UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UM MODELO PARA PROTEÇÃO DE TRILHAS DE AUDITORIA EM SISTEMAS DE IDENTIFICAÇÃO ELETRÔNICA

DISSERTAÇÃO DE MESTRADO

Ernâni Teixeira Liberali

Santa Maria, RS, Brasil
2012

UM MODELO PARA PROTEÇÃO DE TRILHAS DE AUDITORIA EM SISTEMAS DE IDENTIFICAÇÃO ELETRÔNICA

Ernâni Teixeira Liberali

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Engenharia de Produção, Área de Concentração em Gestão da Produção, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Engenharia de Produção.

Orientador: Prof. Dr. Raul Ceretta Nunes

Santa Maria, RS, Brasil
2012

Ficha catalográfica elaborada através do Programa de Geração Automática da Biblioteca Central da UFSM, com os dados fornecidos pelo(a) autor(a).

Teixeira Liberali, Ernâni
Um Modelo para Proteção de Trilhas de Auditoria em Sistemas de Identificação Eletrônica / Ernâni Teixeira Liberali.-2012.
102 p.; 30cm

Orientador: Raul Ceretta Nunes
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Produção, RS, 2012

1. Proteção de Trilhas de Auditoria 2. Sistemas de Identificação Eletrônica I. Ceretta Nunes, Raul II. Título.

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

UM MODELO PARA PROTEÇÃO DE TRILHAS DE AUDITORIA EM SISTEMAS DE IDENTIFICAÇÃO ELETRÔNICA

elaborada por
Ernâni Teixeira Liberali

como requisito parcial para obtenção do grau de
Mestre em Engenharia de Produção

COMISSÃO EXAMINADORA:

_____
Raul Ceretta Nunes, Dr.
(Presidente/orientador)

_____
João Batista dos Santos Martins, Dr. (UFSM)

_____
Sergio João Limberger, Dr. (UFSM)

Santa Maria, 21 de maio de 2012.

AGRADECIMENTOS

Agradeço, primeiramente, a minha família, em especial, aos meus pais, Antonio e Marta, e minha irmã, Léia, que sempre me apoiaram e deram condições para que me desenvolvesse pessoal e profissionalmente.

Também quero agradecer a minha namorada, Tatiele, que sempre me apoiou e incentivou.

Quero agradecer ao meu orientador, professor Raul Ceretta Nunes, que, juntamente com o Programa de Pós-Graduação em Engenharia de Produção, me deu a oportunidade de participar deste trabalho e orientou-me durante estes dois últimos anos.

Também agradeço a oportunidade de ter conhecido e trabalhado com os amigos e colegas dos Programas de Pós-graduação e Engenharia de Produção e em Computação.

Por fim, agradeço aos Professores e aos funcionários do CT e CPD da UFSM, pelo apoio e amizade.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Engenharia de Produção
Universidade Federal de Santa Maria

UM MODELO PARA PROTEÇÃO DE TRILHAS DE AUDITORIA EM SISTEMAS DE IDENTIFICAÇÃO ELETRÔNICA

AUTOR: ERNÂNI TEIXEIRA LIBERALI
ORIENTADOR: DR. RAUL CERETTA NUNES
Data e Local da Defesa: Santa Maria, 21 de maio de 2012.

Com a contínua demanda por disponibilidade de serviços e de informações em diversos locais e em tempo real, as empresas estão tendo que lidar com informações cada vez mais sensíveis aos negócios, onde muitas delas não estão preparadas para realizar a gestão destas informações. Nos sistemas de informação, trilhas de auditoria, também chamadas de logs de auditoria, são registros das atividades de usuários e administradores. As trilhas de auditoria auxiliam as empresas a manterem um controle histórico sobre alterações nas informações, mas não eliminam a vulnerabilidade de manipulação indevida destas trilhas para eliminar rastros de modificações maliciosas, tal como o que pode acontecer no uso de *smart cards* para realização de micro pagamentos em instituições do setor educacional, o que é uma tendência. Este trabalho apresenta um modelo de proteção de trilhas (logs) que pode ser utilizado como solução para o problema do tratamento e proteção das trilhas de auditoria. O modelo é baseado em criptografia dos dados e em divisão de responsabilidades na guarda das chaves do registro, possibilitando condições de se garantir a legitimidade das informações em sistemas de identificação e pagamento, e foi validado junto ao banco de dados réplica ao sistema de pagamentos do Restaurante Universitário da Universidade Federal de Santa Maria.

Palavras-chave: Gestão de Segurança da Informação, Proteção de Trilhas de Auditoria, Distribuição de Chaves, *Smart Cards*.

ABSTRACT

Master Dissertation
Graduate Program in Production Engineering
Federal University of Santa Maria

A MANAGEMENT MODEL FOR AUDIT TRAILS IN IDENTIFICATION ELECTRONIC SYSTEMS

AUTHOR: ERNÂNI TEIXEIRA LIBERALI
ADVISER: RAUL CERETTA NUNES, DR.
Date and Local: Santa Maria, 21th May 2012.

With the continuing demand for services and information in multiple places in real time, companies are dealing with increasingly sensitive information for their business and many of them are not prepared to undertake the management of these information. In information systems, audit trails, also called audit logs, are records of activities from users and administrators. Audit trails help companies to keep a historical control of changes in information, but they do not safeguard the vulnerability of improper handling of these tracks nor eliminate traces of malicious changes, such as what might happen with the use of smart cards for micro-payments in educational institutions, which is a trend. This dissertation presents a model for protection of trails (logs) that can be used as a solution to problems on treatment and protection of audit trails. The model is based on data encryption and the sharing of responsibility in the care of registry keys, giving condition to guarantee the validity of information in systems of identification and payments. It was validated in the replica database to the payment system from the restaurant at Federal University of Santa Maria.

Keywords: *Information Security Management, Audit Trails Protection, Key Distribution, Smart Cards.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVEATURAS

ABNT	Associação Brasileira de Normas Técnicas
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
APDU	Applications Protocol Data Units
API	Application Program Interface
CPD	Centro de Processamento de dados
CPF	Cadastro de Pessoas Físicas
DES	Data Encription Standard
GTSeg	Grupo de Pesquisa em Gestão e Tecnologia em Segurança da Informação
HTTPS	HyperText Transfer Protocol secure
IBM	International Business Machines
IEC	International Eletrotechnical Comission
ISO	International Organization for Standardization
JDK	Java Development Kit
MAC	Message Authentication Code
NFC	Near Field Communication
NSA	National Security Agency
PC/SC	Personal Computer/SmartCard
RFID	Radio Frequency Identification
RU	Restaurante Universitário
SDK	Software Development Kit
SIE	Sistema de Informações para o Ensino
SIM	Sistema Integrado Municipal
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UID	Identificador Único
UFSM	Universidade Federal de Santa Maria
UNICAMP	Universidade Estadual de Campinas
UNISINOS	Universidade do Vale do Rio dos Sinos

SUMÁRIO

1. INTRODUÇÃO

A informação é um ativo que, como qualquer outro ativo importante para os negócios, tem valor para as organizações e, como consequência, necessita ser adequadamente protegida (ABNT NBR 17799, 2005). Nos sistemas de informação, falhas e desastres podem levar a grandes prejuízos e até a falência de uma empresa. Por outro lado, empresas de cartões de crédito, a Internet e a globalização potencializaram as relações comerciais por meio eletrônico, de forma que a segurança é uma preocupação da empresa, mas também do cliente ou do consumidor (FREITAS, 2009).

A segurança da informação corresponde tanto à proteção dos sistemas de informação contra a negação de serviço a usuários autorizados, bem como à proteção contra a intrusão e a modificação não autorizada de dados ou informações (armazenados, em processamento ou em trânsito) (ABNT NBR 17799, 2005). Por isso, cuidados especiais devem ser aplicados na Gestão da Segurança da Informação para verificar a integridade e para garantir que dados sensíveis estejam adequadamente protegidos.

A auditoria é uma das atividades fundamentais do processo de Gestão da Segurança da Informação. A tarefa da auditoria inclui recomendar ações para eliminar ou minimizar as perdas, através da identificação de vulnerabilidades e riscos, determinar se os controles de segurança adequados estão em vigor, garantir que os dispositivos de auditoria e segurança sejam válidos e verificar se os controles, as trilhas de auditoria e as medidas de segurança estão funcionando de forma eficaz (BOSWORTH; KABAY, 2002). Em sistemas automatizados, o processo de auditoria envolve a coleta, a manutenção e a análise das trilhas de auditoria, também chamadas de *logs* de auditoria, que são formadas por registros das atividades de usuários e administradores.

Os registros dos *logs* que compõem as trilhas de auditoria devem dar condições de se reconstruir o cenário da ação, com a identificação de quem fez a ação, qual foi a ação, quando e onde foi feita. Informações relacionadas com as operações realizadas (trilhas) também devem ser mantidas em arquivos de *logs s*eguros (armazenamento seguro das trilhas) para possibilitar a detecção de intrusão e violações de integridade. Portanto, é importante garantir que, caso ocorra alguma violação do sistema, os *logs* não sejam comprometidos e a violação possa ser detectada posteriormente (XU *et al*. 2005).

Neste contexto, para não ser descoberto, o primeiro alvo de um atacante experiente costuma ser o sistema de trilha de auditoria, pois o seu objetivo é apagar os rastros do ataque para escapar da detecção e manter o método de ataque em segredo, fazendo com que as falhas de segurança exploradas não sejam detectadas pelos administradores do sistema (BELLARE; YEEY, 1997). Como consequência, há necessidade de proteção dos registros de *logs* através de técnicas que permitam garantir a segurança das trilhas de auditoria para protegê-las de intrusões, modificações e danos causados por qualquer usuário ou administrador, sendo de forma intencional ou não.

Um tipo de aplicação em que é essencial o uso de trilhas de auditoria é o de identificação eletrônica e micropagamentos por meio de *smart cards*, que busca mais mobilidade e segurança nos processos de identificação e micropagamentos. O *smart card* é um dispositivo que possibilita a identificação positiva do usuário, ou seja, valida a identidade de seu portador, permitindo simplificar as atividades de pagamento e identificação (TAHERDOOST et al, 2010). Sua capacidade de armazenamento de informações viabiliza a sua utilização como dinheiro "de plástico" ou mesmo como identificador único (ID) para transações financeiras.

O baixo custo e a facilidade de uso favorecem a proliferação de soluções com *smart card* como cartão universitário, tanto para identificação no acesso, quanto para a realização de micropagamentos no âmbito de instituições universitárias. Porém, por não ser do segmento financeiro, neste tipo de instituição, pode haver a violação de informações financeiras por parte de usuários internos, incluindo programadores e administradores do sistema. Em geral, estas pessoas têm permissões de acesso que as habilitam a entrar no sistema e fazer modificações sem levantar suspeitas, inclusive, nos registros das trilhas de auditoria.

Este trabalho apresenta e desenvolve uma proposta para a manutenção segura de trilhas de auditoria em ambientes sujeitos a atacantes internos e aplica-o no sistema de cartão universitário da Universidade Federal de Santa Maria (UFSM).

1.1. Objetivos

O objetivo principal do trabalho é propor, desenvolver e validar um modelo de proteção de trilhas de auditoria baseado num mecanismo seguro de divisão de responsabilidades na guarda de chaves criptográficas e aplicá-lo ao sistema de cartão

universitário com a tecnologia de *smart cards*. O foco é a proteção contra violações feitas por programadores e administradores vinculados a própria empresa (atacantes internos).

Os objetivos específicos compreendem:

- Propor e desenvolver um modelo funcional que dê provimento à proteção de trilhas de auditoria em sistemas de identificação eletrônica para o cartão universitário;
- Validar o modelo, nas trilhas de auditoria, do sistema de pagamentos via *smart card* do Restaurante Universitário da UFSM, de forma que possibilite ao Diretor do restaurante e/ou ao gestor financeiro terem garantias sobre a integridade do sistema de trilhas de auditoria com relação a pagamentos recebidos e refeições servidas.

1.2. Justificativa

Em modelos tradicionais de proteção de trilhas de auditoria, como os elaborados nos trabalhos Bellare e Yee (1997 e 2003) e Schneier e Kelsey (1998 e 1999), há uma expressiva preocupação com o desenvolvimento do algoritmo de criptografia das trilhas auditoria. Porém, os trabalhos encontrados na literatura não se preocupam com a gestão das chaves criptográficas ou sobre a divisão de responsabilidade de quem ficará com a guarda das chaves criptográficas do registro de *log*. Isto demonstra que, em sistemas com políticas de segurança menos especializados, como os modelos de proteção de trilhas de auditoria para sistema de cartão universitário, só um mecanismo de criptografia simples não poderá resolver o problema da gestão destas trilhas no âmbito da segurança. É preciso ter um modelo de proteção das trilhas de auditoria que apoie o sistema criptográfico, uma vez que de nada adianta ter um sistema de criptografia eficiente se as chaves não forem corretamente gerenciadas.

O uso de *smart cards* como uma forma de controle, para ter acesso às instalações de instituições de ensino superior, já pode ser considerado difundido. Porém, diversas instituições de ensino superior estão automatizando, além da identificação, os serviços de pagamento com *smart cards*, o que exige maior cuidado na proteção das trilhas de auditoria.

