalhes com componentes ativos que se atualizam baseados na configuração atual do HSM. Outro ponto forte dessa interface é sua portabilidade. A figura 5.3 apresenta um exemplo da interface gráfica conectada no HSM.

As duas aplicações de administração remota do módulo são internacionalizadas, suportando atualmente os idiomas inglês e português.

[2]linguagem de programação, disponível em http://java.sun.com/

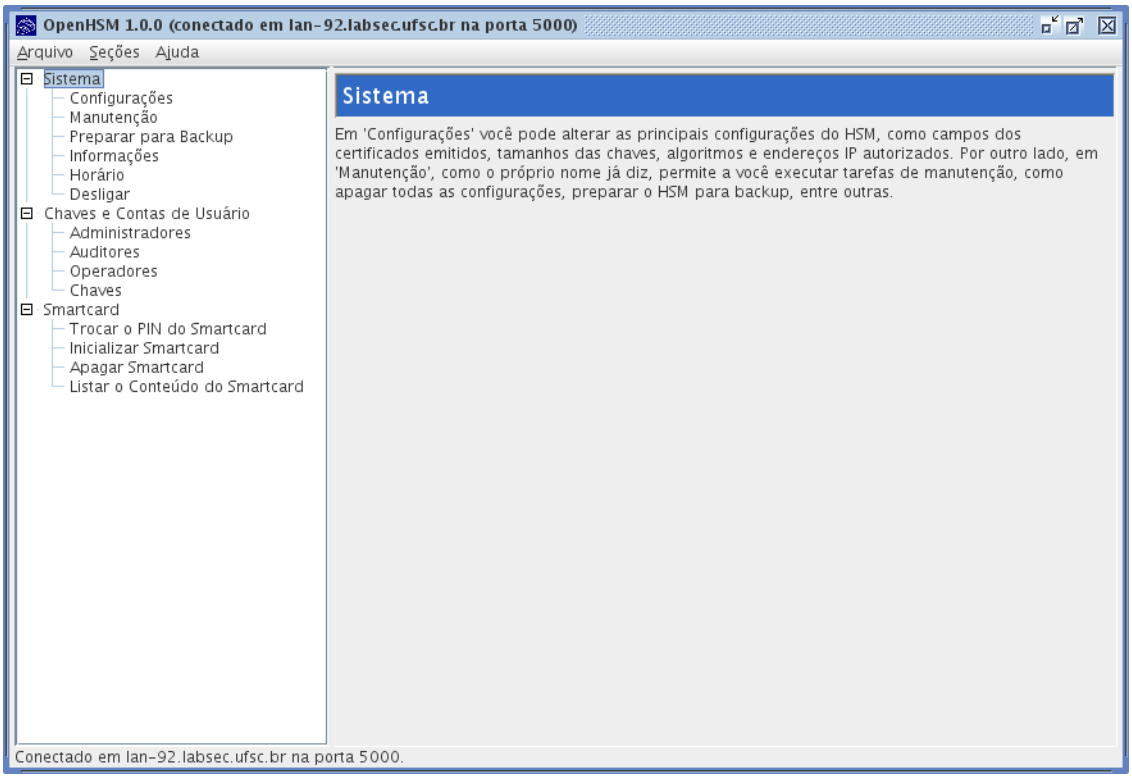

Figura 5.3: Interface gráfica para administração remota do HSM do GT ICPEDU II

5.4 Engine OpenSSL

Um módulo criptográfico deve prover uma API de integração que permita a uma aplicação utilizar suas chaves criptográficas gerenciadas. Essa API pode seguir um padrão já estabelecido, como por exemplo PKCS#11, CryptoAPI, Engine OpenSSL, ou ser uma interface própria do fabricante do HSM.

A escolha da API de integração adequada levou em consideração dois principais fatores: grau de complexidade de implementação e o número de aplicativos afetados. Desta forma, os serviços criptográficos do HSM são providos através de uma engine OpenSSL, que pode ser carregada de forma estática ou dinâmica. O módulo provê as seguintes funcionalidades através desta engine:

- operações sobre chaves privadas RSA: permite o uso das chaves criptográficas gerenciadas pelos HSM;

- gerador de números aleatórios: este serviço utiliza o gerador de números aleatórios disponível na unidade de segurança.

5.5 Conclusão

Este capítulo apresentou a arquitetura e os principais componentes do HSM desenvolvido no projeto ICPEDU II, o ASI-HSM. Este desenvolvimento, que parecia distante devido a falta de documentação e publicações existente no mundo acadêmico, por serem normalmente guardadas a sete chaves por empresas comerciais, se mostrou totalmente viável e os objetivos do projeto foram alcançados, sendo concebido utilizando apenas software livre.

Os próximos dois capítulos irão descrever em detalhes o protocolo de gerenciamento do ciclo de vida de chaves criptográficas utilizados no ASI-HSM, o OpenHSM.

Capítulo 6

OpenHSM - Operacionalização

O OpenHSM é um protocolo para gerenciamento completo do ciclo de vida de chaves criptográficas de HSMs. O foco principal deste protocolo é suprir as necessidades encontradas na implantação de infra-estruturas de chaves públicas, tais como rastreabilidades de chaves e com operações totalmente auditáveis.

O gerenciamento do ciclo de vida de chaves criptográficas de forma segura é o objetivo de qualquer HSM, sendo que este objetivo pode levar um longo tempo de maturação até ser plenamente alcançado. Sabendo disso, este capítulo apresenta o protocolo do OpenHSM, que foi primeiramente apresentado em uma dissertação de mestrado[4] defendida em 2005 e vem sendo aprimorado em várias ocasiões, tais como trabalhos de conclusão de curso[3], outras dissertações e artigos[5][6]. Apesar disso, o conjunto destas publicações não apresenta de forma coerente todos os sub-protocolos do OpenHSM.

As mudanças nos sub-protocolos do OpenHSM, além de estruturais, também ocorreram na forma de sua representação, com o objetivo descrevê-los formalmente. A descrição formal é o primeiro passo para poder-se realizar a verificação formal da corretude dos protocolos. As alterações e aprimoramentos estruturais são relativos aos protocolos de criação e recuperação de cópias de segurança do material criptográfico, que são descritos no Capítulo 7.

6.1 Aspectos Gerais

Esta seção apresenta as definições e características inerentes aos protocolos do OpenHSM, de forma a ajudar na compreensão dos mesmos. Além disso, dois sub-protocolos, que também darão suporte para as próximas seções, serão abordados: a criação e a autenticação de grupos de usuários.

O conjunto de definições e características utilizadas no protocolo OpenHSM são:

- Uma vez inicializado, o HSM gera um par de chaves assimétricas kr_h e ku_h, possibilitando a emissão de um certificado auto-assinado c_h. Esta autoridade certificadora interna define o elemento central de confiança do HSM, a partir do qual todos os certificados de seus membros serão gerados;

- O protocolo define 3 diferentes grupos de usuários: administradores, auditores e operadores;

- Cada tipo de grupo suportado pelo HSM possui um sistema de armazenamento (DS) diferente. Sendo o dos administradores ADS, operadores ODS e auditores $AudDS$. Os dados enviados para esses sistemas são armazenados sem nenhum tipo adicional de criptografia;

- Cada grupo no HSM possui um identificador, sendo este único em grupos do mesmo tipo. Entretanto, existe apenas um grupo de administradores válido para um dado momento, com o identificador ADM sempre apontando para o grupo corrente.

- Todos os grupos são submetidos ao esquema de segredo compartilhado[42], onde a chave simétrica do grupo ks, é dividida em n partes das quais k são necessárias para recuperar ks. Os limiares seguem a seguinte regra: $1 \leq k \leq n$. Um conjunto de partes de um segredo é representado por Ks;

- Durante a criação de um grupo no HSM, cada membro recebe um par de chaves assimétricas, kr_i e ku_i, além de um certificado emitido pela AC interna do HSM,

c_i. Esse material é então armazenado no smartcard do membro e será utilizado novamente durante sua autenticação;

- Cada parte do segredo compartilhado do grupo, ks_i, passa a representar um de seus membros, sendo sua chave pública utilizada para cifrar sua parte do segredo antes que este seja armazenado no HSM. Este é o material que será decifrado durante o processo de autenticação do grupo;

- O sistema de armazenamento de dados não-exportáveis, NXD, é um sistema de armazenamento como qualquer outro, diferenciando-se somente porque na criação de uma cópia de segurança do HSM, os dados nele contidos não são copiados;

- A interação dos membros dos grupos com o HSM é feito através de um computador diretamente conectado, referenciado por hm.

Os protocolos do OpenHSM, dada sua complexidade e quantidade de passos, serão descritos aqui na forma de algoritmos. Esta mudança na sua forma de representação é justificada pelos comentários recebidos nas publicações citadas anteriormente, garantindo legibilidade e melhor entendimento.

O processo de criação de grupos do HSM permanece igual independente do seu tipo, sendo abordado aqui e referenciado posteriormente quando for necessário. A execução do algoritmo $createGroup$ requer alguns parâmetros, entre eles os dados sobre o novo grupo, como: identificador (id) e limiares (k e n), o sistema de armazenamento referente ao tipo de grupo a ser criado, DS, e por fim a chave privada e o certificado do HSM (kr_h e c_h).

A execução inicia com a geração da chave simétrica do grupo ks no passo 1. A chave é então submetida ao esquema de compartilhando de segredo, utilizando n como o número total de partes e k o número mínimo de partes para reconstruir ks, resultando no conjuntos de partes da chave Ks.