O *smart card* usado como cartão universitário funciona como sistema de identificação eletrônica em mais de 200 universidades de todo o mundo (GEMALTO, 2011). Nos Estados Unidos da América, por exemplo, várias universidades usam esta tecnologia para integrar diversos serviços em um único cartão, que aproxima os serviços universitários dos usuários e permite criar maior proteção para este grupo (HENDRY, 2007).

No Brasil, tem-se pelo menos dois exemplos: a Unicamp (Universidade Estadual de Campinas) e a UNISINOS (Universidade do Vale do Rio dos Sinos). O projeto da Unicamp foi iniciado em 2002 e os cartões são utilizados nas bibliotecas da universidade e no refeitório para o controle de acesso e pagamento de refeições. O *chip* contido no cartão propicia, ainda, que sejam armazenadas informações como a assinatura digital do portador e até mesmo a sua impressão digital (UNICAMP, 2011). Já o projeto da UNISINOS possibilita que os alunos e os professores empreguem o cartão para ter acesso a instalações, para retirar livros e como chave eletrônica nos armários guarda-volumes da biblioteca (UNISINOS, 2010).

O modelo de proteção de trilhas de auditoria desenvolvido, neste trabalho, apresenta um diferencial dos demais modelos estudados. Este diferencial refere-se à forma como os dados são criptografados e como as duas chaves resultantes do processo são gerenciadas. Na prática, o modelo desenvolvido permite que o Diretor do restaurante universitário, ou o seu gestor financeiro, possa ter garantias sobre a integridade do sistema de trilhas de auditoria sobre transações de pagamentos de refeições.

1.3. Metodologia

De acordo com Silva e Menezes (2005), o presente trabalho caracteriza-se como pesquisa aplicada, visto que propõe um modelo de proteção de trilhas de auditoria que visa a solucionar o problema de uma aplicação já existente, o da proteção de trilhas de auditoria de sistema de identificação eletrônica e pagamentos com *smart cards*. Por outro lado, a abordagem do problema assume uma forma de pesquisa qualitativa, pois o desenvolvimento da pesquisa fez-se por um estudo bibliográfico inicial das tecnologias existentes na área de proteção das trilhas de auditoria, seguido do desenvolvimento do modelo de proteção de trilhas de auditoria para sistemas de cartão universitário e a sua correspondente implementação. A análise qualitativa, na avaliação, ocorreu por meio de observação da conformidade de seu funcionamento em relação à proteção das trilhas de auditoria no âmbito gestão da segurança de trilhas de auditoria em sistemas de cartão universitário.

Ainda conforme os autores Silva e Menezes (2005), pode-se apontar que existem procedimentos técnicos que também foram utilizados no trabalho que se apresenta:

a) Pesquisa bibliográfica: que foi elaborada a partir de material já publicado, constituído, principalmente, por livros, artigos de periódicos e, atualmente, com material

disponibilizado através da Internet. A realização de uma revisão bibliográfica aprofundada foi fundamental para relacionar estratégias existentes.

b) Estudo de caso: envolveu o estudo profundo e exaustivo de um objeto, o sistema de cartão universitário da UFSM, de maneira que se teve um conhecimento detalhado sobre o mesmo.

1.4. Organização da Dissertação

No capítulo 2 (dois) traz-se a revisão bibliográfica que norteia a execução deste trabalho. O capítulo 3 (três) apresenta a modelagem e o desenvolvimento do modelo de proteção de trilhas de auditoria proposto e o capítulo 4 (quatro) descreve a o funcionamento do Restaurante Universitário e o desenvolvimento do Sistema de Identificação Eletrônica com *smart card*. Também é apresentado, no capítulo 4 (quatro), o processo de validação do modelo de proteção de trilhas de auditoria, e no capítulo 5 (cinco) estão presentes as principais conclusões do trabalho.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo, apresenta-se uma revisão bibliográfica dos principais conceitos de gestão de segurança da informação do ponto de vista da auditoria e das trilhas de auditoria. Primeiramente, são expostos os conceitos de classificação da informação referente ao seu nível de criticidade para o negócio (seção 2.1), os critérios de segurança da informação (seção 2.2) e também os conceitos sobre Processo de Auditoria (seção 2.3) e Trilhas de Auditoria (seção 2.4). Na (seção 2.5) são apresentados os principais trabalhos referente a proteção de trilhas de auditoria. Em seguida, é apresentada a tecnologia de identificação por *smart cards* e a sua aplicação como mecanismo de identificação e micropagamentos em instituições de ensino (seção 2.5). Além disso, são apresentadas as Técnicas Criptográficas, Conceitos de Gerenciamento das Chaves Criptográficas e os Principais Tipos de Algoritmos Criptográficos (seções 2.5, 2.6 e 2.7, respectivamente). Estes conceitos são importantes para que se entenda o processo e a necessidade de desenvolver um modelo de proteção de trilhas de auditoria que assegure que, mesmo os responsáveis pelo sistema, não o violem sem deixar rastros auditáveis.

2.1. A Informação do Ponto de Vista Empresarial

A informação, na visão de Rezende e Abreu (2000), é o dado com uma interpretação lógica ou natural agregada pelo usuário. A informação é um ativo que, como qualquer outro ativo importante para os negócios, tem um valor para a organização e, como decorrência, necessita ser adequadamente protegido (ABNT NBR 17799, 2005). Como salienta Dias (2000), a informação é o principal patrimônio da empresa e está sob risco constante. Ela representa a inteligência competitiva dos negócios e é reconhecida como um ativo crítico para a continuidade operacional e a saúde da empresa (SÊMOLA, 2003). A informação e o conhecimento constituem os diferenciais das empresas e dos profissionais que pretendem destacar-se no mercado e manter a sua competitividade (REZENDE e ABREU, 2000).

2.2. Critérios da Segurança da Informação

A segurança da informação corresponde tanto à proteção dos sistemas de informação contra a negação de serviço a usuários autorizados, bem como à proteção contra a intrusão e a

modificação não autorizada de dados ou informações (armazenados, em processamento ou em trânsito). Sua realização, pois, abrange a segurança dos recursos humanos, da documentação, dos materiais, das áreas e instalações, das comunicações e de recursos computacionais, assim como ações destinadas a prevenir, detectar, deter e documentar eventuais ameaças a seu desenvolvimento (ABNT NBR 17799, 2005; DIAS, 2000; KRAUSE e TIPTON, 1999).

A maioria das atividades humanas está cada vez mais dependente das tecnologias e esta precisam proporcionar confidencialidade, integridade e disponibilidade. Portanto, é muito importante que mecanismos de segurança de sistemas de informação sejam projetados de maneira a prevenir acessos não autorizados aos recursos e aos dados dos sistemas. Em conformidade com Laureano e Morais (2005), há três princípios básicos para garantir a segurança da informação, principalmente no que se referem a sistemas que envolvam dados pessoais e financeiros, tais como:

Confidencialidade - a informação somente pode ser acessada por pessoas explicitamente autorizadas;

Disponibilidade - a informação deve estar disponível no momento em que for necessária; e

Integridade - a informação deve ser recuperada em sua forma original (no momento em que foi armazenada).

Rezende e Abreu (2000) e Sêmola (2003) defendem ainda que, para que uma informação seja considerada segura, o sistema, que a administra, ainda deve respeitar os seguintes critérios:

Autenticidade - garante que a informação ou o usuário é autêntico;

Não repúdio - não é possível negar uma operação ou serviço;

Legalidade - garante a legalidade (jurídica) da informação; em que todos os ativos estão de acordo com as cláusulas contratuais pactuadas ou com a legislação nacional ou internacional vigente;

Privacidade - foge do aspecto de confidencialidade, pois uma informação pode ser considerada confidencial, mas não privada. Uma informação privada deve poder ser vista / lida / alterada somente pelo seu dono; e

Auditoria - rastreabilidade dos diversos passos de um negócio ou processo, identificando os participantes, os locais e os horários de cada etapa. A auditoria aumenta a credibilidade da empresa e é responsável pela adequação da empresa às políticas legais e internas.

Destes critérios, os princípios básicos de segurança devem ser garantidos para que o critério de auditoria seja seguro, em especial, para evitar a manipulação indevida por colaboradores internos que tenham permissão de acesso às informações.

2.3. O processo de Auditoria

A auditoria pode ser entendida como um processo que procura identificar e evitar ações suspeitas e fraudulentas por parte do usuário (externo e interno), coletando dados sobre as suas atividades em registros de *log*. As informações coletadas podem ser analisadas a fim de descobrir problemas de segurança e a sua origem (SIMON et al. 2008). A auditoria pode ser vista também como um protocolo de contestação e resposta entre o auditor e o sistema a ser auditado (PETERSON et al, 2007). Porém, salienta-se que a principal funcionalidade de um serviço de auditoria é oferecer armazenamento seguro e permanente dos registros de *log*, para que os registros possam ser utilizados para detectar quando e como uma falha de segurança ocorreu (XU *et al*. 2005).

Identificar quais foram as ações e determinar os padrões suspeitos são importantes requisitos para a segurança do sistema. Por isso, a auditoria deve ser realizada de maneira independente e transparente, de forma que todas as informações relevantes sejam catalogadas (HAWTHORN et al. 2006). Uma trilha de auditoria, ou conjunto de registros de *log*, é usada para assegurar o fluxo preciso das transações em um sistema. Cada detalhe de uma fonte, entrada de um determinado documento ou transação deve ser feito com base em um relatório ou arquivo. A técnica de rastreamento pode ser aplicada em uma única transação para teste rápido; no entanto, para garantir que o controle funcione consistentemente, o teste deve cobrir grandes volumes de dados em diferentes períodos de tempo (BOSWORTH; KABAY, 2002). As trilhas de auditoria devem ser construídas como parte integrante dos sistemas de controles internos, possibilitando o registro de auditoria automatizado.

Na prática, a trilha de auditoria de um sistema computacional pode incluir uma entrada para cada operação aplicada ao banco de dados ou recurso computacional. Estas entradas registradas na trilha podem, então, ser necessárias para a recuperação de uma falha de operação ou falha do sistema (ELMASRI; NAVATHE, 2004). É possível expandir as entradas da trilha para que elas também incluam o número da conta do usuário e terminal *on line* para que seja aplicado a cada operação registrada na trilha. Se há suspeita de violação da

base de dados, uma auditoria do banco de dados pode ser realizada, o que consiste em analisar a trilha de auditoria para examinar todos os acessos e as operações aplicadas ao banco de dados durante um determinado período de tempo (ELMASRI; NAVATHE, 2004). Quando uma operação ilegal ou não autorizada é encontrada, o responsável pelo sistema ou o auditor do sistema pode determinar o número da conta usada para executar a referida operação. Auditorias de banco de dados são particularmente importantes para bancos de dados sensíveis, que são atualizadas por muitas transações e usuários, como um banco de dados bancário, que é atualizado por vários terminais de atendimento (ELMASRI; NAVATHE, 2004).

Para realizar uma auditoria, o auditor acessa aos metadados[1] e faz contestações ao sistema de arquivos, confrontando as informações obtidas com as representadas nos metadados (PETERSON *et al*. 2007). Porém, o sistema deve estar preparado para resistir a ataques, como a criação de históricos e versões falsas das trilha de auditoria (PETERSON *et al*. 2007). Um ataque pode incluir a criação de versões falsas do arquivo de dados que coincide com os metadados publicados, mas diferem dos dados utilizados na sua criação. Além disso, pode incluir a criação de históricos falsos; inserção ou exclusão de versões em uma sequência sem detecção.

Diante disso, no contexto da auditoria de sistemas, é fundamental que haja um planejamento a fim de que se possa contemplar o máximo de requisitos para análise, ou seja, é necessário observar alguns requisitos relevantes para a efetiva realização de uma auditoria bem sucedida.

Dentre os itens que se pode destacar no processo de planejamento da auditoria, estão:

- conhecimento do ambiente computacional;
- determinação dos pontos de controle;
- estabelecimento dos objetivos de validação e avaliação dos pontos de controle, como técnicas de auditoria, prazos de execução da validação, custos incorridos com a validação, nível de tecnologia exigida do auditor e natureza da fraqueza do controle interno passível de ser alcançada;
- verificação da sensibilidade de cada ponto de controle: matriz ponto de controle, parâmetro, voto, fraqueza, técnica de auditoria a aplicar.
- hierarquização dos pontos de controle; e

[1] Metadado: Os metadados representam informações que caracterizam, de forma sintética, a informação documentada (DevMedia, 2011).

- documentação e registro do processo de planejamento da auditoria.

Finalmente, destaca-se a importância de um processo de auditoria como uma atividade que objetiva garantir a segurança e visa à continuidade do negócio.

2.4. Trilhas de Auditoria

Segundo a norma ABNT NBR 17799 (2005), as trilhas de auditoria, que podem ser compostas por um ou mais *log* de auditoria, podem conter dados pessoais confidenciais e de intrusos e convém que medidas apropriadas de proteção de privacidade sejam tomadas. Quando possível, é aceitável também que administradores de sistemas não tenham permissão de exclusão ou desativação dos registros (*log*) de suas próprias atividades. A modificação ou a exclusão de registros pode causar falsa impressão de segurança.

Ainda, conforme a ABNT NBR 17799 (2005), as trilhas de auditoria podem conter dois tipos de registros (*log*):

- Registros de administrador e operador - que contêm informações sobre atividades no sistema e podem ser utilizados para monitorar a conformidade das atividades dos administradores e operadores do sistema e da rede. Este tipo de registro deve abarcar basicamente:

    a) a hora em que o evento ocorreu;

    b) informações sobre o evento ou a falha;

    c) que conta e que administrador ou operador estava envolvido e

    d) que processos que estavam envolvidos.

- Registros de falhas e erros - que contêm informações sobre falhas e erros informados pelos usuários ou pelos programas de sistema relacionados a problemas com processamento da informação ou sistemas de comunicação.

A automação do processo de construção de trilhas de auditoria, através de mecanismos que possibilitam garantir a segurança da trilha, é questão chave no processo de gestão da segurança de informação, principalmente, em sistemas de identificação e micropagamentos com *smart cards*.

2.5. Modelos de Proteção de Trilhas de Auditoria

De maneira similar, os logs também são alvos de atacantes do sistema, que objetivam apagar os registros armazenados em seu ataque escapando de uma posterior detecção, mantendo assim, o método de ataque em segredo para que as falhas de segurança exploradas não sejam detectadas pelos administradores do sistema (BELLARE; YEE, 1997). Como consequência, há uma necessidade de proteção aos registros de logs utilizando técnicas que permitam garantir a segurança das trilhas de auditoria para protegê-los de adulterações e danos causados por qualquer usuário, quer seja de forma intencional ou não.

Bellare and Yee (BELLARE; YEE, 1997) introduziram uma nova propriedade de segurança em logs de auditoria denominada *forward integrity*. A proposta de Bellare and Yee é baseada no uso de MAC (*message authentication code*) e não utiliza a replicação em *host* remoto, prevenindo a alteração de informações por um atacante. Schneier e Kelsey (SCHNEIER e KELSEY, 1998 e 1999) além de utilizar MAC, usam cadeias de *hash* para interligar os registros da trilha de auditoria após cada entrada de log. Desta forma, cada registro de log possui uma ligação realizada para autenticar todas as entradas prévias. Wu, ZhuGe and Wang (Wu et. al 2010) propõem um novo esquema de proteger logs com assinatura digital assimétrica. Conforme Tanenbaum (TANENBAUM, 2003), a maioria das aplicações que usam criptografia assimétrica também utilizam algum algoritmo simétrico para a codificação da mensagem. Estes modelos tratam do processo de codificação das trilhas, mas não deixam claro como serão armazenadas as chaves criptográficas necessárias para a encriptação e decriptação, das trilhas protegidas.

2.6. Sistemas de Identificação e MicroPagamentos com Cartão Universitário Inteligente

O termo *cartão inteligente*, ou *smart card*, refere-se a qualquer cartão que contenha um circuito integrado (*chip*) para realizar o armazenamento e o processamento de informações (CHEN, 2000). Além disso, o termo *cartão criptograficamente inteligente* refere-se a cartões inteligentes acrescidos de coprocessadores especializados em alguns sistemas de criptografia específicos, como RSA e ECC (RANKL; EFFING, 2003). A maioria

dos cartões inteligentes lembra o tamanho de um cartão de crédito padrão e segue o padrão da série IS0 7800 (FABCHER, 1997).

A família MIFARE é a tecnologia de cartões inteligentes sem contato baseada na tecnologia dos *smart cards* com *chip*. A sua frequência de trabalho é de 13,56 MHz com capacidade de leitura e escrita. Existem diferentes modelos, segundo a quantidade de informação que armazenam, além de respeitar a norma ISO/IEC 14443 referente a cartões de proximidade (GARCIA *et al*, 2009).

De acordo com o fabricante de *smart cards*, Gemalto (2011), o sistema de identificação baseado em *smart cards* vem tendo o seu uso difundido nos serviços de transporte urbano e identificação de acesso e serviços de micropagamentos. O *cartão universitário inteligente* funciona como uma credencial de identidade em mais de 200 universidades de todo o mundo. Nos Estados Unidos da América, a Universidade Estadual da Flórida utiliza o Cartão Inteligente (FSU *SmartCard*) como meio para a identificação de alunos. Esta solução contém chaves de segurança, serve para registro da utilização na biblioteca no caso de retirada de livros, dentre outras funcionalidades (FSU, 2011).

No Brasil, instituições de ensino como Unicamp (Universidade Estadual de Campinas), Unesp (Universidade Estadual Paulista) e UNISINOS (Universidade do Vale do Rio dos Sinos) são alguns exemplos que já fazem uso de sistemas de identificação e micropagamentos com *smart cards*. Estes sistemas utilizam *smart cards* do tipo Mifare 1k como plataforma de seus sistemas de identificação.

2.7. Técnicas Criptográficas

Nesta seção, descreve-se o processo criptográfico e faz-se a descrição dos algoritmos assimétrico e simétrico. Ambos os algoritmos são utilizados no desenvolvimento do algoritmo de proteção de trilhas de auditoria.

2.7.1. Entendendo o processo criptográfico

A criptografia surgiu pela necessidade de enviar informações sensíveis por meios de comunicação não confiáveis, onde um intruso possa interceptar o fluxo de dados para leitura, cópia ou modificação da informação. A palavra criptografia vem do grego *kriptós*, que

significa escondido, oculto, e *grápho*, que significa grafia, escrita. Como apresentado por Schneier em Uchôa (2003), a criptografia é a arte e a ciência de manter mensagens seguras.

As mensagens a serem criptografadas, chamadas de texto simples, são transformadas por uma função que é parametrizada por uma chave criptográfica (um código alfanumérico - *string*). A saída da função, ou a saída do processo de criptografia, conhecida como texto cifrado, pode, então, ser transmitida por um meio de comunicação. Mesmo que um intruso intercepte e copie o texto cifrado, ele não conseguirá violar a confidencialidade da informação, pois para descriptografar o texto, necessitará conhecer a chave.

Segundo Tanenbaum (2003), a arte de criar mensagens cifradas (criptografia) e solucioná-las (criptoanálise) é chamada coletivamente criptologia. Neste trabalho, adota-se a notação C = EK (P) para denotar que a criptografia (E) do texto simples P, usando a chave K, gera o texto cifrado C. Da mesma forma, P = DK (C) representa a descriptografia (D) de C com a chave K para obter-se o texto simples P, outra vez. Observe que E e D são funções matemáticas, onde E é o método de criptografia e D, o de descriptografia.

O princípio do processo criptográfico é o de que todo texto criptografado pode ser descriptografado. Se o processo usa a mesma chave para a criptografia e a descriptografia, tem-se que:

$$D_{\mathbf{K}}(E_{\mathbf{K}}(P)) = P$$

Esta notação segue a definida em (TANENBAUM, 2003).

2.7.2. A importância do tamanho das chaves

Em um sistema criptográfico, o sigilo está na chave e o seu tamanho é uma questão muito importante do projeto. Uma chave com um tamanho de dois dígitos decimais significa que existem 100 possibilidades de criptografar os dados. De forma similar, uma chave de três dígitos decimais indica mil possibilidades e uma chave de seis dígitos decimais significa um milhão de possibilidades. Logo, facilmente observa-se que quanto maior for a chave, mais alto será o fator de trabalho com que o criptoanalista terá de lidar.

O fator de trabalho para decodificar o sistema através de uma exaustiva pesquisa no espaço da chave é exponencial em relação ao tamanho da chave. O sigilo é decorrente da presença de um algoritmo forte (mas público) e de uma chave longa. Por exemplo, para desencorajar que alguém leia as suas mensagens de correio eletrônico, serão necessárias chaves de 64 bits (TANENBAUM, 2003). Para uso comercial de rotina, devem ser usados

pelo menos 128 bits. Para manter atacantes à distância, são necessárias chaves de pelo menos 256 bits, ou de preferência maiores (TANENBAUM, 2003).

2.7.3. Algoritmos de chave simétrica

Os algoritmos de chave simétrica, tal como o AES (*Advanced Encryption Standard*), utilizam a mesma chave para codificação e decodificação. Brocardo *et al*. (2006) explicam que, para acontecer a troca e a guarda da chave, é necessário uma política de segurança definida, tendo em vista que ocorre o chamado "problema de distribuição de chaves", porque a chave tem de ser enviada para todos os usuários autorizados antes que as mensagens possam ser trocadas, ela pode ser alvo de ataques.

Algoritmos simétricos são vulneráveis em dois aspectos: conservação do segredo (chave) – basta um participante agir com más intenções para o processo todo ser comprometido; distribuição da chave – sempre que uma nova entidade é admitida no grupo, a chave deve ser distribuída e pode ser alvo de ataques.

Alves *et al*. (2007) citam algumas desvantagens da criptografia simétrica: a impossibilidade de implementação de não-repúdio, já que qualquer pessoa pode cifrar e decifrar o código; a necessidade de gerar chaves frequentemente, com uma certa periodicidade, para não comprometer a segurança de todo o sistema; e o número elevado de chaves necessário. Entretanto, os autores apresentam duas características vantajosas deste método: o alto desempenho e a facilidade de implementação.

2.7.4. Algoritmo de chave assimétrica

Ao contrário do modelo simétrico, na criptografia assimétrica, a chave utilizada na encriptação difere daquela usada na decriptação. Matematicamente, os modelos são relacionados, mas, no modelo assimétrico, cada participante do grupo possui duas chaves, uma pública, que deve ser distribuída, e outra privada, que deve ser guardada em segredo (AMARO, 2007).

As chaves assimétricas são geradas em pares, pública e privada, fazendo-o a partir de números aleatórios. Estas chaves podem ser utilizadas de duas formas diferentes para que se garanta a autenticidade ou a confidencialidade da informação criptografada:

- a autenticidade é obtida quando a chave privada é usada para cifrar as mensagens, com isso, garante-se que apenas o dono da chave privada poderia ter cifrado a mensagem que foi decifrada com a ”chave pública”, e que a mensagem não foi forjada; e
- a confidencialidade é obtida quando a chave pública é usada para cifrar mensagens, sendo que apenas o dono da chave privada pode decifrá-la.

2.8. Conceitos de Gerenciamento das Chaves Criptográficas

Se as chaves utilizadas no processo de criptografia não forem tratadas e armazenadas corretamente, o uso de modelos criptográficos eficientes não garante a autenticidade e a confidencialidade das informações protegidas. Segundo a norma ABNT NBR 17799 (2005), todas as chaves criptográficas devem ser protegidas contra modificação, perda e destruição. As chaves secretas e privadas devem ser protegidas contra a divulgação não autorizada e que os equipamentos empregados para gerar, armazenar e guardar as chaves sejam fisicamente protegidos.

Ainda, em consonância com a norma ABNT NBR 17799 (2005), um sistema de gerenciamento de chaves deve ser baseado em um conjunto estabelecido de normas, procedimentos e métodos de segurança para:

a) gerar chaves para diferentes sistemas criptográficos e diferentes aplicações;

b) gerar e obter certificados de chaves públicas;

c) distribuir chaves para os usuários devidos, incluindo a forma como as chaves devem ser ativadas, quando recebidas;

d) armazenar chaves, incluindo a forma como os usuários autorizados obtêm acesso a elas;

e) mudar ou atualizar chaves, incluindo regras relativas sobre quando as chaves devem ser mudadas e como isso será feito;

f) lidar com chaves comprometidas;

g) revogar chaves, incluindo regras sobre como elas devem ser retiradas ou desativadas, por exemplo, quando chaves tiverem sido comprometidas ou quando um usuário deixa a organização (convém que, também, neste caso, as chaves sejam guardadas);

h) recuperar chaves perdidas ou corrompidas, como parte da gestão da continuidade do negócio, por exemplo, para recuperação de informações cifradas;

i) guardar chaves, por exemplo, para as informações guardadas ou armazenadas em cópias de segurança;

j) destruir chaves;

k) manter registro e auditoria das atividades relacionadas com o gerenciamento de chaves.

A norma ISO/IEC 11770 também orienta que: em técnicas de chaves secretas, onde a mesma chave é utilizada para a cifragem e decifragem das mensagens e dois ou mais usuários compartilham a mesma chave. A chave deve ser mantida secreta, uma vez que qualquer um que tenha acesso a ela será capaz de decifrar todas as informações cifradas com ela ou modificar as informações existentes de forma não autorizada. Técnicas criptográficas também podem ser adotadas para proteger chaves criptográficas, a fim de aumentar a segurança e o desempenho.

2.9. Principais tipos de Algoritmos Criptográficos

2.9.1. Algoritmos que utilizam apenas técnicas de criptografia simétrica

Nesta subseção, são descritos os algoritmos simétricos estudados no processo de desenvolvimento da aplicação de proteção de trilhas de auditoria.