No passo 3, inicia-se o processo de criação dos membros, que deve ser executado n vezes, até que todos os membros do grupo estejam criados. No passo 4, o par

[1] Um sumário das descrições das variáveis e funções está disponível no apêndice A

Algoritmo createGroup(DS, id, k, n, c_h, kr_h) [1]

Cria um grupo utilizando o esquema de compartilhamento de segredo, gerando um par de chaves (kr_i e ku_i) e um certificado (c_i) para cada membro do grupo (s_i). Os certificados dos membros são assinados pela chave privada do HSM kr_h. O certificado de cada membro inclui seu nome e informações requisitadas durante o processo de criação dos membros. O token criptográfico de cada membro ct_{s_i} é utilizado para armazenar seus objetos criados no processo, c_i e kr_i. A chave simétrica do grupo ks, que foi compartilhada em n partes, é retornada como resultado do algoritmo.

1: $ks \leftarrow$ genSessionKey()
2: $Ks \leftarrow$ splitSecret(ks, k, n)
3: for ks_i in Ks do
4: $(kr_i, ku_i) \leftarrow$ genKeyPair()
5: $id_i \leftarrow$ load(hm, s_i)
6: $c_i \leftarrow$ genCert(id_i, ku_i, c_h, kr_h)
7: store(ct_{s_i}, kr_i, c_i, c_h)
8: $eks_i \leftarrow$ encrypt(ks_i, c_i)
9: store(DS, id, c_i, eks_i)
10: end for
11: store(DS, id, k, n)
12: return ks

de chaves do membro é gerado, kr_i e ku_i. Na seqüência, o nome do membro é solicitado à máquina hospedeira, informação que será utilizada para emitir o seu certificado c_i no passo 6. Uma vez que o certificado está gerado, kr_i, c_i e c_h pode ser armazenado no token criptográfico do membro.

Continuando, uma das partes do segredo do grupo ks_i é cifrada com o certificado do membro c_i (seu certificado possui sua chave pública). E finalmente, no passo 9, com todos os procedimentos para o membro finalizados, seus dados são armazenados em DS. Salientando que a chave privada do membro kr_i não é armazenada, preservando sua característica principal de privacidade.

No passo 11 os dados referentes ao grupo são armazenados. E finalmente, no passo 12, a chave simétrica do grupo é retornada.

Outro algoritmo que permanece igual independente do tipo (grupo) é a autenticação de um grupo $authenticGroup$. O processo requer o sistema de armazenamento e o identificador referentes ao grupo a ser autenticado.

O algoritmo $authenticGroup$ inicia carregando k do grupo identificado

Algoritmo authenticGroup(DS**,** id **)**

Autentica um grupo do HSM identificado por id no sistema de armazenamento DS. Os certificados dos membros c_i a serem autenticados são lidos de seus tokens criptográficos ct_{s_i}, identificando-o e possibilitando a decifragem da parte do segredo do grupo referente ao membro. É necessário que k membros sejam autenticados para recuperar o segredo do grupo ks.

1: $k \leftarrow$ load(DS, id)
2: for i = 1 to k do
3: $\quad c_i \leftarrow$ load(ct_{s_i})
4: $\quad eks_i \leftarrow$ load(DS, id, c_i)
5: $\quad u \leftarrow$ genSessionKey()
6: $\quad euks_i \leftarrow$ encrypt($u \mid\mid eks_i$, c_i)
7: $\quad eks_u \leftarrow$ ctDecrypt(ct_{s_i}, $euks_i$)
8: $\quad ks_i \leftarrow$ decrypt(eks_u, u)
9: end for
10: $ks \leftarrow$ joinSecret(Ks)
11: return ks

por id do sistema de armazenamento DS. No passo 2, inicia-se a iteração para autenticar os k membros do grupo, k foi definido no processo de sua criação e define o número mínimo de membros que precisam ser autenticados para permitir a reconstrução do segredo do grupo.

As operações entre os passos 2 e 9 serão realizadas k vezes, até ser possível recuperar o segredo do grupo. No passo 3 carrega-se do token criptográfico do membro seu certificado c_i, objeto necessário para, no passo 4, identificar no DS a parte do segredo cifrado do membro eks_i.

Os próximos dois passos, 4 e 5, são necessários para evitar o ataque de replay do Dolev-Yao[43], gerando u e recifrando seu valor concatenado a eks_i, resultando em $euks_i$. Este último então é enviado ao token criptográfico do membro para decifragem, liberado u e ks_i. Antes de retorná-los ao HSM, o token cifra ks_i com u. Como u é conhecido pelo HSM, o valor de ks_i é recuperado no passo 8.

Finalmente no passo 10, recupera-se a chave simétrica do grupo ks a partir do conjunto de partes decifradas nos passos anteriores. O algoritmo termina retornando o valor de ks.

6.2 Inicialização do HSM e criação do grupo de Administradores

Este é o primeiro passo na preparação de um ambiente para gerenciamento do ciclo de vida de chaves privadas. Este algoritmo irá realizar a criação da autoridade certificadora raiz interna e do grupo de administradores do HSM, estabelecendo, respectivamente, seu ponto de confiança e o grupo responsável por suas operarações administrativas.

O HSM, como apresentado nos aspectos gerais, possui apenas um grupo de administradores válido para um determinado momento no tempo, sendo que esse grupo pode ser alterado a qualquer momento utilizando o algoritmo $changeAdmGroup$ descrito na seção 6.7.

O algoritmo é disparado com a especificação dos valores de k e n, respectivamente o número mínimo de membros para autenticar o grupo e seu número total de membros.

Algoritmo createAdm(k**,** n **)** [1]

Inicializa o HSM, criando a AC interna que servirá de ponto de confiança para suas operações. O grupo de administradores também é criado.

1: kr_h, $ku_h \leftarrow$ genKeyPair()
2: $c_h \leftarrow$ genSelfSignedCert(kr_h, ku_h, id)
3: $ks_{ad} \leftarrow$ createGroup(ADS, ADM, k, n, c_h, kr_h)
4: $ekr_h \leftarrow$ encrypt(kr_h, ks_{ad})
5: store(ADS, c_h, ekr_h)
6: store(CTL, c_h)
7: return c_h

No primeiro passo, o par de chaves assimétricas do HSM, kr_h e ku_h, é gerado e a partir dele o certificado auto-assinado do HSM é emitido. Estes são os componentes necessários para criar o grupo de administradores no passo 3. Como já explicado anteriormente, a criação de um grupo do HSM é genérica e permanece a mesma para todos os tipos de grupos. A chave simétrica do grupo de administradores ks_{ad} é retornada como resultado. Essa chave é utilizada para cifrar a chave privada do HSM no passo 4, gerando ekr_h.

Até o passo 4, todos as operações necessárias para a inicialização e criação dos administradores foram finalizadas, restando apenas armazenar todos os componentes recém criados, o que permitirá a execução da próxima etapa do processo de tornar o HSM operacional, a criação de um grupo de auditores.

Então, no passo 5 o certificado do HSM e sua chave privada cifrada são armazenados no sistema de armazenamento de administradores e o passo 6 adiciona o certificado auto-assinado c_h na lista de certificados confiáveis CTL, estabelecendo o ponto de confiança do HSM.

Finalmente, c_h é retornado para a máquina hospedeira que iniciou a execução do algoritmo, o que identifica unicamente o HSM.

6.3 Criação de um grupo de Auditores

Apesar de não existir restrição criptográfica que impeça a criação de um grupo de operadores diretamente após a inicialização do HSM, o protocolo do OpenHSM define que a criação de um grupo de auditores seja obrigatoriamente o segundo passo para criar um HSM operacional, de forma a garantir que o processo de auditoria possa ser realizado antes mesmo de uma chave ser criada ou até utilizada.

Tratando-se de uma operação administrativa, o grupo de administradores do HSM é autenticado nos primeiros passos do algoritmo, permitindo a reconstrução de sua chave símetrica, possibilitando a decifragem da chave privada do HSM.

Cada grupo de auditores recebe um par de chaves assimétrica que será utilizado para assinar os registros do HSM que serão exportados pelo grupo. Outros pares de chaves não podem ser associados a um grupo de auditores, como acontece aos grupos de operadores, mas este é assunto das próximas seções.

O algoritmo requer os valores de k e n para o novo grupo de auditores, como acontece na criação do grupo de administradores, e um identificador para o grupo id, sendo que este deve identificar unicamente o grupo dentro do conjunto de grupos de auditores.

O algoritmo começa autenticando o grupo de administradores do sis-

Algoritmo createAudGroup(id, k, n)

Após a autenticação dos administradores do HSM, é criado um novo grupo de auditores identificado por id. A chave simétrica desse grupo é submetida ao mecanismo de compartilhamento de segredo, utilizando os limiares k e n. Adicionalmente, um par de chaves assimétricas é associado ao grupo recém criado.

1: $ks_{ad} \leftarrow$ authenticateGroup(ADS, ADM)
2: c_h, $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks_{ad})
4: $ks_{au} \leftarrow$ createGroup($AudDS$, id, k, n, c_h, kr_h)
5: kr_{au}, $ku_{au} \leftarrow$ genKeyPair()
6: $c_{au} \leftarrow$ genCert(id, ku_{au}, c_h, kr_h)
7: $ekr_{au} \leftarrow$ encrypt(kr_{au}, ks_{au})
8: store($AudDS$, id, c_{au}, ekr_{au})
9: return c_{au}

tema, especificando o sistema de armazenamento dos administradores ADS e o identificador do grupo ADM. Como definido nas premissas, o HSM possui apenas um grupo válido de administradores para um determinando periodo de tempo e o identificador ADM sempre aponta para o grupo atual.

O segundo e terceiro passo visam acessar a chave privada do HSM, carregando-a cifrada de ADS, decifrando em seguida com a chave simétrica do administradores. Portanto, os três primeiros passos dos algoritmos considerados operações administrativas serão os mesmos, autenticando o grupo de administradores e permitindo acesso a chave privada do HSM.

No passo 4, o grupo de auditores é criado utilizando o algoritmo $createGroup$ previamente definido, resultando na chave simétrica do novo grupo ks_{au}.