2.9.1.1. Algoritmos RC2 e RC4

O RC2 e o RC4 são algoritmos criados pelo Professor Ronald Rivest, proprietário da *RSA Data Security* (TRINTA, 1998). Estes algoritmos usam chaves que variam de 1 a 1024 bits de extensão. Com chaves pequenas (menores que 48 bits), são códigos fáceis de serem quebrados e, como são proprietários, não se tem muitas informações sobre a sua segurança com chaves extensas (ALVARENGA, 2011).

RC2 é uma cifra de bloco, similar ao DES (seção 2.8.1.2). RC4 é uma cifra de corrente, onde o algoritmo produz uma corrente de pseudonúmeros que são cifrados através de uma operação lógica XOR com a própria mensagem.

2.9.1.2. DES e Triple-DES

O algoritmo DES (*Data Encription Standard*) é baseado num algoritmo desenvolvido pela IBM, chamado Lúcifer, e é o padrão utilizado pelo governo norte-americano para a criptografia de seus dados desde 1978 (IBM, 2004). Em 1981, foi adotado como padrão pela ANSI com o nome de DEA, como tentativa de padronizar procedimentos de cifragem do segmento privado, especialmente em instituições financeiras. O DES utiliza cifras de blocos de 64 bits usando uma chave de 56 bits, fazendo uma substituição monoalfabética sobre um alfabeto de 264 símbolos. O algoritmo Triple-DES baseia-se na utilização de três vezes seguidas do DES com chaves diferentes. E passou a ser utilizado pelo governo norte-americano em meados de 1998.

A limitação destes algoritmos está no seu tamanho de chave e desempenho. O limite no tamanho da chave gerada é de 168 bits e o algoritmo, com o tempo, mostrou-se ineficaz na velocidade em cifrar/decifrar os dados, se comparado com os novos algoritmos.

2.9.1.3. AES

O AES foi um algoritmo criado por Vincent Rijmen e Joan Daemen, concebido depois de um concurso criado pelo governo norte-americano, em 2001, para a apresentação de novos algoritmos de criptografia. O AES tornou-se o algoritmo de criptografia padrão do governo norte-americano em 2001, ocupando o lugar do Triple DES (IBM, 2004), dado a característica de poder gerar chaves maiores, de 192 ou 256 bits, e de ser muito mais rápido, proporcionando mais segurança e velocidade na comunicação.

Não se conhece, hoje, nenhuma forma de ataque por criptoanálise ao AES que o tenha quebrado (TANENBAUM, 2003). Ainda que a NSA (*National Security Agency*), órgão do governo norte-americano especializado em decifrar códigos, que é o maior empregador de matemáticos e criptólogos do mundo, consiga construir uma máquina com 1 bilhão de processadores paralelos, cada um capaz de avaliar uma chave por picossegundo, tal máquina levaria cerca de 1010 anos para pesquisar o espaço de chaves (TANENBAUM, 2003).

2.9.2. Algoritmos que utilizam técnicas de criptografia assimétrica

Nesta subseção, são descritos os algoritmos assimétricos estudados no processo de desenvolvimento da aplicação de proteção de trilhas de auditoria.

2.9.2.1. RSA

O algoritmo RSA (KALISKI, 1998) foi criado por Ron Rivest, Adi Shamir e Len Adleman, sendo amplamente usado em sistemas bancários na internet com uso de chaves de 1024 bits de tamanho, praticamente impossíveis de quebrar na prática. O RSA também traz a flexibilidade de possibilitar que essa chave seja dobrada para 2048 bits, caso necessário.

Com o uso de chave pública e chave privada, o RSA promove a garantia da segurança tanto no tamanho da chave que pode ser gerada, quanto na confiabilidade que promove a comunicação. O problema do RSA é a demora na geração das chaves. Contudo, isto só é realizado num primeiro e único momento, na negociação da chave simétrica que será usada para encriptação dos dados na transmissão, utilizada para cifrar o *login* e a senha do usuário, como ocorre em páginas seguras que utilizam o HTTPS (TANENBAUM, 2003). Como consequência, o RSA é lento demais para codificar grandes volumes de dados, mas é amplamente empregado para a distribuição de chaves.

Também existem outros algoritmos assimétricos como o Diffie-Hellman e o DAS. No entanto, estes não são completos como o RSA que permite a troca de chaves e a assinatura digital.

2.9.2.2. Outros algoritmos assimétricos

Outros exemplos de algoritmos que utilizam o sistema assimétrico são os criados por: El Gamal (1985) e Schnorr (1991) que se baseiam na dificuldade de calcular logaritmos discretos e os que se baseiam em curvas elípticas (Menezes e Vanstone, 1993), mas as duas principais categorias são aquelas que estão calcadas na dificuldade de fatorar números extensos e no cálculo de logaritmos discretos, cuja base é um número primo extenso (TANENBAUM, 2003). Estes algoritmos não foram avaliados nos testes de desempenho por sua complexidade de implementação ou pelo baixo desempenho. Assim sendo, atualmente, o RSA é o algoritmo assimétrico mais amplamente utilizado.

2.10. Considerações Finais

Neste capítulo, foram apresentados os principais conceitos dos temas abordados no presente trabalho. Dentre os conceitos destacados estão: conceito de informação do ponto de vista empresarial, critérios da segurança da informação, o processo de auditoria trilhas de auditoria, técnicas criptográficas, conceitos de gerenciamento das chaves criptográficas, principais tipos de algoritmos criptográficos. Os referidos conceitos servem como base teórica para o desenvolvimento dos capítulos seguintes.

3. UM MODELO DE PROTEÇÃO PARA TRILHAS DE AUDITORIA

O processo de gestão da segurança da informação engloba o processo de auditoria interna. Em sistemas automatizados, a auditoria faz uso das trilhas de auditoria para a identificação de violações no tratamento das informações. Cumpre lembrar que as trilhas de auditoria são registros que armazenam informações sobre acesso e operações de usuários e administradores. Manter os registros íntegros é de fundamental importância para um processo de auditoria, ou ainda para identificar tentativas de fraudes. Este trabalho apresenta um método para proteção de trilhas de auditoria, empregando técnicas criptográficas juntamente com um processo de distribuição e armazenamento seguro das chaves utilizadas na encriptação, as quais são mantidas sob responsabilidade de diferentes administradores. Este capítulo propõe um modelo de proteção de trilhas de auditoria.

O capítulo está organizado da seguinte forma: primeiramente, é apresentada a justificativa para a construção do modelo, assim como a sua descrição geral e os seus processos de encriptação e decriptação (seções 3.1, 3.2, 3.3 e 3.4, respectivamente). Em continuidade, é discutida a utilização, a distribuição e o armazenamento das chaves criptográficas, a escolha e a definição dos algoritmos criptográficos adotados, bem como a implementação do modelo (seção 3.5, 3.6 e 3.7, respectivamente). Por fim, é feita uma análise do modelo frente aos trabalhos relacionados e a apresentação das considerações finais do capítulo (seções 3.8 e 3.9, respectivamente).

3.1. Justificativa para Construção de Modelos de Proteção de Trilhas de Auditoria

Um sistema tradicional de criptografia para os registros de *log*, que compõem uma trilha de auditoria, é significativo em um sistema de auditoria, mas não dá garantia que o sistema não seja violado. Uma vulnerabilidade em modelos tradicionais é que o segredo das trilhas (chave criptográfica) fica sob a responsabilidade de uma única pessoa, contudo, se ela estiver mal intencionada, poderá alterar dados no sistema alvo e, depois, apagar os seus rastros. Tradicionalmente, um colaborador da empresa, com permissão de administrador do sistema, fica com a chave para a descifragem do registro de *log*.

Outra vulnerabilidade bastante comum é a existência de um único arquivo de *log*, o que facilita a alteração das trilhas de auditoria sem detecção futura. Esta vulnerabilidade é, frequentemente, explorada por atacantes que invadem o servidor utilizando o perfil de

administrador do sistema. Após a invasão e a ação indevida, o atacante altera as informações da trilha de auditoria (*logs*) e sai do sistema sem deixar qualquer vestígio de suas ações.

Tais vulnerabilidades podem ser acentuadas com a dificuldade de manter a segurança na distribuição das chaves, demonstrando, com isso, que mesmo com um sistema eficiente de criptografia (criptografia forte), o sistema de trilhas de auditoria ainda pode estar vulnerável a certos tipos de ataques, justificando a proposição do modelo para a proteção de trilhas de auditoria.

3.2. Descrição Geral do Modelo de Proteção Proposto

A segurança na manutenção de registros de *logs* permite evitar que adulterações indevidas sejam realizadas nas trilhas de auditoria. A utilização de técnicas criptográficas aplicadas nos registros da trilha de auditoria, juntamente com um processo seguro para o armazenamento e a distribuição das chaves usadas para encriptação e decriptação dos *logs*, é o ponto central do modelo proposto para a proteção de trilhas de auditoria. O modelo trabalha com um sistema de criptografia híbrido, unificando técnicas de criptografia assimétrica e simétrica para a proteção dos registros de *logs* (geralmente uma *string* com informações do tipo: data, hora, identificação do operador, ação realizada, etc.).

No modelo proposto, neste trabalho, o processo para a segurança das trilhas inicia com a obtenção do registro de *log* recebido da aplicação a ser auditada e utiliza dois tipos de chaves, que são: chave simétrica e assimétrica. A chave simétrica é usada para encriptação e decriptação dos registros de *logs* e a chave assimétrica é utilizada para encriptar a chave simétrica. Durante o processo de encriptação dos *logs*, as chaves simétricas são geradas de forma aleatória. A chave assimétrica do tipo pública é utilizada para a encriptação da chave simétrica e, depois, é descartada. A chave assimétrica pública é adotada mais de uma vez para encriptar as chaves simétricas e pode ser renovada. A chave simétrica é gerada a cada novo registro de *log*. A chave assimétrica do tipo privada é utilizada para a recuperação da chave simétrica e pode ser armazenada em um local seguro, assim como a chave simétrica.

Ao final do processo de encriptação, têm-se dois segredos, sendo que um é a cifra da chave simétrica (aleatória e desconhecida mesmo pelo administrador de sistema, que programa os instantes de geração dos *logs*) e o outro é a chave privada (utilizada e conhecida por um auditor) que será necessária para a recuperação da chave simétrica e, consequentemente, a recuperação das trilhas encriptadas.

Estes segredos são, então, distribuídos através de um canal de comunicação confiável e seguro (dados novamente criptografados na entrada e decriptografados na saída) e são armazenados em diferentes bases de dados, conforme é descrito na seção 3.5. As seções 3.3 e 3.4 explicam como é realizado o processo de encriptação e decriptação, respectivamente.

3.3. Processo de Encriptação

O processo de encriptação consiste em transformar um registro de *log* da aplicação auditada em uma cifra ilegível. A encriptação dos registros de *logs* é realizada com uma chave simétrica gerada em tempo de execução. Em seguida, a chave simétrica é encriptada com uma chave assimétrica pública para, depois, ser distribuída e armazenada.

Seja o registro de *log* representado por log, a chave simétrica gerada pelo método representado por SK e o processo de encriptação representado por E, obtêm-se o *log* cifrado a partir de:

$$C_{log} = E_{SK}(log).$$

Após obter a cifra do log (C_{log}), gerada com o uso da chave simétrica SK, realiza-se o processo para o armazenamento seguro da chave simétrica, o qual é representado por:

$$C_{SK} = E_{PbK}(SK),$$

onde a chave simétrica SK é cifrada com a chave assimétrica pública E_{PbK} . Dessa forma, obtém-se uma cifra da chave simétrica (C_{SK}). Após esse processo, a chave assimétrica privada e a cifra da chave simétrica C_{SK} são armazenadas em base de dados distintas. Este processo de encriptação está representado na figura 1.

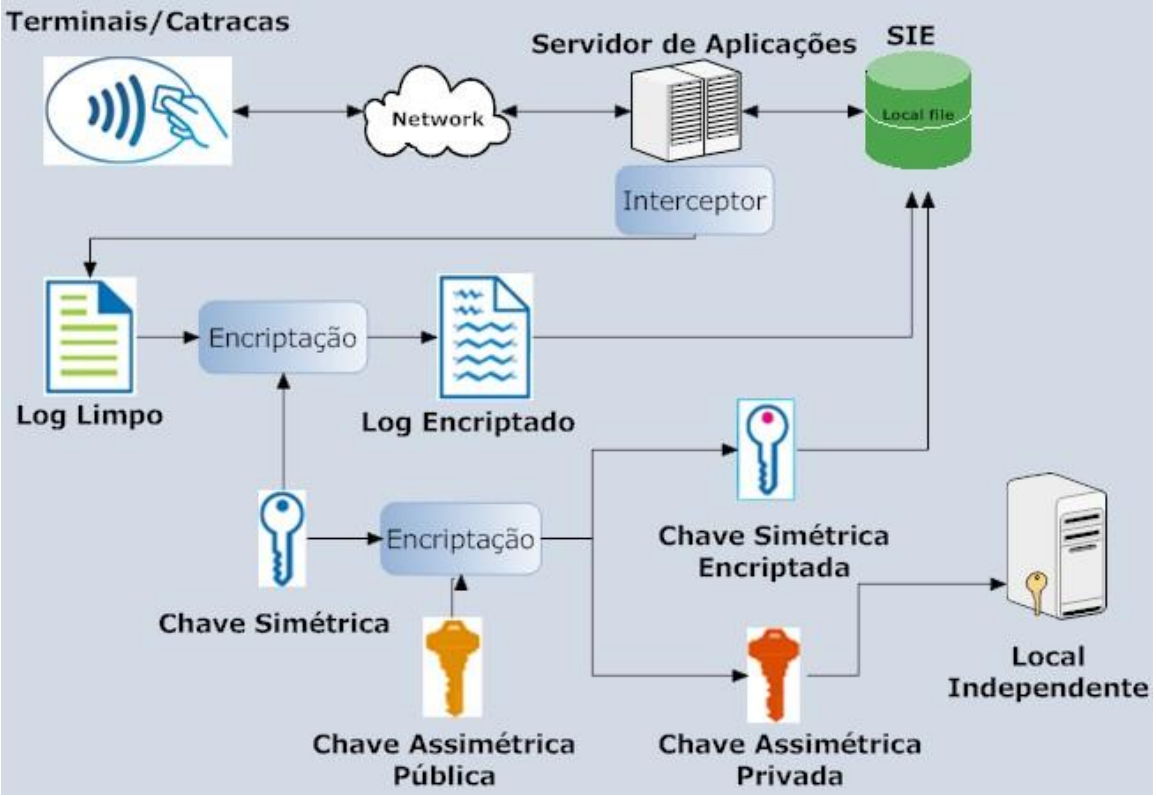

Figura 1: Processo de encriptação.