Logo após, um par de chaves criptográficas é gerado, sendo que um certificado utilizando-o será emitido pela AC interna do HSM no passo 6. Este certificado será útil para verificação da integridade dos registros exportados do HSM, através do algoritmo abordado na seção 6.10. Posteriormente, no passo 7, a chave privada dos auditores é cifrada com ks_{au}, garantindo confidencialidade da mesma.

Finalmente, o certificado e a chave privada cifrada do novo grupo podem ser armazenados no sistema de armazenamento de auditores, identificados por id, no passo 8. O certificado do grupo de auditores é retornado como resultado da execução do

algoritmo.

6.4 Criação de um grupo de Operadores

Os grupos de operadores no protocolo do OpenHSM detém a responsabilidade de liberação para uso das chaves gerenciadas de um HSM, podendo cada grupo ter nenhuma ou várias chaves associadas. Definido como uma tarefa administrativa, o processo de criação de um grupo de operadores requer a autenticação do grupo de administradores.

A peculiaridade neste tipo de grupo é a existência de um link com o grupo de administradores, habilitando este último a realizar operações sobre um grupo de operadores. Este link, por exemplo, permite aos administradores do HSM associar novas chaves ao grupo de operadores como também desassociá-las.

Os parâmetros necessários para criar um grupo de operadores são os mesmos requeridos no criação de um grupo de auditores, com o identificador do novo grupo id e os limiares para o esquema de segredo compartilhado k e n.

Algoritmo createOperGroup(id**,** k**,** n**)**

Cria um grupo de operadores, autenticando primeiramente o grupo de administradores. O ponto de confiança do novo grupo em relação aos administradores é estabelecido.

1: $ks_{ad} \leftarrow$ authenticateGroup(ADS, ADM)
2: c_h, $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks_{ad})
4: ks_o createGroup(ODS, id, k, n, c_h, kr_h)
5: $ks_{o*} \leftarrow$ genSessionKey()
6: $ks_{ao} \leftarrow$ xor(ks_o, ks_{o*})
7: $eks_{ao} \leftarrow$ encrypt(ks_{ao}, c_h)
8: store(ADS, id, eks_{ao})
9: store(NXD, id, ks_{o*})

Os três primeiros passos deste algoritmo são exatamente iguais ao processo de criação de auditores. Tem o objetivo de autenticar o grupo de administradores e permitir acesso à chave privada kr_h e ao certificado c_h do HSM.

No passo 4, o novo grupo de operadores é criado, com sua chave

simétrica, ks_o, sendo retornada como resultado da execução. Essa chave será utilizada para cifrar as chaves que forem associadas com esse grupo de operadores.

Os passos seguintes, de 5 a 9, servem para criar o ponto de confiança dos operadores nos administradores, permitindo a realização de operações administrativas sobre os mesmo. Inicia-se com a geração de uma segunda chave simétrica para o grupo de operadores ks_{o*}. No passo 6, realiza-se a operação ou-exclusivo (XOR) entre as duas chaves simétricas do grupo de operadores, resultado em ks_{ao}. Uma das propriedades da função de XOR é sua reversibilidade, isto é, $A \oplus B = C$ e $C \oplus B = A$. Portando, se realizar a operação $ks_{o*} \oplus ks_{ad}$, o resultado será ks_o. Desta forma, o grupo de administradores pode reconstruir ks_o tendo acesso apenas a ks_{o*} e ks_{ad}.

No passo 7, o resultado da operação XOR ks_{ad} é cifrado com a chave pública do HSM, garantindo que só quem possui acesso a chave privada do HSM, no caso o grupo de administradores, poderá ter acesso a mesma. E por fim, os dados sensíveis para futuras execuções são armazenadas nos devidos sistemas de armazenamento, passos 8 e 9.

O valor de ks_{o*} é armazenado em um sistema de armazenamento diferente do utilizado para os dados dos grupo de operadores. A utilização do NXD garante uma característica importante ao protocolo do OpenHSM, a rastreabilidade das chaves gerenciadas. Durante o procedimento de backup do HSM, coberto no capítulo 7, o NXD não é copiado, resultando que, uma vez recuperado o backup, NXD permaneça vazio. Sem a existência do ponto de confiança dos operadores em relação aos administradores, não se pode realizar operações administrativas sobre os mesmos.

Por isso que, logo após a recuperação do backup, o grupo de administradores, juntamente com cada grupo de operadores, deverão realizar o procedimento de re-criação do ponto de confiança entre os mesmos (seção 6.9). Uma vez obrigatória a intervenção do grupo de operadores, garante-se que uma chave gerenciada nunca será utilizada ou terá seu responsável trocado sem prévio conhecimento.

6.5 Criação de Chave Gerenciada

A criação de chaves gerenciadas é uma das operações mais importante em um HSM e pode ser executada logo após a criação do primeiro grupo de operadores. O grupo de administradores precisa ser autenticado e deve existir o ponto de confiança entre os administradores e o grupo de operadores que será responsável pela nova chave.

As chaves, como acontece nos grupos de operadores e auditores, possuem um identificador único dentro no conjunto de chaves gerenciadas, representado por id_k. Este identificador será utilizado posteriormente quando o grupo de operadores carregar a chave para uso ou quando uma aplicação externa realizar operação criptográficas com a chave.

Além do identificador a ser atribuído a nova chave, o algoritmo de criação de chaves gerenciadas precisa do identificador do grupo de operadores que será responsável pela chave id_o e, também, as características da nova chave, tais como algoritmo e tamanho.

Algoritmo createManagedKey(id_k, key_params, id_o)

Cria um par de chaves, identificada por id_k, para ser gerenciada pelo HSM. Essa nova chave terá as características definidas por key_params (algoritmo e tamanho) e será associada ao grupo de operadores identificado por id_o.

1: $ks_{ad} \leftarrow$ authenticateGroup(ADS, ADM)
2: c_h, $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks_{ad})
4: $eks_{ao} \leftarrow$ load(ADS, id_o)
5: $ks_{ao} \leftarrow$ decrypt(eks_{ao}, kr_h)
6: $ks_{o*} \leftarrow$ load(NXD, id_o)
7: $ks_o \leftarrow$ xor(ks_{ao}, ks_{o*})
8: kr, $ku \leftarrow$ genKeyPair(key_params)
9: $ekr \leftarrow$ encrypt(kr, ks_o)
10: store(KDS, id_k, ekr, ku, id_o)
11: return ku

O algoritmo inicia autenticando o grupo de administradores, com o objetivo de obter acesso ao certificado e chave privada do HSM nos três primeiros passos. Estes passos já foram descritos na criação de grupo de auditores (vide seção 6.3).

No passo 4, baseado no identificador do grupo de operadores id_o,

carrega-se o resultado da operação XOR cifrada do sistema de armazenamento dos administradores. Com a chave privada do HSM já aberta, pode-se decifrar eks_{ao} para ter acesso a ks_{ao}. Logo após carrega-se a segunda chave simétrica do grupo de operadores, ks_{o*}, identificado por id_o. Lembrando que ks_{o*} estabelece o ponto de confiança entre o grupo de adminitradores e o grupo de operadores.

No passo 7, se faz uso da propriedade de reversibilidade da operação XOR e recupera-se a chave simétrica do grupo de operadores ks_o. Esta é a chave que será utilizada no passo 9 para cifrar a nova chave gerenciada. O par de chaves assimétricas é gerado no passo 8 e então armazenado no KDS no passo 10, juntamente com id_k e id_o. Finalmente, a chave pública recém criada é retornada como resultado do algoritmo.

6.6 Liberando uma Chave Gerenciada para Uso

O gerenciamento do ciclo de vida de chaves privadas deve permitir em algum momento a utilização destas chaves. A partir do algoritmo detalhado nesta seção, é possível ao grupo de operadores responsável por uma chave liberá-la para uso por aplicações externas.

O HSM autentica o grupo de operadores responsável pela chave, deixando-a carregada na memória volátil do sistema sob uma política de uso explicitamente definida.

Essa política pode ser a quantidade máxima de usos da chave e/ou um período de tempo determinado. Por exemplo, pode-se carregar uma chave para 5 usos em um prazo de 5 minutos, sendo que a chave é descarregada assim que a primeira restrição expirar.

O algoritmo para liberação de uma chave requer o identificador da chave id_{key} e a política sob a qual a chave irá operar, $policy$.

O algoritmo carrega no passo 1 os dados da chave id_{key} do sistema de armazenamento de chaves KDS. Entre os dados carregados estão o identificador do grupo de operadores responsável pela chave, id_{oper}, e o par de chaves propriamente dito, ku e ekr, sendo que a chave privada encontra-se cifrada.

Algoritmo loadManagedKey(id_{key}**,** $policy$ **)**

Carrega uma chave gerenciada identificada por id_{key} no HSM. Primeiramente, o grupo de operadores responsável pela chave é autenticado. O parâmetro $policy$ definirá a política de uso da chave.
1: ekr, ku, $id_{oper} \leftarrow$ load(KDS, id_{key})
2: $ks \leftarrow$ authGroup(ODS, id_{oper})
3: $kr \leftarrow$ decrypt(ekr, ks)
4: manageKey(id_{key}, kr, ku, $policy$)
5: return(ku)

No passo 2 autentica-se o grupo de operadores, obtendo-se acesso a sua chave simétrica ks, que será utilizada no passo 3 para decifrar ekr.

A chave privada, kr, é então submetida ao controlador de chaves carregadas, definindo id_{key}, a política definida pelos operadores, ku e kr. Esta chave permanecerá disponível enquanto sua política de uso permitir ou o procedimento de descarga da mesma for acionado.