3.4. Processo de Decriptação

A decriptação consiste na utilização de chaves criptográficas para descriptografar os registros de *log* cifrados. Para a execução desse processo, inicialmente, é necessário recuperar a chave assimétrica privada da base de dados para, com ela, decifrar a chave simétrica que foi usada para criptografar o registro do *log* e que está armazenada em outra base de dados.

Seja a cifra da chave simétrica C_{SK}, a chave assimétrica privada gerada pelo método representada por PvK e o processo de decriptação representado por D, obtém-se a chave simétrica (SK) a partir de:

$$SK = D_{PvK}\,(C_{SK}).$$

A chave simétrica é, então, utilizada para decifrar o *log* encriptado, numa operação representada por:

$$log = D_{SK}\,(C_{log}),$$

onde C_{log} corresponde a cifra do registro de log e $D_{SK,}$ o processo de decriptação com a chave simétrica recuperada. Desse modo, recupera-se o registro de auditoria original (log).

3.5. Utilização, Distribuição e Armazenamento das Chaves Criptográficas

A proteção de *logs* de auditoria é de extrema relevância para a manutenção da segurança dos ativos informacionais de uma instituição. Dessa forma, utilizar métodos eficientes para evitar que adulterações sejam realizadas é imprescindível. No entanto, a segurança de uma informação confidencial, como os registros de *log*, não depende somente da eficiência dos mecanismos de criptografia utilizados, tendo em vista que a segurança da informação também depende da forma em que as chaves utilizadas para a proteção das informações são geridas, isto é, manipuladas, distribuídas e armazenadas. Além disso, a responsabilidade dos administradores dos sistemas em manter os mecanismos de transmissão e armazenamento das chaves também é outro fator de vital importância para conservar a integridade dos registros de auditoria. Esta seção descreve como o modelo de proteção de trilhas de auditoria manipula as chaves criptográficas.

3.5.1. Utilização das Chaves Criptográficas

Para maior segurança do sistema, o modelo proposto gera uma nova chave simétrica para cada entrada do registro de *log*, a fim de ser utilizada no processo de encriptação dos *logs*. Esta característica reduz a possibilidade de um atacante obter uma das chaves criptográficas e conseguir comprometer uma quantidade significativa ou toda a informação cifrada (ABNT NBR 17799, 2003).

Outra possibilidade é utilizar a mesma chave em blocos de informações. Esta solução diminui o custo de geração das chaves, mas não o custo de criptorafia da informação. Além disso, expõe uma quantidade maior de dados, caso uma das chaves seja exposta.

A chave simétrica utilizada na encriptação das informações é criptografada por uma chave pública assimétrica que é gerada pela aplicação. A chave simétrica poderá ser assim recuperada pela chave privada assimétrica gerada juntamente com a chave pública. Esta solução tem baixo custo computacional, pois a adoção de chave assimétrica para criptografar

grandes volumes de dados, caso das cifras dos *logs*, aumenta muito os custos computacionais do processo de criptografia e descriptografia (TANENBAUM, 2003).

O uso de chaves assimétricas e simétricas em conjunto provê, assim, maior robustez e flexibilidade ao sistema. A utilização dos dois modelos permite que se tenham dois segredos distintos para a recuperação da informação que são a chave simétrica, criptografada pela chave assimétrica pública, e a chave assimétrica privada, que será necessária para a recuperação da chave simétrica criptografada.

3.5.2. Distribuição das Chaves Criptográficas

A distribuição das chaves criptográficas é realizada através de um canal de comunicação baseado no protocolo TCP (*Transmission Control Protocol*) com o uso do protocolo SSL (*Secure Sockets Layer*), com o propósito de transmiti-las de forma segura. Este modo de transmissão de dados permite garantir a autenticação mútua entre servidor e cliente, bem como a integridade e a confidencialidade da mensagem no meio de comunicação usado para transportar os segredos criptográficos e as cifras dos *logs* (XU *et al*. 2005). O SSL é um protocolo de segurança criado para prover autenticação e encriptação sobre redes TCP/IP, inclusive na internet. No modelo proposto, utiliza-se o SSL sobre o protocolo TCP, pois se trata do principal protocolo de transporte da Internet (YU *et al*. 2009). No protocolo SSL, clientes e servidores podem autenticar-se e, em seguida, trocar dados cifrados entre si.

O SSL possui algumas limitações de segurança devido às suas características e propósitos fundamentais. Entre as suas limitações, está o fato de não assegurar a proteção dos dados locais (XU *et al*. 2005). O SSL somente garante a segurança dos dados durante a sua transmissão entre duas aplicações.

3.5.3. Armazenamento das Chaves Criptográficas

Para que o modelo torne-se confiável, é preciso armazenar as chaves criptográficas de forma segura, o que nem sempre é possível, pois fatores além dos computacionais, como o fator humano, interferem neste processo. O modelo proposto neste trabalho trabalha com duas chaves para a cifragem e a descifragem das informações, assim, propõe-se que a cifra da chave simétrica fique em uma base de dados que esteja sob responsabilidade de um administrador local, enquanto a chave assimétrica privada fique armazenada em um

dispositivo seguro como um *smart card* ou CD ou mesmo em outra base de dados, sempre sob responsabilidade de um auditor externo.

O processo de armazenamento proposto está representado na figura 2. Observa-se a necessidade do uso de duas chaves para o acesso ao conteúdo do *log*, o que permite um intertravamento das chaves para o acesso às informações. O *log* cifrado é gerado de forma independente ao gerado pelo sistema do qual se quer proteger o *log*. Junto com o processo de uso das chaves descritas na figura 1, este processo provê maior segurança para o sistema, pois, para obter acesso às informações das trilhas, são necessárias que as duas bases de dados sejam acessadas. Dessa forma, mesmo as ações feitas por um atacante, com direitos de super-usuário, que pode acessar e alterar os registros de *log* originais da base de dados, não conseguirá modificar o segundo registro de *log*, pois, para isso, ele terá que obter, além da chave local, a chave do auditor externo.

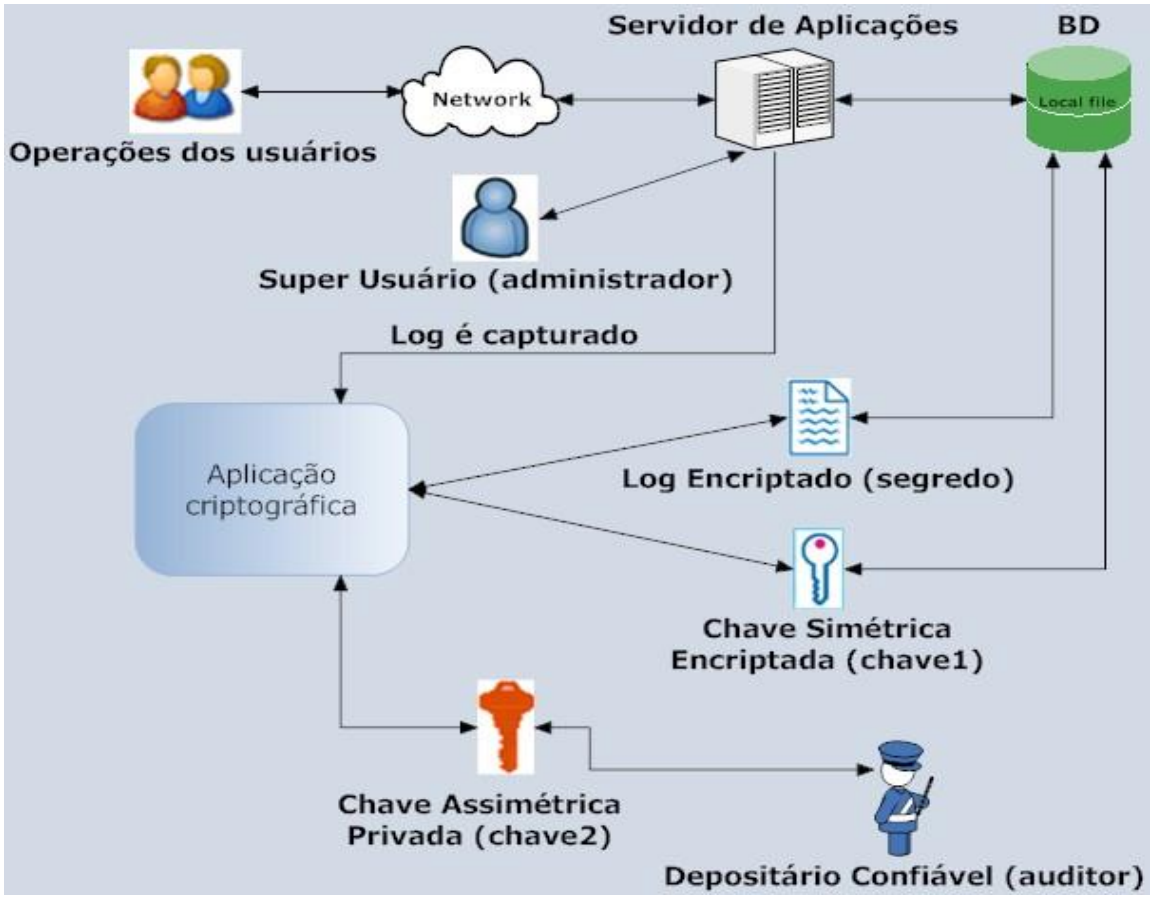

Figura 2: Processo de codificação do *log* e distribuição das chaves.

3.6. Definição dos Algoritmos Criptográficos

Os algoritmos criptográficos dividem-se em dois grupos: os algoritmos que utilizam técnicas de criptografia simétrica (AES, RC2, RC4, DES e Triple DES) e os algoritmos híbridos que são formados pela criptografia simétrica e pelo algoritmo assimétrico RSA.

Nas implementações realizadas, o algoritmo AES faz uso de um bloco de tamanho fixo de 128 bits com uma chave de mesmo tamanho, e o algoritmo DES utiliza uma chave de 56 bits. Por sua vez, o algoritmo Triple-DES usa três chaves com tamanho de 64 bits, oferecendo maior segurança, sendo, porém, mais lento que o DES original. Os algoritmos RC2 e RC4 também utilizam chaves com tamanho de 128 bits.

A presente seção enfoca um comparativo entre os tempos de execução de diferentes algoritmos de criptografia e uma discussão dos resultados obtidos com a aplicação do método proposto neste capítulo. O objetivo é comparar o desempenho do processo de encriptação dos algoritmos e verificar se o método proposto possui um desempenho competitivo. Para a implementação dos algoritmos e do método proposto, optou-se pela linguagem de programação Java[2], juntamente com o ambiente de desenvolvimento integrado *NetBeans*[3].

Assim como os algoritmos estão divididos em dois grupos (simétricos e assimétricos), os resultados também são apresentados respeitando essa divisão. Os testes realizados consideram os tempos de execução somente do processo de encriptação dos registros de *log*. O equipamento utilizado nos testes foi um computador com processador Intel Core I7 com 4GB de memória DDR3 PC 1333MHz e sistema operacional Windows 7 *Ultimate*, empregando-se SDK JDK6 sem a execução paralela de outras aplicações.

A realização dos testes tabelas 1 e 2 deu-se com três tomadas de tempo para cada algoritmo. Cada teste executado possui um fluxo de 1000 registros encriptados, gerando uma média aritmética após as tomadas de tempo.

A Tabela 1 mostra os resultados dos testes utilizando os algoritmos simétricos juntamente com o algoritmo assimétrico RSA. Estes testes comparam o desempenho com a unificação de diferentes técnicas de criptografia, levando em consideração os aspectos de segurança de cada algoritmo. Os tempos de execução estão representados em segundos, sendo que se pode verificar uma variação maior de tempo na execução conjunta do algoritmo

[2] http://www.java.com/pt_BR/

[3] http://netbeans.org/

assimétrico RSA com o algoritmo simétrico AES, 59,826 segundos contra 58,978 do algoritmo assimétrico RSA com o algoritmo simétrico RC4. Esta variação de tempo justifica-se pelo processo criptográfico que cada algoritmo utiliza.

Tabela 1 - Resultados dos testes com algoritmos simétricos junto com o algoritmo RSA.

RSA+AES	RSA+RC2	RSA+RC4	RSA+DES	RSA+ Triple-DES
59,826s	59,259s	58,978s	59,306s	59,186s

A Tabela 2 apresenta os resultados dos testes usando os algoritmos simétricos. Os testes somente com esses algoritmos mostram o desempenho das criptografias com as diferentes características de segurança que cada algoritmo possui. Os tempos de execução estão representados em segundos e demonstram que os algoritmos simétricos RC2 e RC4 são os mais rápidos.

Tabela 2 - Resultados dos testes com algoritmos simétricos.

AES	RC2	RC4	DES	Triple-DES
0,483s	0,416s	0,379s	0,452s	0,484s

Os tempos de execução demonstrados nas Tabelas 1 e 2 correspondem ao processo de encriptação dos registros de *logs*. Uma comparação simplificada entre os tempos mostra que a aplicação de um método, que emprega somente um algoritmo simétrico possui um maior desempenho. No entanto, o uso de um algoritmo simétrico, utilizando uma única chave para cifrar e decifrar os dados torna o processo mais vulnerável, pois, ao obter acesso à chave simétrica (que precisa ser trocada entre diferentes computadores), os registros são facilmente decifrados.

Os tempos resultantes dos testes com algoritmos simétricos evidenciam que os algoritmos RC2 e RC4 obtiveram maior desempenho no processo de encriptação. Porém,

Fluhrer *et al.* (2001) afirmam que algumas chaves do algoritmo RC4 são fracas, tornando-o inseguro. O algoritmo DES, em sua forma original, já não é mais seguro (TANENBAUM, 2003). No entanto, a sua forma modificada, o 3DES, mantém um bom nível de segurança, ainda que seja mais lento.

Conforme a Tabela 2 não houve uma expressiva variação de tempo dos algoritmos testados. Com isso, optou-se pela utilização do algoritmo AES no método proposto na seção 2 por suas características de segurança e pela sua ampla utilização prática.

O RSA é considerado um algoritmo robusto, pois sobreviveu a todas as tentativas de rompimento por um longo tempo (TANENBAUM, 2003). A sua utilização, juntamente com algoritmos simétricos, torna o processo de encriptação mais seguro. Os resultados dos testes do RSA+AES (Tabela 1) não provocam significativa variação de tempo entre os conjuntos de algoritmos testados, o que possibilita uma segurança mais robusta em função de se utilizar as características de segurança e velocidade dos algoritmos RSA e AES, respectivamente, sem um aumento significativo no tempo de processamento dos dados.

3.7. Descrição do Algoritmo Desenvolvido neste Trabalho

Nesta seção, são apresentadas e descritas às implementações dos métodos de geração de chaves, criptografia e descriptografia do algoritmo de proteção de trilhas de auditoria desenvolvido neste trabalho.

3.7.1. Processo de criptografia

O processo de criptografia da trilha de auditoria inicia-se com a chamada da aplicação denominada *JavaCrypt* onde é transferido um *string* com a trilha a ser cifrada. Para isso, é necessário criar uma instância da classe *logs* e executar o método *GravaLog,* passando o texto para criptografar como parâmetro.

Na execução do método *GravaLog,* é gerada uma instância da classe *GeradorParChaves*, classe responsável por gerar o par de chaves (pública e privada) para, então, usá-las no processo de criptografia. Neste caso, o método a ser usado é denominado *geraParChaves*. O processo para geração do par de chaves é demonstrado na figura 3.

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance ("RSA");
kpg.initialize (new RSAKeyGenParameterSpec(RSAKEYSIZE, RSAKeyGenParameterSpec.F4));
KeyPair kpr = kpg.generateKeyPair();
PrivateKey priv = kpr.getPrivate();
PublicKey pub = kpr.getPublic();
```

Figura 3: Rotina para gerar o par de chaves

Conforme demonstra a figura 3, a instância da classe *KeyPairGenerator* produz um objeto que implementa o algoritmo de criptografia especificado em parâmetro, no caso presente, utiliza-se o algoritmo de criptografia RSA. Tendo esse objeto, instancia-se a classe *KeyPar* para obter o par de chaves, uma pública e outra privada.

Com as chaves geradas, o método cria, na memória, dois arquivos, para, a seguir, gravar as chaves geradas neles. Em um arquivo, armazena a chave pública e, em outro arquivo, a chave privada, os nomes desses arquivos são recebidos como parâmetro no método *geraParChaves*.

Na sequência da rotina implementada na classe *MainCriptografia*, a chave pública é recuperada no arquivo criado na memória e, em continuidade, é criada uma instância da classe *Cifrador* onde está o método *cifra* responsável por criptografar o texto recebido como parâmetro.

Na execução do método *Cifrador.cifra*, é gerada outra chave secreta utilizando a classe *KeyGenerator* que gera um objeto que implementa o algoritmo simétrico AES e o seu tamanho é definido em 128 bytes. Essa chave simétrica serve para criptografar o texto recebido como parâmetro. Após isso, a chave pública recebida, nesse método, é utilizada para criptografar a chave simétrica. O método retorna o texto criptografado com a chave simétrica e a chave simétrica criptografada com a chave pública, em formato de uma matriz de *bytes*. O código que executa esta operação é demonstrado na figura 4.

```
byte[] textoCifrado = null;
byte[] chaveCifrada = null;

//--Gerando uma chave simétrica de 128 bits
KeyGenerator kg = KeyGenerator.getInstance ("AES");
kg.init (128);
SecretKey sk = kg.generateKey ();
byte[] chave = sk.getEncoded();
//--Cifrando o texto com a chave simétrica gerada
Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
aescf.init (Cipher.ENCRYPT_MODE, new SecretKeySpec (chave, "AES"), ivspec);
textoCifrado = aescf.doFinal (textoClaro);
//--Cifrando a chave com a chave pública
Cipher rsacf = Cipher.getInstance ("RSA");
rsacf.init (Cipher.ENCRYPT_MODE, pub);
chaveCifrada = rsacf.doFinal (chave);
```

Figura 4: Código do cifrador

Após o retorno do *log* cifrado e das chaves simétrica cifrada e assimétrica pública para o método *GravaLog*, as mesmas serão distribuídas e gravadas em seus respectivos bancos de dados, neste caso, com o uso do *framework* Hibernate[4].

3.7.2. Processo de descriptografia

O processo de descriptografia da trilha de auditoria inicia-se com a chamada do método *LeLog* da aplicação *JavaCrypt* através da criação de uma instância da classe *logs*. Na sequência, o método *LeLog* através do *framework* Hibernate recupera as trilhas armazenadas na base de dados, juntamente com a chave simétrica criptografada. Posteriormente, é solicitada a chave assimétrica do repositório confiável para se iniciar o processo de recuperação das trilhas cifradas. O segmento de código responsável por isso é apresentado na figura 5.

[4] http://www.hibernate.org/

```
public class Decifrador {
     public byte[] decifra (PrivateKey pvk, byte[] textoCifrado, byte[] chaveCifrada) throws NoSuchAlgorithmException,
    NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException,
    BadPaddingException, InvalidAlgorithmParameterException  {
        byte[] textoDecifrado = null;

        //-- A) Decifrando a chave simétrica com a chave privada
        Cipher rsacf = Cipher.getInstance ("RSA");
        rsacf.init (Cipher.DECRYPT_MODE, pvk);
        byte[] chaveDecifrada = rsacf.doFinal (chaveCifrada);
        //-- B) Decifrando o texto com a chave simétrica decifrada
        Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
        IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
        aescf.init (Cipher.DECRYPT_MODE, new SecretKeySpec (chaveDecifrada, "AES"), ivspec);
        textoDecifrado = aescf.doFinal (textoCifrado);