6.7 Troca do grupo de Administratores

O grupo de administradores de um HSM pode ser trocado utilizando o algoritmo coberto nesta seção. Essa operação, como qualquer outra de troca de grupos, deve ser realizada sempre que houver comprometimento do grupo. Considera-se um grupo comprometido quando o uso do mecanismo de compartilhamento de segredo for inviabilizado.

As chaves simétricas dos grupos são compartilhadas utilizando os limiares k e n, sendo k o número mínimo de membros presentes para autenticar o grupo e n o número total de membros. Sabendo disso, para um grupo de 5 membros onde 3 precisam estar presentes, a existência de qualquer 2 membros não é capaz de recuperar nenhum bit da chave. Portanto, se dois membros do grupo perderem de alguma forma acesso a seus token criptográficos, e conseqüentemente acesso as suas chaves privadas, o grupo ainda pode ser autenticado, porém, se mais um tiver problema, o grupo está comprometido.

Considerando-se que um HSM só possui um grupo de administradores válido, a sua troca se torna peça fundamental para a continuidade do ciclo de vida das

chaves gerenciadas do HSM, já que o grupo de administradores também é utilizado para recuperação das cópias de segurança. Este processo autentica o grupo de administradores atual e requer os limiares k e n que serão utilizados para criação do novo grupo.

Algoritmo changeAdmGroup(k**,** n **)**

Troca o grupo de administradores de um HSM, autenticando o atual e gerando o novo. Depois de transferida a administração do HSM, a chave privada do HSM passa a ser protegida pela chave simétrica do novo grupo ks_2 e os certificados de todos os membros do grupo antigo são revogados.

1: $ks_1 \leftarrow$ authenticateGroup(ADS, ADM)
2: c_h, $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks_1)
4: $ks_2 \leftarrow$ createGroup(ADS, ADM_2, k, n, c_h, kh_h)
5: $ekr_h \leftarrow$ encrypt(kr_h, ks_2)
6: store(ADS, ekr_h)
7: revokeGroupCerts(ADS, ADM)
8: $ADM \leftarrow ADM_2$
9: return c_h

O algoritmo $changeAdmGroup$ inicia como qualquer outra operação administrativa já vista até aqui, onde os 3 primeiros passos visam autenticar o grupo de administradores e liberando acesso a chave privada do HSM, kr_h.

No passo 4, o novo grupo de administradores é criado, referenciado pelo identificador temporário ADM_2. Esse identificador será posteriormente atribuído a ADM, que sempre aponta para o grupo de administradores corrente do HSM.

A chave privada do HSM é então cifrada utilizando a chave simétrica do novo grupo de administradores ks_2 no passo 5 e logo após armazenada em ADS. No passo 7, os certificados de todos os membros do antigo grupo de administradores são revogados, garantindo que não serão mais aceitos no HSM. Seguindo, é atualizado o identificador do grupo de administradores, ADM, para apontar para o novo grupo ADM_2.

Finalmente, com o grupo de administradores trocado, o certificado do HSM é retornado no passo 9. Como visto, essa é uma operação crítica e requer bastante atenção, já que um HSM possui apenas um grupo de administradores válido. Deve-se garantir atomicidade na execução dos passos 6, 7 e 8, já que se falharem destes pontos, o HSM possuirá um grupo de administradores inválido.

Uma característica interessante após a execução deste algoritmo é que, todas as cópias de segurança criadas antes desta execução são apenas recuperáveis com a autenticação do antigo grupo de administradores do HSM. A troca proposta por este algoritmo só irá se refletir em novas cópias de segurança criadas. O grupo de auditores desempenham um papel fundamental neste processo, controlando as atividades do antigo grupo de administradores em relação a recuperação de cópias de segurança. Alternativamente, os smartcards não mais necessários podem ser destruídos.

6.8 Alterando os responsáveis por uma chave gerenciada

A troca dos responsáveis de uma chave no HSM não é uma operação tão crítica quando a troca de seu grupo de administradores, porém, tão importante quanto. Isso porque uma vez que o ponto de confiança do grupo atualmente responsável e o grupo que receberá a responsabilidade existam, os administradores podem realizar a troca.

Esta operação pode representar em uma situação real, a troca da diretoria responsável por uma AC em uma instituição, permitindo que seus novos membros passem a ser responsáveis pelo uso de sua chave privada. Como dito anteriormente, somente o grupo de administradores é necessário para essa troca, desde que os pontos de confiança existam.

Basicamente, o algoritmo autentica o grupo de administradores e através do ponto de confiança recupera a chave simétrica dos dois grupos de operadores, podendo então decifrar a chave privada com a chave simétrica de um grupo, posteriormente cifrando com a do outro. Para iniciar o processo de troca, são necessários o identificador da chave que terá seu grupo responsável trocado, id_{key}, e o identificador do grupo de operadores que se tornará seu novo responsável, id_{o2}.

O algoritmo $changeKeyOwner$ inicia como qualquer outra operação administrativa já vista até aqui, onde os 3 primeiros passos visam autenticar o grupo de administradores e liberando acesso a chave privada do HSM, kr_h.

No passo 4, os dados da chave id_{key} são carregados de KDS, tais como a chave privada cifrada, ekr, e o identificador do atual responsável pela chave, id_{o1}.

Algorithm changeKeyOwner(id_{key}**,** id_{o2} **)**

Troca o grupo de operadores responsável por uma chave gerenciada no HSM. É necessária a autenticação do grupo de administradores além de existir o ponto de confiança dos dois grupos de operadores envolvidos na operação.

1: $ks \leftarrow$ authGroup(ADS, ADM)
2: $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks)
4: ekr, $id_{o1} \leftarrow$ load(KDS, id_{key})
5: $eks_{ao1} \leftarrow$ load(ADS, id_{o1})
6: $ks_{ao1} \leftarrow$ decrypt(eks_{ao1}, kr_h)
7: $ks_{o1*} \leftarrow$ load(NXD, id_{oper1})
8: $ks_{o1} \leftarrow$ xor(ks_{ao}, ks_{o1*})
9: $kr \leftarrow$ decrypt(ekr, ks_{o1})
10: $eks_{ao2} \leftarrow$ load(ADS, id_{o2})
11: $ks_{ao2} \leftarrow$ decrypt(eks_{ao2}, kr_h)
12: $ks_{o2*} \leftarrow$ load(NXD, id_{oper2})
13: $ks_{o2} \leftarrow$ xor(ks_{ao}, ks_{o2*})
14: $ekr \leftarrow$ encrypt(kr, ks_{o2})
15: store(KDS, id_{key}, ekr, id_{o2})

Os passos de 5 a 8, como os passos de 10 a 13, visam recuperar chaves simétricas de grupos de operadores, identificados por id_{o1} e id_{o2} respectivamente. Estas operações já foram cobertas anteriormente (vide seção 6.5). Ao final do passo 8, obtem-se acesso a ks_{o1} e pode-se decifrar a chave privada gerenciada no passo 9. E ao final do passo 13, obtém-se acesso a ks_{o2}, que é utilizada no passo 14 para cifrar a chave gerenciada.

Finalmente, a chave kr cifrada pode ser armazenada juntamente com o identificador do seu novo responsável no passo 15, concluindo o algoritmo de troca do grupo responsável por uma chave gerenciada.

6.9 Criando o ponto de confiança de um grupo de Operadores em relação aos Administradores

O ponto de confiança do grupo de operadores em relação ao grupo de administradores garante que operações administrativas sejam realizadas sobre o primeiro. Este ponto de confiança deixa de existir quando os administradores recuperam o backup

de um HSM, estabelecendo outro ambiente operacional.

O algoritmo $activatesAdmTasksOverOperGroup$ autentica o grupo de administradores e o grupo de operadores a ser "ativado", regerando a segunda chave do grupo de operadores e aplicando a operação XOR à chave principal do grupo. O idenficador do grupo de operadores é o único requisito para disparar a execução deste algoritmo.

Algoritmo activatesAdmTasksOverOperGroup(id_{oper} **)**

Estabelece o ponto de confiança de um grupo de operadores identificado por id_{oper}, em relação ao grupo de administradores. A existência deste ponto possibilita que o último, utilizando a propriedade de reversibilidade do XOR, possa recuperar a chave simétrica do grupo de operadores ks_o, permitindo assim, a associação e/ou desassociação de chaves gerenciadas.

1: $ks_o \leftarrow$ authGroup(ODS, id_{oper})
2: $ks_{o*} \leftarrow$ genSessionKey()
3: $ks_{ao} \leftarrow$ xor(ks_o, ks_{o*})
4: $c_h \leftarrow$ load(ADS)
5: $eks_{ao} \leftarrow$ encrypt(ks_{ao}, c_h)
6: store(ADS, id_{oper}, eks_{ao})
7: store(NXD, id_{oper}, ks_{o*})

O algoritmo inicia autenticando o grupo de operadores identificado por id_{oper}, obtendo acesso sua chave simétrica, ks_o. No passo 2, uma nova chave simétrica secundária do grupo, ks_{o*}, é gerada, permitindo a realização da operação XOR no passo 3, que resultará em ks_{ao}.

No passo 4, o certificado do HSM é carregado do ADS e posteriormente utilizado para cifrar ks_{ao}, resultado em eks_{ao}. Nos passos 6 e 7, os dados gerados durante a execução do algoritmo, eks_{ao} e ks_{o*}, são gravados nos devidos sistemas de armazenamento, ADS e NXD respectivamente, reestabelecendo o ponto de confiança.

6.10 Exportação dos Registros de Atividades

O protocolo do OpenHSM define que todas as operações realizadas em um HSM devem ser registradas de forma sequencial para posterior análise por grupos de auditores, permitindo que seja reconstruído toda a trilha das operações realizadas.

Apenas grupos de auditores podem exportar os registros de atividades do HSM, sempre assinados, devendo fazer isso informando seu identificador id e, opcionalmente, especificar um período de tempo para exportação. Se não especificado, todos os registros existentes são exportados.

Algoritmo exportLog(id **[,** $rangeDate$**])**

Permite um grupo de auditores exportar os registros de atividade do HSM assinados, podendo ou não definir o período de tempo para os mesmos

1: $ks \leftarrow$ authGroup($AudDS$, id)
2: $ekr \leftarrow$ load($AudDS$, id)
3: $kr \leftarrow$ decrypt(ekr, ks)
4: $L \leftarrow$ load(LDS, $rangeDate$)
5: $sL \leftarrow$ sign(L, kr)
6: return(sL)

O algoritmo $exportLog$ inicia autenticando o grupo de auditores identificado por id, resultando na sua chave simétrica, ks. No passo 2, carrega-se a chave privada cifrada do grupo, ekr, que, no passo seguinte, é decifrada, resultando em kr.

No passo 4, o HSM carrega os registros, L, do sistema de armazenamento de registros, LDS, respeitando o período de tempo especificado, $rangeDate$. O HSM então assina L para garantir a integridade destes registros. Os registros assinados são retornados como resultado do algoritmo.

6.11 Limpeza dos Registros de Atividades

O ciclo de vida de um HSM dura vários anos e considerando o fato do mesmo rodar em ambiente embarcado, com recursos de armazenamento limitado, definiu-se um operação administrativa para apagar os registros do sistema.

Este procedimento de limpeza dos registros de atividades do HSM engloba o algoritmo para exportar registros também, garantindo que qualquer registro que for apagado do HSM, já tenha sido exportado pelo menos uma vez, permitindo assim que a criação da trilha de auditoria cubra todas as operações realizadas em todo o tempo de vida de um HSM.

O algoritmo autentica o grupo de administradores e um grupo de auditores, identificado por id. Opcionalmente, pode-se informar um período de tempo no qual os registros serão exportados e apagados.

Algoritmo eraseLog(id **[,** $rangeDate$**])**

Apaga os registros de atividades do HSM, autenticando o grupo de adminitradores e um grupo de auditores. Este último, recebe uma cópia assinada dos registros apagados.

1: authGroup(ADS, ADM)
2: $sL \leftarrow$ exportLog(id, $rangeDate$)
3: delete(LDS, $rangeDate$)
4: return(sL)

O algoritmo $eraseLog$ inicia autenticando o grupo de administradores do HSM no passo 1. Essa autenticação é requirida apenas por questão de segurança, já que o conjunto dos registros do sistema é parte fundamental para o controle do ciclo de vida das chaves gerenciadas.

No passo 2 executa-se o algoritmo $exportLog$ visto anteriormente, que irá resultar nos registros assinado do HSM, sL, referente ao tempo especificado, $rangeDate$. No passo seguinte, os registros recém exportados podem ser apagados. O HSM retorna sL como resultado da execução.

Os protocolos do OpenHSM abordados até aqui descrevem a construção de um ambiente seguro para gerenciamento de chaves criptográficas de um HSM. Porém, a continuidade do ciclo de vida das chaves gerenciadas está comprometida em caso de falha de hardware ou de desastres, o que pode causar danos irreparáveis a uma infraestrutura de chaves públicas. Essa lacuna será coberta na próxima seção, que abordará a criação e restauração de cópias de segurança de um HSM de forma segura e confiável.

6.12 Conclusão

Este capítulo apresentou todos os sub-protocolos do OpenHSM usados na operacionalização de um HSM. Através deles é possível a realização de dois das principais operações em um HSM: criação e utilização de chaves criptográficas.

Entretanto, duas etapas essencial para o completo ciclo de vida de chaves criptográficas ainda não foram cobertos, a criação e recuperação de cópias de segurança (cópias das chaves gerenciadas). Os sub-protocolos responsáveis por estas etapas serão abordados no próximo capítulo.

Capítulo 7

OpenHSM - Cópias de Segurança

Os protocolos de criação de cópias de segurança do conteúdo de um HSM de forma segura e confiável, devem permitir sua posterior recuperação, de forma a restaurar todo o ambiente operacional que existia no momento que a cópia de segurança foi criada.

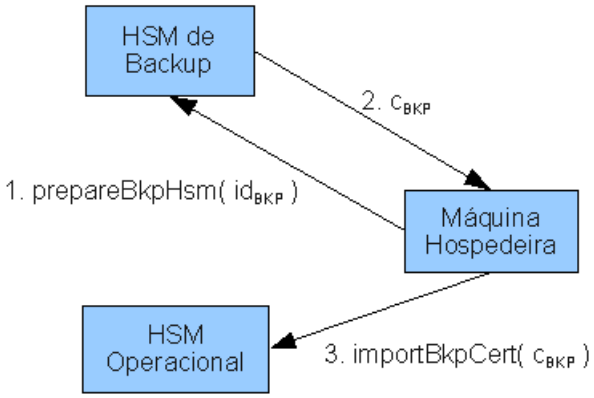

Figura 7.1: Processo de criação de um HSM de backup

A criação destas cópias, como pode ser visto na figura 7.1, requer a utilização de um segundo módulo, que deverá ser configurado como uma unidade de backup (seção 7.1), não podendo ser utilizado para nenhum outro fim, a não ser restaurar uma cópia de segurança de um HSM. Entretanto, uma unidade de backup pode servir de receptor de backup para vários HSM ao mesmo tempo.

Após a preparação de um HSM para backup, o certificado c_{bkp} deve ser importado no HSM operacional (seção 7.2). Este processo permitirá que cópias de segurança sejam criadas (seção 7.3) para posterior recuperação em caso de algum problema acontecer com o HSM operacional (seção 7.4).

Por fim, a seção 7.5 discute a questão do controle de ciclo de vida de múltiplas cópias da mesma chave criptográfica em hardware criptográficos diferentes.

7.1 Preparando um HSM para ser uma unidade de backup

O primeiro passo para criação de cópias de segurança de um módulo de segurança criptográfico é preparar um HSM adicional para ser uma unidade de backup, sendo que qualquer HSM que não tenha sido inicializado pode ser utilizado neste processo.

Nenhuma informação dos módulos criptográficos ativos é necessária para a criação de uma unidade de backup, que pode ser criado antes mesmo de existir um HSM operacional.

O algoritmo $prepareBkpUnit$ implementa o protocolo que prepara um HSM para ser uma unidade de backup, requerendo como parâmetro o identificador id_{bkp} que será utilizado como nome comum do certificado desta unidade.

Algoritmo prepareBkpUnit(id_{bkp} **)**

Prepara um HSM para ser uma unidade de backup, gerando um par de chaves assimétricas kr_{bkp} e ku_{bkp} e emitindo um certificado auto-assinado a partir deste par de chaves e o identificador do HSM id_{bkp} passado como parâmetro. O certificado do HSM de backup é exportado como resultado do algoritmo.

1: kr_{bkp}, ku_{bkp} ← genKeyPair()
2: c_{bkp} ← genSelfSignedCert(kr_{bkp}, ku_{bkp}, id_{bkp})
3: store(BDS, kr_{bkp}, c_{bkp})
4: return c_{bkp}

O processo de preparação de uma unidade de backup inicia com a criação de um par de chaves criptográficas de backup (kr_{bkp} e ku_{bkp}). No passo 2, um

certificado auto-assinado c_{bkp} é gerado a partir do identificador id_{bkp} e o par de chaves recém criado.

No passo 3, o certificado e a chave privada de backup são armazenados no sistema de armazenamento de dados de backup para que possam ser carregados posteriormente no processo de recuperação de uma cópia de segurança. O algoritmo termina exportando c_{bkp}, que deverá ser importado nos HSM operacionais para iniciar o processo de geração de cópias de segurança.

A chave privada de backup kr_{bkp} é armazenada em texto claro com proteção do perímetro criptográfico do HSM, que é capaz de detectar, com a utilização de sensores, e responder a uma tentativa de invasão, apagando dados sensíveis a segurança do módulo.

Uma vez preparado para ser um receptor de cópias de segurança, o HSM deve ser armazenado em local seguro, aguardando a necessidade de recuperação de um backup.

7.2 Importando o Certificado de Backup em HSM Operacional

Como visto na seção anterior, a preparação de um HSM receptores de backup inclui a exportação de um certificado. Este certificado deve ser então importado nos HSM operacionais, possibilitando a criação de cópias de segurança do mesmo.

O processo de importação de certificados de backup em um HSM é simples, mas requer muito cuidado. Por se tratar de um certificado X.509 auto-assinado, que pode ser gerado em qualquer lugar, o grupo de administradores deve estar ciente da sua procedência, cabendo aos grupos de auditores a análise dos registros de atividades do HSM, a fim de descobrir eventuais falhas de gerenciamento.

Um HSM pode importar vários certificados de backup, aumentando o número de unidades onde uma cópia de segurança pode ser recuperada. O algoritmo $importBkpCert$ implementa o protocolo de importação destes certificados, autenticando

o grupo de administradores do HSM para isso.

Algoritmo importBkpCert(c_{bkp} **)**

Importa o certificado exportado no processo de criação de uma unidade de backup.
O grupo de administrador é autenticado na execução deste algoritmo.
1: $ks \leftarrow$ authenticateGroup(ADS, ADM)
2: $ekr_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks)
4: $sc_{bkp} \leftarrow$ sign(c_{bkp}, kr_h)
5: store(BDS, sc_{bkp})

O algoritmo $importBkpCert$ inicia realizando a autenticação do grupo de adminitradores, possibilitando a liberação da chave privada do HSM no passo 3. No passo 4, o certificado de backup passado como parâmetro é reassinado com kr_h e posteriormente armazenado no sistema de armazenamento de backup.

Após a importação do primeiro certificado de backup, cópias de segurança do ambiente operacional do HSM podem ser criadas. Este é o assunto da próxima seção.