        return textoDecifrado;
    }
}
```

Figura 5: Código do decifrador

Os dados recuperados são exportados para um arquivo texto, ficando à disposição dos auditores.

3.8. Discussão Comparativa com Outros Trabalhos

A segurança nos registros de *logs* é fundamental para os modelos organizacionais atuais. Esta seção apresenta alguns trabalhos existentes que buscam garantir a segurança em trilhas de auditoria, a fim de confrontá-los com o modelo proposto no presente trabalho.

Uma das abordagens para proteção de registros de *log* é a utilização de um *host* remoto que armazena as informações, evitando a violação por *hackers* no sistema local (WU et al. 2010). Nesse método, uma cópia idêntica do *log* é direcionada ao *host* remoto. Outra técnica possível é a distribuição das trilhas de auditoria para múltiplas máquinas, replicadas na rede, sendo que, para a violação da integridade, os atacantes precisam comprometer mais de uma máquina (WU *et al*. 2010). Contudo, tais abordagens fazem uso de vários recursos de *hardware*, demandando de um excessivo uso da rede e, ademais, possuem um alto custo para implementação, não atentando para atacantes internos, tais como administradores maliciosos.

Bellare e Yee (1997 e 2003) definiram integridade em um sistema de *logs* de auditoria propondo uma nova propriedade de segurança denominada *forward integrity*. Esta propriedade é baseada no uso de MAC (*Message Authentication Code*) e não utiliza a replicação em *host* remoto. A ideia é prevenir a alteração ou a inserção de informações pelo atacante, mesmo quando os registros de *log* ficam disponíveis, no caso do atacante conseguir

o controle total do sistema. Para garantir proteção, vários registros de *log* são indexados e identificados de forma independente em um intervalo de tempo. No final do intervalo de tempo, outro registro de *log*, definido como entrada especial, contendo o número de registros de *log* no período atual, é criado para indicar o fim desse intervalo de tempo. Tornando, desse modo, a manipulação dos registros de *logs* restrita ao intervalo de tempo corrente, a técnica impede que um atacante, que obtenha a chave, possa adulterar registros gerados em intervalos anteriores. Apesar disso, de posse da chave MAC, o atacante poderá forjar novas entradas de *logs*, sendo possível comprometer processos de auditoria.

Uma proposta que incrementa o uso de MAC (SCHNEIER e KELSEY, 1998 e 1999) é usar cadeias de *hash* unidirecional para interligar os registros da trilha de auditoria após cada entrada de *log*. Ao contrário da proposta de Bellare e Yee, que indexa e identifica os registros de *log* em um intervalo de tempo, Schneier e Kelsey interligam os *logs* em uma cadeia de *hash* segura após cada registro de entrada. Cada registro de *log* possui uma ligação, realizada para autenticar todas as entradas prévias. No trabalho de Schneier e Kelsey, após a criação e o encerramento de um registro de *log*, eles são enviados para uma máquina confiável, mantendo, assim, a necessidade de investimento adicional. Schneier e Kelsey desenvolveram também um mecanismo criptográfico para proteger os *logs* de leituras não autorizadas. O mecanismo é baseado em criptografia simétrica, com um servidor central de segurança, que armazena as chaves criptográficas. Dessa forma, caso um atacante invada a máquina com as chaves, ou o segredo da chave esteja com apenas uma pessoa que também tenha acesso ao sistema, ele conseguirá acesso aos registros de *logs*, podendo adulterá-los.

Holt (2006) propôs o modelo *Logcrypt* baseado no modelo de Schneier e Kelsey (1998), adicionando a configuração de chaves públicas. A criptografia de chave pública permite criar assinaturas com uma chave e verificá-las com outra diferente. Estas assinaturas podem ser usadas no lugar de MAC para permitir a verificação de registro sem a possibilidade de modificá-lo, bem como possibilitar a publicação da chave inicial, usada para criar o *log*, pois só a chave pública é necessária para a verificação. Em seu trabalho, Holt não aborda a forma como é feita a distribuição das chaves utilizadas para cifrar os registros de log.

Em (WU *et al*. 2010), um novo esquema de proteger os *logs* com criptografia e assinatura digital foi proposto. O esquema indica o uso de assinatura digital assimétrica baseado em *forward-secure*, mesmo que a chave privada do assinante seja exposta, o atacante não consegue obter as mensagens cifradas em um tempo anterior, por isso, podendo cumprir a

criptografia *forward-secure*. Esse modelo tem a capacidade de separar a assinatura da verificação e separar a criptografia da descriptografia.

O método proposto, neste trabalho, é composto por um sistema de criptografia híbrido. O uso de algoritmos assimétricos com algoritmos simétricos trata a vulnerabilidade destacada no trabalho de (SCHNEIER e KELSEY, 1998 e 1999), que utilizam um mecanismo baseado em criptografia simétrica. Além disso, o método gera as chaves criptográficas em tempo de execução para cifrar cada registro de *log* e distribui-as através de um canal de comunicação confiável baseado no protocolo SSL que permite criar um canal de comunicação seguro para a transmissão dos segredos, armazenando-as em bases de dados diferentes e independentes sob responsabilidade de diferentes autoridades que são o administrador do sistema e um auditor externo. Dessa forma, dificulta-se o acesso a todas as chaves utilizadas no processo de encriptação, prevenindo que novos registros de *logs* sejam forjados, conforme a vulnerabilidade identificada no trabalho de Bellare e Yee (1997 e 2003). O método proposto, no presente trabalho, renova as chaves criptográficas simétricas a cada novo registro de *log* e mantém a chave assimétrica pública que criptografa as chaves simétricas. Assim sendo, o processo de criação das novas chaves fica menos custoso se comparado com a proposta de Wu (2010), o qual trabalha com assinatura digital.

3.9. Considerações Finais

Este capítulo apresentou um modelo para a manutenção segura de trilhas de auditoria. O modelo é baseado na cifragem dos registros de *log*, que explora a combinação dos algoritmos simétrico (AES) e assimétrico (RSA), no uso de trilhas independentes (redundância no armazenamento dos registros), bem como na divisão de responsabilidades na guarda das chaves.

A combinação dos algoritmos AES e RSA é que torna viável e necessário o uso de duas chaves (segredos) para obter-se o acesso à trilha externa. As chaves são enviadas e armazenadas de forma segura em diferentes locais (dois na proposta do modelo), sendo que uma fica sob a responsabilidade do administrador local e a outra torna-se responsabilidade de um auditor externo. Este intertravamento garante que se a máquina local ou a máquina externa fique exposta e sofra alguma violação seja por parte de pessoal autorizado ou um atacante externo, em nenhum dos casos, eles conseguirão ler e modificar a trilha externa para apagar seus rastros.

4. PROCESSO DE DESENVOLVIMENTO DO SISTEMA IDENTIFICAÇÃO ELETRÔNICA E VALIDAÇÃO DO MODELO DE PROTEÇÃO DE TRILHAS DE AUDITORIA

No presente capítulo, é traçada a estrutura operacional do Restaurante Universitário da UFSM. O conhecimento desta estrutura serve para a apresentação da API desenvolvida para o sistema de Identificação Eletrônica, produzido para o restaurante Universitário da UFSM, assim como para a validação do modelo de trilhas de auditoria. A validação do modelo de trilhas de auditoria foi feita junto ao Centro de Processamento de Dados da Universidade Federal de Santa Maria. O sistema alvo da validação foi o de créditos e débitos do sistema de pagamento de refeições do Restaurante Universitário da instituição. O objetivo da referida validação é consolidar o modelo como ferramenta de auditoria de forma a possibilitar que o Diretor do Restaurante Universitário e também o gestor financeiro tenham garantias sobre a integridade do sistema de trilhas de auditoria, responsável pelos sistemas de pagamentos recebidos e refeições servidas.

4.1. Dados sobre a Estrutura e Funcionamento do Restaurante Universitário

O Restaurante Universitário da Universidade Federal de Santa Maria conta com três unidades sendo a primeira inaugurada no ano de 1963 no centro de Santa Maria. O RU do *Campus* foi inaugurado 12 anos depois, em 1975. Em 2010, iniciaram as atividades do RU - Refeitório 2, no *Campus* Camobi. Os serviços do Restaurante eram terceirizados e a produção dos alimentos ocorria em cada uma de suas sedes. A partir de 1984, o RU passou a ser administrado pela UFSM, sendo vinculado e coordenado pela Pró-Reitoria de Assuntos Estudantis/PRAE (RU, 2011). O RU possui um sistema de compras e administração centralizado, sendo assim passou a funcionar como Órgão Suplementar Central da UFSM.

A partir de 1997, a produção passou a concentrar-se no RU *Campus* Refeitório I. Também neste período, começou a implantação do sistema de informatização de acesso ao Restaurante - SRU, através de carteiras com códigos de barra. O RU adotou uma nova metodologia gerencial que inclui a informatização dos processos administrativos.

4.2.1. Descrição de Cada Unidade

Nesta subseção, acham descritas as características funcionais e físicas de cada unidade do Restaurante Universitário da UFSM. As informações apresentadas foram coletadas a partir de visitas as instalações do RU, assim como encontram-se baseadas em dados coletados em reuniões com responsáveis pelo sistema SRU (funcionários do CPD/UFSM).

4.2.1.1. RU *Campus* - Refeitório I

No RU refeitório I, situam-se os setores de produção de alimentos, administração e nutrição, além de ser o local onde se encontram todos os equipamentos para a produção e o armazenamento dos alimentos. O Refeitório I conta com dois setores: um para os usuários, com quatro *buffets* e 700 lugares; outro, para os funcionários, com um *buffet* e 18 lugares (RU, 2011). Há, ainda, quatro caixas de recepção (roleta) e um caixa de venda de créditos que funciona, de segunda à sexta-feira, nos seguintes horários: das oito às 10 horas e das 11h às 14 horas.

O Refeitório I serve almoço, café da manhã e janta, sendo café e janta apenas para carentes.

4.2.1.2. RU *Campus* - Refeitório II

Com uma área total de 1.177 metros quadrados, o Refeitório II do Restaurante Universitário (RU) conta com dois *buffets* e 360 lugares. A comida servida é preparada na cozinha junto ao Refeitório I e transportada até o Refeitório II, da mesma forma que ocorre com a unidade do RU localizada no centro da cidade (RU, 2011). No Refeitório II, são servidos somente almoços, mas o refeitório conta com copa, cozinha, quatro caixas de recepção (roleta) e um caixa de venda de créditos que funciona, de segunda à sexta-feira das nove às 13h e 30 min.

4.2.1.3. RU Centro

O RU Centro não faz comida, apenas serve refeições preparadas no RU *Campus* que são transportadas até o Centro em contêineres térmicos, com exceção do desjejum que é

preparado no local. Possui um refeitório com um *buffet* e 120 lugares (RU, 2011). Há, ainda, um caixa de recepção (roleta) e um de venda de créditos, que funciona de segunda à sexta-feira nos seguintes horários: das oito às 10h, das 11h às 13h e 15 min e, à tarde, das 18 h às 19h e 45 min.

4.2.2. Colaboradores

Em 2011, nas três unidades do Restaurante Universitário, existia um total de 105 colaboradores (RU, 2011). O quadro de colaboradores é composto por pessoal técnico-administrativo, terceirizados e bolsistas. A Figura 7 ilustra o quadro geral de colaboradores. Dos 105 colaboradores, 90 trabalham nas unidades do *Campus* e 15 na unidade do Centro de Santa Maria. Destes, 55 são do quadro de carreira da UFSM (10 no Centro e 45 no *Campus*), 41 são terceirizados (37 no Campus e quatro no Centro) e nove são bolsistas (oito no Campus e um no Centro).

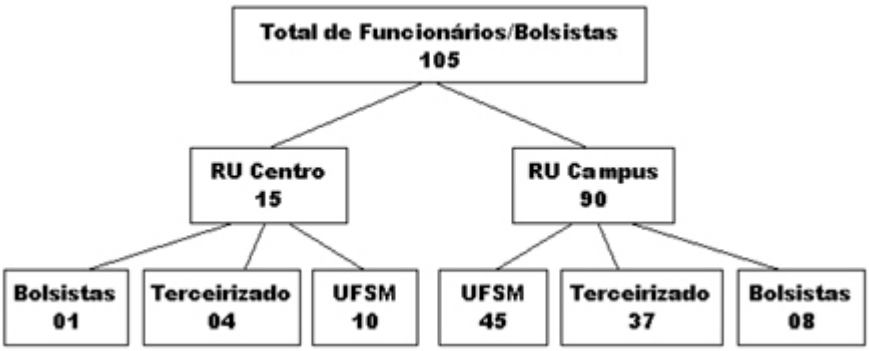

Figura 6: Corpo de colaboradores (RU, 2011).

A distribuição do pessoal é detalhada na Tabela 3, onde se pode perceber quais atividades de pagamento e registro das refeições são feitas por pessoal técnico-administrativo e terceirizado. As atividades do pessoal técnico-administrativo são direção do RU, manutenção do Sistema RU, atividades de cozinheiro, copeiro e faxineiro. Já os funcionários terceirizados desempenham as atividades de cozinheiro, copeiro e operadores de caixas de recepção e compra de créditos.

Tabela 3 - Distribuição de cargos nas unidades do RU.

Funções	UFSM		Terceirizado	
	Campus	Cidade	*Campus*	Cidade
Diretor	01	-	-	-
Vice-diretor	01	-	-	-
Contador	-	-	-	-
Assistente administrativo	01	-	-	-
Cozinheiro	15	04	11	-
Copeiro	08	03	15	03
Açougueiro	01	-	02	-
Chefe de cozinha	01	01	-	-
Nutricionista	02	-	-	-
Auxiliar de nutrição	03	-	-	-
Auxiliar de limpeza	-	-	01	-
Lavanderia	-	-	01	-
Mestre de ofício	02	-	-	-
Chefe de almoxarifado	01	-	-	-
Caldeirista	01	-	01	-
Motorista	02	-	-	-
Técnica em Nutrição	01	-	-	-
Recepcionista	02	01	-	-
Jardineiro	-	-	01	-
Vigilantes	03	-	-	-
Operador de caixa recepção	-	-	8	1
Operador de caixa venda	-	-	2	1
Total	45	9	46	5
	54		51	

4.2.3. Números Operacionais

O RU oferece a alunos, servidores técnico-administrativos e docentes da UFSM refeições a baixo custo. Para alunos carentes, a UFSM viabiliza, mediante prévia seleção, as três principais refeições com custo subsidiado em 80%. As duas unidades do Restaurante servem, em média, 5.300 refeições diárias. Na tabela 4 pode-se observar a distribuição das refeições, ficando evidente que o Almoço no *Campus*, ao servir em média 3.200 refeições no almoço, é o que mais exige do sistema.