7.3 Backup

A geração de cópias de segurança do ambiente operacional de um HSM é uma atividade essencial para continuidade do ciclo de vida das chaves gerenciadas e deve ser executada regularmente, já que qualquer operação executada após a criação do backup será completamente perdida em caso de falha do HSM. As declarações de práticas de certificação das ICPs devem estabelecer a regularidade na qual as cópias de segurança devem ser geradas.

Todas as chaves gerenciadas pelo módulo são exportadas de forma cifrada, garantindo que, mesmo fora do perímetro criptográfico do HSM, elas estejam seguras, seguindo os requisitos existentes nas normas FIPS PUB 140-2 e MCT-7.

O algoritmo $bkpHsm$, que implementa o protocolo de criação de uma cópia de segurança de um HSM, autentica o grupo de administradores, por se tratar de uma operação administrativa. Além disso, o HSM deve possuir pelo menos um grupo

de auditores, que serão autenticados durante a recuperação da cópia de segurança, garantindo que não existam cópias paralelas do HSM sem prévio conhecimento da equipe de auditoria.

Algoritmo bkpHsm()

Cria um cópia de segurança de um HSM operacional, autenticando seu grupo de admistradores. A cópia de segurança poderá ser recuperada em todos as unidades de backup que tiveram seu certificado importado no HSM. Lembrando que o sistema de armazenamento de dados não exportáveis NXD, como o nome já diz, não é copiado.

1: $ks \leftarrow$ authenticateGroup(ADS, ADM)
2: ekr_h, $c_h \leftarrow$ load(ADS)
3: $kr_h \leftarrow$ decrypt(ekr_h, ks)
4: store(BPF, load(CTL))
5: store(BPF, load(ADS))
6: store(BPF, load($AudDS$))
7: store(BPF, load(ODS))
8: store(BPF, load(KDS))
9: store(BPF, load(BDS))
10: $C_{bkp} \leftarrow$ load(BDS))
11: $ks_{bkp} \leftarrow$ genSessionKey()
12: $eBPF \leftarrow$ encrypt(BPF, ks_{bkp})
13: $seBPK \leftarrow$ sign($eBPK$, kr_h)
14: for c_{bkp_i} in C_{bkp} do
15: $\quad eks_{bkp} \leftarrow$ encrypt(ks_{bkp}, c_{bkp_i})
16: end for
17: return $seBKP$, EKs_{bkp}

O algoritmo inicia com os três passos comuns para uma operação administrativa, autenticando os administradores e obtendo acesso a chave privada e o certificado do HSM, kr_h e c_h respectivamente.

Nos passos de 4 a 9, todos os dados relevantes ao ambiente operacional do HSM são copiados para o pacote do backup BPF, incluindo os sistemas de armazenamento de dados dos grupos (ADS, ODS, $AudDS$), de chaves gerenciadas KDS, de certificados confiáveis CTL e de dados de backup BDS. O último é necessário porque mais de uma certificado de backup pode ter sido importado no HSM, permitindo que mesmo após recuperação da cópia de segurança, novos backups possam ser criados.

O sistema de armazenamento de dados não exportáveis não é copiado durante este algoritmo, desabilitando operações administrativas sobre chaves gerenciadas

enquanto seu grupo de operadores responsável não tenha conhecimento de sua existência.

O BPF então precisa ser cifrado utilizando todos os certificados de backup previamente importados no HSM, permitindo que a cópia possa ser recuperada em qualquer um deles. No passo 10, o conjunto de certificados de backup é carregado. Logo após, gera-se uma chave simétrica, ks_{bkp}, que, no passo 12, será utilizada para cifrar BPF. O passo 13 visa garantir a integridade do cópia de segurança, assinando $eBPF$ com a chave privada do HSM kr_{bkp}.

Nos passos 14 a 16, realiza-se uma iteração por todos os certificados de backup existentes dentro do HSM, utilizando cada um deles para cifrar ks_{bkp}, que irá resultar em um conjunto de chaves simétricas cifradas com diferentes chaves públicas, EKs_{bkp}.

O processo de criação do backup está completo ao final do passo 16, retornando a cópia de segurança do ambiente operacional cifrado e assinado e o conjunto de chaves simétricas cifradas pelos certificados de backup, $seBKP$ e EKs_{bkp} respectivamente.

7.4 Recuperação do Backup

A recuperação de cópias de segurança é último passo para permitir o completo ciclo de vida de chaves criptográficas gerenciadas em um HSM, incluindo os casos de falhas de hardware ou desastres. Os certificados de backup existentes na cópia de segurança delimitam o conjunto de unidades de backup que podem ser utilizadas para recuperação.

Esta recuperação, além de ser um operação administrativa, requer a autenticação de um grupo de auditores, permitindo o conhecimento da equipe de auditoria que uma nova instância do HSM está operacional, diminuindo a carga de responsabilidade do grupo de administradores.

O algoritmo $recoverBkp$ requer como parâmetros de entrada o identificador do grupo de auditores que será autenticado durante o processo e os dados retornados do algoritmo de criação do backup, $seBKP$ e EKs_{bkp}, respectivamente o pacote de

backup cifrado e assinado e a chave simétrica que cifra o pacote de backup cifrada com cada um dos certificados de backup importados no HSM.

Algoritmo recoverBkp($seBKP$**,** EKs_{bkp}**,** id_{audit}**)**

Restaura o ambiente operacional de um HSM em um módulo previamente preparado para backup. O grupo de administradores e pelo menos um grupo de auditores são autenticados neste processo, sendo estas autenticações somente requeridas por questão de segurança, isto é, não têm objetivos criptográficos.

1: $kr_{bkp} \leftarrow$ load(BDS)
2: $ks_{bkp} \leftarrow$ decrypt(eks_{bkp}, kr_{bkp})
3: $BPF \leftarrow$ decrypt($seBPF$, ks_{bkp})
4: store(CTL, load(BPF))
5: store(ADS, load(BPF))
6: $c_h \leftarrow$ load(ADS)
7: verify($seBKP$, c_h)
8: $ks_{adm} \leftarrow$ authenticateGroup(ADS, ADM)
9: store($AudDS$, load(BPF))
10: $ks_{audit} \leftarrow$ authenticateGroup($AudDS$, id_{audit})
11: store(ODS, load(BPF))
12: store(KDS, load(BPF))
13: store(BDS, load(BPF))

O processo de recuperação de backup inicia carregando a chave privada de backup kr_{bkp} do sistema de armazenamento de dados de backup BDS. Logo após, a chave simétrica que cifra o backup, ks_{bkp} é decifrada. A chave liberada no passo 2 permite que o pacote de backup seja decifrado no passo 3.

Os passos 4 e 5 restauram, respectivamente, o sistema de armazenamento de certificados confiáveis CTL e de administradores ADS. Este é o conjunto mínimo de dados necessário para verificar a integridade de $seBKP$ no passo 7 e autenticar o grupo de administradores no passo 8.

O passo 9 restaura o sistema de armazenamento de auditores, que permite a autenticação do grupo de auditores identificado por id_{audit} no passo 10.

E finalmente, nos passos 11, 12 e 13, os sistemas de armazenamento restantes são restaurados, respectivamente, os sistemas de armazenamento de dados de operadores, chaves e backup.

O ambiente operacional do HSM, uma vez restaurado, será exatamente igual ao momento em que a cópia de segurança foi criada, com exceção do sistema de ar-

mazenamento de dados não exportados, NXD, que precisa ser recriado com o algoritmo apresentado na seção 6.9.

7.5 Utilização de Múltiplos Ambientes Operacionais

À medida que o número de instâncias de chaves criptográficas operacionais aumenta, maior é a dificuldade para gerenciá-las, tornando mais complexo também o processo de auditoria, já que, esta paralelização das operações, difunde os usos de uma chave criptográfica em vários ambientes.

Apesar disso, alguns situações requerem este tipo de abordagem, isto é, a criação de vários ambientes operacionais, com o objetivo de rapidez no processo de retomada das atividades em caso de falha do ambiente principal ou em ambiente de alta demanda, com necessidade de balanceamento de carga.

Muitas vezes, a realização de uma cerimônia implica em custos financeiros e/ou logísticos elevados em algumas organizações, se considerar a quantidade de pessoas envolvidas e a dificuldade de reuní-las para uma nova tentativa. Por isso, para garantir rapidez em caso de falha do ambiente operacional principal, pode-se configurar um segundo ambiente idêntico ao primeiro, diminuindo assim os riscos de falha de todo o processo, evitando que uma nova cerimônia seja necessária.

A criação de uma ambiente operacional idêntico pode ser realizado utilizando os mecanismos de criação e recuperação de cópias de segurança do HSM. O novo HSM, uma vez restaurado, terá os mesmos grupos e chaves gerenciadas que existiam no primeiro. Assim, no caso de uma autoridade certificadora por exemplo, duas cópias da sua chave privada estão operacionais, bastando que o grupo de operadores responsável a libere para uso.

Entretando, como forma de diminuir ainda mais os riscos de falhas que resultem no cancelamento de uma cerimônia, pode-se criar ambientes totalmente independentes, isto é, com a troca dos grupos de administradores e operadores do módulo, sendo que os novos grupos podem ser compostos dos mesmos membros e possuir a mesma configuração dos grupos originais. Desta forma, é possível contornar até falhas nos smart-

cards dos membros dos grupos. Esta abordagem também se aplica em casos onde deseja-se ambientes operacionais paralelos em localizações completamente diferentes umas das outras.

A impotência dos grupos de administradores e operadores em relação aos grupos de auditores de um HSM permite que estes últimos realizem auditoria de qualquer uma das instâncias dos módulos operacionais, desde que tenham sido criados durante a preparação da primeira instância do HSM. Se novos grupos de auditores forem criados após a existência de múltiplas instâncias, estes só podem auditar o HSM em que foram criados e as novas instâncias que foram recuperadas de cópias de segurança criadas posteriormente a sua criação.

Além disso, digamos que um HSM operacional possua um grupo de administradores, um de operadores, um de auditores e gerencia uma chave criptográfica. O processo de criação de cópias de segurança deste ambiente requer a autenticação do grupo de administradores. A sua posterior restauração para estabelecimento de um segundo ambiente operacional requer a autenticação do grupo de administradores e de pelo menos um grupo de auditores, portanto, a equipe de auditoria sempre vai saber quando uma nova instância do HSM for estabelecida. Nessa nova instância, a utilização da chave gerenciada está limitada ao grupo de operadores que possui a responsabilidade por seu uso, já que a propriedade de rastreabilidade das chaves gerenciadas do OpenHSM desfaz o ponto de confiança dos grupos de operadores em relação ao grupo de administradores em todas as cópias de segurança recuperadas. Este exemplo procura demostrar todos os recursos providos pelos protocolos do OpenHSM de modo a estabelecer um controle rigoroso sobre o controle do ciclo de vida das chaves gerenciadas, mesmo nos processos de criação e restauração de cópias de segurança.

A utilização de múltiplos ambientes operacionais também se aplica no balanceamento de carga para aplicações com alta demanda, se aplicando nestes casos o uso de ambientes operacionais idênticos. Um exemplo de aplicabilidade desta arquitetura é a paralelização da emissão de certificados e lista de certificados revogados por uma AC.

7.6 Conclusão

O protocolo do OpenHSM, utilizado para o gerenciamento do ciclo de vida das chaves criptográficas gerenciadas pelo ASI-HSM, foi publicado em eventos internacionais da área para apreciação do mundo acadêmico, com sua versão mais atual apresentada nestes dois últimos capítulos, utilizando uma nova forma de representação.

Deu-se especial atenção aos protocolos de criação e recuperação de cópias de segurança do material criptográfico, de forma a cobrir as lacunas em caso de falhas de hardware ou desastres. Além disso, extende-se essa funcionalidade de cópias de segurança para o estabelecimento de múltiplas instâncias de uma chave gerenciada.

Desta forma, tanto o OpenHSM quanto o ASI-HSM são uma realidade hoje, já estando instalado em 7 instituições de ensino brasileiras, aonde um projeto piloto de avaliação da ICPEDU está em curso.

Capítulo 8

Conclusão

O suporte ao controle do ciclo de vida de múltiplas cópias de uma chave em módulos criptográficos de mercado no contexto de uma infra-estrutura de chaves públicas praticamente inexiste e quando existe, o esquema é proprietário e restritivo à mesma família de produtos do fabricante. Mesmo assim, o controle é ineficaz no sentido de que não se consegue uma forte amarração entre cada uma das cópias das chaves, o que dificulta sobremaneira a auditoria.

O problema é maior quanto é preciso manter as chaves por longo período de tempo, como é o caso de uma autoridade certificadora. Normalmente, o tempo de manutenção de uma chave de uma AC Raiz é superior a 10 anos, o que é muito maior que o tempo médio esperado de vida útil de um HSM de mercado. É imperativo, portanto, a possibilidade de criação e recuperação de cópias de segurança, com total controle, em equipamento de diferentes fabricantes. Contudo, esta não é a praxe de mercado.

Foi realizado um estudo das principais normas e recomendações nacionais e internacionais que regem a construção de módulos criptográficos. Percebeu-se que nessas normas, o tratamento de múltiplas cópias do material criptográfico é pouco explorado. Aproveitou-se tal estudo para avaliar os protocolos do OpenHSM em relação ao requisitos apostos nestas normas.

Este trabalho tem como maior contribuição o aprimoramento do OpenHSM neste contexto, ou seja, como realizar uma cópia de segurança das cha-

ves criptográficas, realizar sua restauração e manter a rastreabilidade de cada uma das cópias. Todos os protocolos do OpenHSM foram analisados e onde necessário foram feitas modificações para se alcançar esta rastreabilidade.

8.1 Resumo das Contribuições

O presente trabalho apresentou diversas contribuições de caráter técnico e teórico-científico, atingindo seus objetivos geral e específicos, tais como os sumarizados a seguir:

- revisou-se os hardware criptográficos de mercado utilizados no gerenciamento do ciclo de vida de chaves criptográficas, com ênfase nos smartcards e HSMs;

- detalhou-se da ferramenta criptográfica OpenSSL, com a apresentação de sua infra-estrutura de suporte para usuários e desenvolvedores, focando-se particularmente no desenvolvimento de engines, que permitem a integração de aplicativos a hardware criptográficos;

- analisou-se os requisitos das normas FIPS 140-2 e MCT-7, utilizados na certificação de HSMs. Esta análise identificou pontos de extrema relevância em relação ao uso de HSMs no âmbito de ICPs que não são cobertos por estas normas, como um forte sistema de auditoria e um rígido controle das cópias de segurança do material sensível;

- apresentou-se a arquitetura e as aplicações adicionais de um HSM como alvo para o embarcamento do OpenHSM. Entres estas aplicações estão as interfaces de gerenciamento remoto e a engine que provê integração entre o HSM e outros aplicativos;

- revisou-se os protocolos de gerenciamento de chaves criptográficas do OpenHSM, propondo-se uma nova forma de representação. Essa revisão detalhou em particular o esquema de criação e recuperação de cópias de segurança;

- apresentou-se formas de utilização de múltiplas instâncias de uma chave gerenciada pelos protocolos do OpenHSM em diferentes ambientes operacionais. Esta abordagem possibilita, entre outras, a rápida retomada das operações em caso de falha do ambiente operacional principal ou visa suprir as necessidades de ambientes com alta demanda de processamento;
- publicação de dois artigos em eventos internacionais, apresentando os protocolos de gerenciamento de chaves criptográficas do OpenHSM;
- preparação de um artigo de evento introduzindo técnicas iniciais de modelagem e validação de cerimônias (ainda não submetido);
- preparação de um artigo de periódico que visa consolidar todo o conhecimento adquirido no projeto dos protocolos do OpenHSM (ainda não submetido).

8.2 Trabalhos Futuros

As áreas a se explorar com os protocolos de gerenciamento do ciclo de vida de chaves criptográficas do OpenHSM são inúmeras, servindo de tema para vários trabalhos futuros.

Uma destas áreas é o estudo detalhado para modelar e analisar cerimônias, que desempenham papel fundamental na operacionalização de um HSM, preenchendo lacunas que hardware e software não são capazes de resolver. Adicionalmente, elas podem ser extendidas, integrando procedimentos de manipulação de outros sistemas, como no caso de uma autoridade certificadora, onde o software de gestão de certificados também requer atenção especial.

Outra área importante de estudo é a análise formal dos protocolos do OpenHSM, com o intuito de comprovar formalmente a efetividade dos mesmos no gerenciamento do ciclo vida das chaves criptográficas gerenciadas em um HSM.

Finalmente, espera-se o aprimoramento dos HSMs comerciais, de forma que estes passem a suportar funcionalidades de rígido controle de suas cópias de segurança e de utilização de múltiplas instâncias operacionais da mesma chave.

Referências Bibliográficas

[1] REDE Nacional de Ensino e Pesquisa RNP. Disponível em: <http://www.rnp.br/>.

[2] SCHILLER, J. *Protecting a Private Key in a CA Context*. out. 2000. Disponível em: <http://www.cren.net/crenca/pkirscpages/private_key.html>.

[3] SOUZA, T. C. S. *Aplicações embarcadas para gerenciamento de chaves criptográficas*. [S.l.], 2005.

[4] MARTINA, J. E. *Project of a Hardware Security Module focused on Public Key Infrastructures and its Applications*. Dissertação (Mestrado) — Federal University of Santa Catarina, March 2005.

[5] MARTINA, J. E.; SOUZA, T. C. S. de; CUSTÓDIO, R. F. OpenHSM: An open key life cycle protocol for public key infrastructure's hardware security modules. In: LOPEZ, J.; SAMARATI, P.; FERRER, J. L. (Ed.). *EuroPKI*. [S.l.]: Springer, 2007. (Lecture Notes in Computer Science, v. 4582), p. 220–235. ISBN 978-3-540-73407-9.

[6] SOUZA, T. C. S. de; MARTINA, J. E.; CUSTÓDIO, R. F. Audit and backup procedures for hardware security modules. In: SEAMONS, K. E.; MCBURNETT, N.; POLK, T. (Ed.). *IDtrust '08: Proceedings of the 7th symposium on Identity and trust on the Internet*. ACM, 2008. (ACM International Conference Proceeding Series, v. 283), p. 89–97. ISBN 978-1-60558-066-1. Disponível em: <http://doi.acm.org/10.1145/1373290.1373302>.

[7] RANKL, W. W.; EFFING, W. *Smart Card Handbook*. Third. pub-WILEY:adr: John Wiley and Sons, Inc., 2004. xxviii + 1120 p. ISBN 0-470-85668-8.

[8] USB 2.0 TECHNICAL WORKING GROUPS. *Universal Serial Bus Revision 2.0 specification*. [S.l.], abr. 2000. Http://www.usb.org/developers/data/usb_20.zip.

[9] RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, 1978. ISSN 0001-0782.

[10] COOPER, D. et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. abr. 2008. Internet RFC 5280.

[11] ISO 7816 - Smart Card Standards Overview. ISO Standards, 1998. Disponível em: <http://www.iso.org/>.

[12] RSA Laboratories. *PKCS #15 v1.1: Cryptographic Token Information Syntax Standard*. pub-RSA:adr, jun. 2000. 81 p. Disponível em: <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-15/index.html>.

[13] LEE, A.; BARKER, E. B.; BARKER, W. C. *Guideline for Implementing Cryptography in the Federal Government*. [S.l.], dez. 2005.

[14] MENEZES, A. J.; OORSCHOT, P. C. van; VANSTONE, S. A. *Handbook of Applied Cryptography*. CRC Press, 1997. (CRC Press series on discrete mathematics and its applications). ISBN 0-8493-8523-7. Disponível em: <http://www.cacr.math.uwaterloo.ca/hac/index.html>.

[15] ISO/IEC STANDARD 15408. *Common Criteria for Information Technology Security Evaluation*. version 3.1. [S.l.], set. 2006. Disponível em: <http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf>.

[16] National Institute of Standards and Technology (NIST). *Security Requirements for Cryptographic Modules*. maio 2001. Federal Information Processing Standards Publication (FIPS PUB) 140-2. Updated 2002-12-03. Disponível em: <http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf, http://csrc.nist.gov/cryptval/140-2.htm>.

[17] National Institute of Standards and Technology (NIST). *Security Requirements for Cryptographic Modules*. jan. 1994. Federal Information Processing Standards Publication (FIPS PUB) 140-1. Disponível em: <http://csrc.ncsl.nist.gov/fips/fips1401.pdf, http://csrc.ncsl.nist.gov/fips/fips1401.htm>.

[18] FEDERAL COMMUNICATIONS COMMISSION. *Code of Federal Regulations, Title 47, Part 15, Subpart B, Unintentional Radiators, Digital Devices*, v. 1. http://frwebgate.access.gpo.gov/cgi-bin/getcfr.cgi?TITLE=47&PART=15&SECTION=101&YEAR=1998&TYPE=TEXT: FEDERAL COMMUNICATIONS COMMISSION, 1998.

[19] Instituto Nacional de Tecnologia da Informação (ITI). *Requisitos, Materiais e Documentos Técnicos para Homologação de Módulos de Segurança Criptográfica (MSC) no Âmbito da ICP-Brasil*. nov. 2007. Manual de Condutas Técnicas (MCT) 7. Disponível em: <http://www.iti.br/twiki/pub/Homologacao/Documentos/MCT7_-_Vol.I.pdf>.

[20] INSTITUTO Nacional de Tecnologia da Informação ITI. Disponível em: <http://www.iti.br/>.

[21] BRASIL. *Medida Provisória No 2.200-2*. 2001. Medida Provisória. Disponível em: <http://www.iti.br/twiki/bin/view/Certificacao/MedidaProvisoria>.

[22] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. out. 1999. FIPS Publication 46-3.

[23] DAEMEN, J.; RIJMEN, V. Rijndael for AES. In: *AES Candidate Conference*. [S.l.: s.n.], 2000. p. 343–348.

[24] RSA Laboratories. *PKCS #1 v2.1: RSA Cryptography Standard*. pub-RSA:adr, jun. 2002. 61 p. Disponível em: <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html>.

[25] PROCESSING Standards Publication 180-2. fev. 29 2004. Disponível em: <http://citeseer.ist.psu.edu/641912.html; http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>.

[26] MICROSOFT CryptoAPI and Cryptographic Service Providers.

[27] RSA. *PKCS #11: Cryptographic Token Interface Standard*. [S.l.], abr. 1997. Version 2.0. Disponível em: <ftp://www.rsa.com/pub/pkcs/ps/pkcs-11.ps, ftp://www.rsa.com/pub/pkcs/ascii/pkcs-11.asc>.

[28] EUROPEAN UNION. *Directive 2002/95/EC of the European Parliamente and of the Council*. [S.l.], jan. 2003.

[29] EUROPEAN UNION. *DIRECTIVE 2002/96/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. [S.l.], jan. 2003.

[30] THE OpenSSL Project. Disponível em: <http://www.openssl.org/>.

[31] FREIER, A. O.; KARLTON, P.; KOCHER, P. C. *The SSL Protocol — Version 3.0*. nov. 1996. Internet Draft, Transport Layer Security Working Group.

[32] DIERKS, T.; ALLEN, C. *The TLS Protocol Version 1.0*. nov. 1997. Internet Draft, TLS working group.

[33] APACHE HTTP Server Project. Disponível em: <http://httpd.apache.org/>.

[34] RSA Laboratories. *PKCS #10 v1.7: Certification Request Syntax Standard*. pub-RSA:adr, maio 2000. 10 p. Disponível em: <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/index.html>.

[35] GORA, W. *ASN.1 - Abstract Syntax Notation One*. Bergheim: Datacom-Verlag, 1992. ISBN 3-89238-062-7.

[36] VIEGA, J.; MESSIER, M.; CHANDRA, P. *Network Security with OpenSSL: Cryptography for Secure Communications*. pub-ORA:adr: O'Reilly & Associates, Inc., 2002. xiv + 367 p. ISBN 0-596-00270-X. Disponível em: <http://safari.oreilly.com/059600270X; http://www.oreilly.com/catalog/openssl>.

[37] OPEN Source Software Institute. [S.l.]. Disponível em: <http://www.oss-institute.org/>.

[38] AUTHORITIES, F. .-. C. M. V. *FIPS 140-2 Validation Certificate #642*. Março 2006. Disponível em: <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140crt/140crt642.pdf>.

[39] KELLAR, S. S. *NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms*. pub-NIST:adr, jan. 2005. 4 p. Disponível em: <http://csrc.nist.gov/groups/STM/cavp/documents/rng/931rngext.pdf>.

[40] AUTHORITIES, F. .-. C. M. V. *FIPS 140-2 Validation Certificate #733*. Fevereiro 2007. Disponível em: <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140crt/140crt733.pdf>.

[41] AUTHORITIES, F. .-. C. M. V. *FIPS 140-2 Validation Certificate #918*. Fevereiro 2008. Disponível em: <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140crt/140crt918.pdf>.

[42] SHAMIR, A. How to share a secret. *Commun. ACM*, ACM Press, New York, NY, USA, v. 22, n. 11, p. 612–613, 1979. ISSN 0001-0782.

[43] DOLEV, D.; YAO, A. C. On the security of public key protocols. *IEEE Transactions on Information Theory*, v. 29, n. 2, p. 198–208, 1983.

Apêndice A

Convenções

Esta seção apresenta as convenções usadas nos capítulos 6 e 7 para descrever os algoritmos que implementam os protocolos do OpenHSM, com a tabela A.1 listando todos os sistemas de armazenamento de dados utilizadas.

Tabela A.1: Sistemas de armazenamento de dados utilizados no OpenHSM

Entidade	**Descrição**
ADS	sistema de armazenamento de dados de administradores
$AudDS$	sistema de armazenamento de dados de auditores
BDS	sistema de armazenamento de dados de backup
BPF	pacote do backup
CTL	lista de Certificados Confiáveis
KDS	sistema de armazenamento de dados de chaves
LDS	sistema de armazenamento de registro de atividades do sistema
NXD	sistema de armazenamento de dados não exportáveis
ODS	sistema de armazenamento de dados de operadores

Adicionalmente, apresenta-se a descrição de todas as funções utilizadas nos algoritmos dos protocolos do OpenHSM, explicando seu objetivo e seus parâmetros:

ctDecrypt(ct**,** $edata$**,** eu **)** Envia para o smartcard ct os valores de $edata$ e eu. Estes dados serão então decifrados utilizando a chave privada dentro do smartcard, resultando, respectivamente, em $data$ e u. O valor retornado do smartcard no final da operação será $data$ cifrado com u.

decrypt($edata$**,** k **)** Decifra $edata$ utilizando a chave criptográfica k (simétrica ou assimétrica).

encrypt($data$**,** k **)** Cifra $data$ utilizando a chave criptográfica k (simétrica ou assimétrica). Se k é um certificado, sua chave pública é extraída e utilizada.

deleteLog(LDS**,** $rangeDate$ **)** Apaga os registros do sistema armazenados em LDS dentro do periodo de tempo especificado por $rangeDate$.

genCert(id**,** ku**,** c_{ca}**,** kr_{ca} **)** Emite um certificado utilizando ku como chave pública e id como nome comum. c_{ca} e kr_{ca} são respectivamente o certificado e a chave privada da autoridade certificadora que emitirá o novo certificado.

genKeyPair() Gera um par de chaves assimétricas.

genSelfSignedCert(kr**,** ku**,** id **)** Emite um certificado auto-assinado contendo a chave pública ku e o nome comum id. A assinatura é realizada com a chave privada kr.

genSessionKey() Gera uma chave simétrica.

joinSecret(Ks **)** Utiliza o conjunto de partes Ks para reconstrução do segredo. Entretanto, Ks deve conter o número mínimo de partes configuradas durante o compartilhamento.

load (DS**[,** id **)]** Carrega informações do sistema de armazenamento DS. (DS também pode representar o computador utilizado na manipulação do HSM). Opcionalmente, pode ser adicionado um identificador, id, que irá restringir o conjunto de informações resultantes.

manageKey(id**,** kr**,** ku**,** $policy$ **)** Submete o par de chaves gerenciadas ku e kr, identificada por id, ao controlador de chaves carregadas, ficando disponível para uso por aplicações externas ao HSM. A chave respeitará a política de uso $policy$.

revokeGroupCerts(DS**,** id **)** Revoga os certificados de todos os membros de um grupo identificado por id. Os dados dos membros e seus certificados são carregados de DS.

sign($data$**,** kr **)** Assina os dados contidos em $data$ utilizando a chave privada kr.

splitSecret(ks**,** k**,** n **)** Compartilha ks utilizando o mecanismo de compartilhamento de segredo, utilizando os limiares k e n, respectivamente, a quantidade de partes resultantes do compartilhamento e o número mínimo de partes necessárias para reconstruir ks.

store (DS**, ...)** Armazena todos os dados passados por parâmetro no sistema de armazenamento DS.

xor ($ks1$**,** $ks2$ **)** Realiza a operação ou-exclusivo entre o conteúdo de $ks1$ e $ks2$.