Tabela 4 - Quadro de distribuição das refeições (RU,2011).

Refeição	RU Campus	RU Centro
Desjejum	700	100
Almoço	3200	450
Jantar	700	150

O valor pago pelas refeições é de R$ 0,20 pelo café da manhã e R$ 0,50 pelo almoço ou jantar para alunos carentes; e R$ 2,50 aos demais usuários, sendo que estes só têm direito ao almoço. A Tabela 5 sintetiza estas informações.

Tabela 5 - Valores das refeições (RU,2011).

Tipo / Usuário	Desjejum	Almoço	Jantar
Alunos carentes	R$ 0,20	R$ 0,50	R$ 0,50
Alunos não carentes	-	R$ 2,50	R$ 2,50
Servidores e Técnicos Administrativos	-	R$ 2,50	-
Docentes	-	R$ 2,50	-
Visitantes	-	R$ 2,50	-

4.2. O Cartão Universitário

Nesta seção, é apresentada a modelagem do sistema de pagamentos com o modelo atual baseado em código de barras. Também traz-se a modelagem do sistema de identificação com *smart cards* desenvolvida neste trabalho.

4.2.1. O Processo de Crédito e Débito Atual com Identificação Eletrônica por Código de Barras

O processo de identificação/pagamentos dos usuários é feito pelo Sistema de Informação de Acesso ao Restaurante, ou Sistema RU (SRU). A identificação, no SRU, é realizada através de carteiras com códigos de barras. Cada usuário deve carregar previamente os seus créditos no caixa disponível nos RUs. O valor é recolhido pelo funcionário e, por meio do código de barras da carteira, é validado/creditado o valor no SRU. Na hora da

refeição, o usuário dirige-se a um dos caixas de recepção que, com um leitor de código de barras, identifica-o. Após a identificação do usuário, o sistema debita o valor da refeição que se dá em função do tipo de vínculo e do horário da refeição. Por fim, o usuário tem a sua entrada liberada na catraca. A Figura 8 mostra um dos caixas do RU *Campus* - Refeitório I, onde este procedimento efetiva-se.

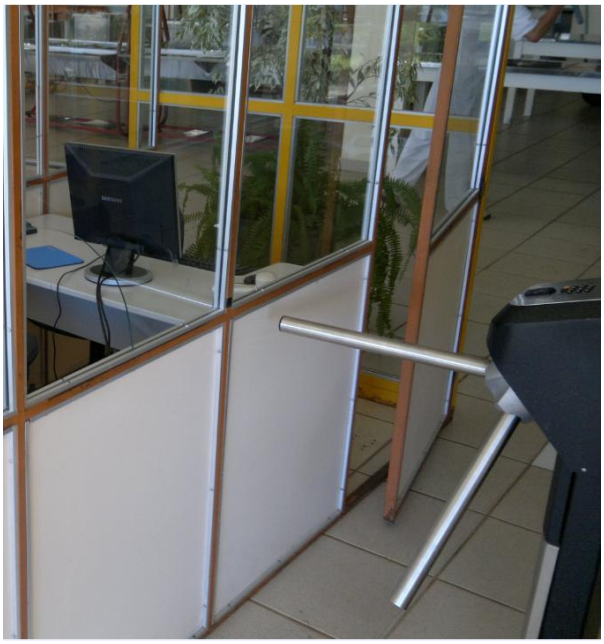

Figura 7: Caixa RU *Campus* - Refeitório I.

As informações das operações de créditos e débito são tratadas em tempo real, ou seja, cada pagamento gera uma transação no SRU e as informações são armazenadas em uma base de dados sob a responsabilidade dos administradores do sistema (funcionários do CPD).

Este modelo de Identificação Eletrônica tem algumas desvantagens com relação às outras tecnologias como a dos *smart cards*. A primeira desvantagem é a pouca segurança referente à clonagem das carteiras, visto que a clonagem pode ser feita por uma máquina de fotocópia colorida. Outra desvantagem é a falta de escalabilidade do cartão universitário, uma vez que há necessidade de integração de diferentes sistemas de pagamento e de identificação dentro e fora do *campus*. Esta limitação deve-se às características do sistema de código de barras que nem sempre é padronizado e possibilita somente o armazenamento de um identificador do usuário. Estas características limitam o uso da identificação por código de barras (que não é padronizado), que também serve para a identificação na retirada de livros das bibliotecas da instituição.

4.2.2. O Processo de Crédito e Débito com Identificação Eletrônica por *Smart Card*

Com a necessidade cada vez maior de integração de sistemas e serviços, também há uma demanda maior por mecanismos de segurança. Diante dessa constatação, este trabalho realizou a modelagem do sistema de identificação eletrônica baseada em *smart cards* para o sistema de pagamentos do Restaurante Universitários da UFSM, conforme descrito a seguir.

O processo de identificação/pagamentos dos usuários continua sendo feito através do Sistema de Informação de Acesso ao restaurante, ou Sistema RU (SRU), conforme ilustra a Figura 8. A identificação no SRU é realizada através de *smart card*. Cada usuário (aluno, servidor, visitante ou administrador) tem a sua identificação eletrônica num *smart card* e deve carregar previamente os seus créditos no caixa disponível nos RUs. O valor recolhido pelo funcionário é validado/creditado no SRU (transação de crédito). Na hora da refeição, o usuário dirige-se a uma das catracas automáticas que, por meio de sua identificação com o cartão, libera a entrada e desconta o valor da refeição que será cobrado em função do tipo de vínculo e do horário da refeição (transação de débito). O usuário também pode consultar os seus créditos (transação de leitura) junto a algum terminal de consulta. O administrador do sistema (servidor da instituição) fica responsável pelas alterações cadastrais dos usuários como cadastramento de identidades, descadastramento de identidades e manutenção do sistema.

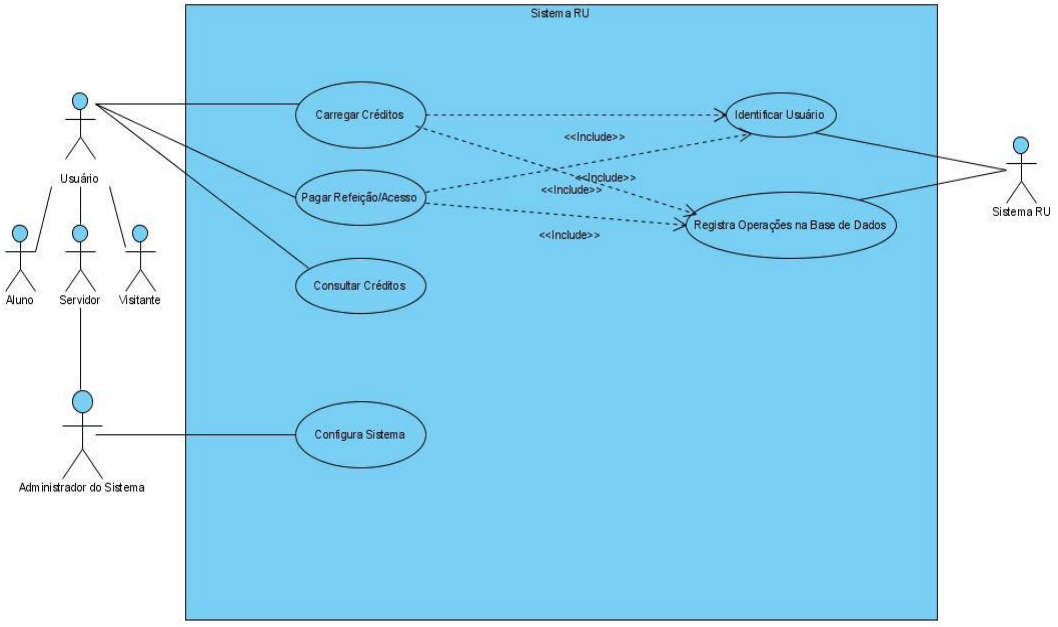

Figura 8: Caso de uso do sistema de identificação eletrônica do RU.

Dentre as informações definidas para identificação, estão o UID (identificador de *hardware*) do cartão e os dados a serem gravadas no cartão: ID do usuário no SIE e CPF. Outras informações como número de créditos não foram adicionadas para diminuir o fluxo de informações nas transações do sistema e, do mesmo modo, para fornecer maior segurança, no caso de violações nos terminais ou na hipótese de clonagem de algum cartão. A Figura 9 ilustra o diagrama de entidade-relacionamento do sistema de identificação eletrônica, onde as informações podem ser claramente identificadas.

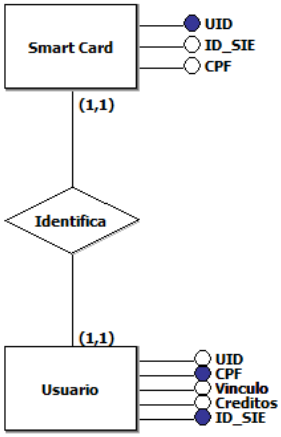

Figura 9: Diagrama Entidade-Relacionamento do Sistema de Identificação Eletrônica

Salienta-se que a utilização de *smart cards* no processo de identificação e pagamentos do SRU da UFSM traz diversas vantagens em relação ao uso de código de barras, que é uma representação gráfica de uma sequência de números. Dentre elas, pode-se apontar: i) maior segurança contra clonagens dos cartões, pois o mecanismo de UID do cartão e a criptografia utilizada para a leitura e escrita nos setores de memória do cartão dificultam e tornam o processo de clonagem desinteressante; ii) capacidade de armazenar informações na memória do *smart card*, proporcionando uma maior flexibilidade no uso do cartão de identificação; possibilidade de ser utilizado em mais de um serviço, mesmo que estes operem com sistemas diferentes, uma vez que se pode criptografar distintos setores da memória do cartão com diferentes chaves criptográficas.

4.3. API para leitura e escrita de informações no Smart Card

Para que a tecnologia de *smart card* pudesse ser utilizada no sistema de identificação eletrônica do Restaurante Universitário da UFSM, uma API (*Application Program Interface*) de leitura e escrita em *smart cards* foi desenvolvida. Esta seção apresenta o processo de desenvolvimento da API, assim como as informações sobre as tecnologias e a escolha do tipo de cartão encontram-se no Apêndice B deste trabalho.

4.3.1. Escolha da Tecnologia de *Smart Card* a ser Utilizada

Antes do desenvolvimento da API, foi necessário definir qual o tipo de *smart card* seria utilizado. Baseado no estudo que se encontra no Apêndice B deste trabalho, pôde-se fazer um comparativo entre os *smart cards* mais usados em sistema de identificação eletrônica e micropagamentos. Foram comparados os tipos de tecnologias dos cartões do tipo memória *versus* cartões microprocessados e cartões com interface com e sem contato. Na tabela 6 apresenta-se o comparativo dos cartões memória e cartões microprocessados. Os cartões memória trabalham como dispositivos de armazenamento seguro de informações, em função de possuir criptografia em sua memória. Estes cartões possuem capacidades que variam de 1kb a 4kb. Já os cartões microprocessados podem executar aplicações diretamente em sua memória e implementar soluções de segurança nos níveis de *hardware* e *software*. No entanto, os cartões do tipo microprocessado têm um custo por unidade mais alto se comparado ao cartão memória, o que pode tornar a sua aplicação inviável em alguns casos.

Tabela 6: Comparativo entre Cartões Memória x Cartões com Microprocessador

Cartões memória	**Cartões com microprocessador**
Tipo mais comum.	Tipo mais completo.
Possui só o chip de memória.	Contêm microprocessador.
Proteção acesso à memória.	Inviolável.
Principal vantagem: baixo custo.	Mais caro.
Área mais utilizada: cartões telefone pré-pago.	Áreas mais utilizadas: cartões financeiros, bolsas eletrônicas e controle de acesso.

Na tabela 7 apresenta-se um comparativo entre as interfaces de comunicação com e sem contato, entre leitora e cartão. No comparativo, pode-se observar que o cartão com

interface sem contato só tem a desvantagem de possuir um custo inicial maior, pois, com o tempo, a sua característica de não utilizar contatos metálicos para a cominação reduz a necessidade de substituição de cartões e leitores o que refletirá num menor índice de manutenção.

Tabela 7: Comparativo entre Cartões de Contato x Cartões sem Contato

Cartões de contato	**Cartões sem contato**
Tipo mais comum.·.	Não é inserido no leitor.
Requer inserção do leitor.	Transferência de dados / energia via RF.
Tem oito contatos banhados a ouro.	Vantagens: maior confiabilidade, maior durabilidade.
Desvantagens: pode ficar desgastado ou danificado.	Desvantagens: mais caro, não é apropriado, quando grande quantidade de dados deve ser transferida.

O objetivo deste comparativo é a definição sobre qual dispositivo utilizar no sistema de identificação eletrônica do RU da UFSM. Do estudo, pode-se perceber que o cartão do tipo memória com a tecnologia de comunicação sem contato, apresentou-se mais promissor em função do seu custo beneficio resultante, do seu menor valor de aquisição comparado ao cartão microprocessado e menor índice de manutenções quando comparado ao cartão com contato. Outro fator é que o serviço de transporte público da cidade de Santa Maria utiliza cartões do tipo *Mifare* 1k como sistema de pagamento de passagens. Sendo que o *smart card Mifare* 1k é um *smart card* do tipo memória desenvolvido pela NXP Semicondutores.

A figura 10 ilustra o leitor e um cartão *Mifare* de 1k e a Figura 11 exemplifica um cartão do sistema de transporte público da cidade de Santa Maria – RS (Sistema Integrado Municipal - SIM), o qual é do mesmo tipo de cartão para identificação por *smart cards*.

Com a leitora e os cartões Mifare de 1k, deu-se início ao processo de construção da API para a leitura e gravação dos mesmos. A utilização dos cartões de transporte público permite validar a utilização conjunta do sistema de leitura e escrita desenvolvido neste trabalho, tanto na UFSM como no sistema SIM.

4.3.2. Tecnologias para o Desenvolvimento da API

Existem, no mercado, pacotes de desenvolvimento para a implementação de aplicações com *smart cards*. Alguns destes pacotes foram testados durante o estudo em pauta. Os pacotes testados foram o da SCM *Microsystems* e o da *Advanced Card Systems* (ACS),

ambos os pacotes de desenvolvimento contêm aplicações e APIs proprietárias já desenvolvidas.

Figura 10: Leitora e Cartão Mifare 1K

Figura 11: Cartão SIM

Estas ferramentas proprietárias não permitem a definição de quais setores específicos podem ser utilizados, o que impossibilita o uso simultâneo do mesmo *smart card* para o sistema de identificação eletrônica da UFSM e do transporte público da cidade de Santa Maria. Esta deficiência deve-se ao fato delas adotarem o protocolo proprietário *Near Field Communication* (NFC) que é um padrão definido pelo NFC Fórum, um consórcio global de companhias de *hardware*, *software*, cartões de crédito, bancos, provedores, entre outros, razão pela qual se torna obrigatório o uso de dispositivos compatíveis com este protocolo. Também por se tratar de uma plataforma fechada, alguns recursos, como a definição de diferentes chaves para diferentes setores e a definição de quais blocos podem ser utilizados por cada aplicação, não estão disponíveis para os dois pacotes estudados. Outra desvantagem dos pacotes é a utilização de DLLs proprietárias, o que restringe a customização das aplicações, sendo possível utilizar somente os recursos e as linguagens de programação previamente definidos em cada pacote.

Diante das limitações dos pacotes de desenvolvimento avaliados, e por serem plataformas proprietárias, decidiu-se desenvolver uma aplicação que ofereça uma API exclusiva para o sistema de identificação eletrônica da UFSM. A API visa a atender os seguintes critérios: ser livre de licenças, para redução de custos e manutenção do sistema; ser customizável; e ser compatível com o protocolo PC/SC (*Personal Computer*/*SmartCard*), que é uma arquitetura para a utilização de *smart cards* em computadores pessoais.

Diante disto, optou-se pelo uso da linguagem de programação Java, que se trata de uma linguagem de desenvolvimento que não exige licenças e suas aplicações podem ser executadas em diferentes plataformas de *hardware* e sistemas operacionais. O desenvolvimento da API exigiu a utilização de comandos APDUs *(Applications protocol data units*) que são usados para trocar informações que trafegam entre o cartão inteligente e o terminal. Um exemplo de comando APDU utilizado está ilustrado na Figura 12, que corresponde a um método para a obtenção do identificador único do cartão o UID.

```
public  String UIDSC() {
    String aux = null;
    try {
        byte[] c = {(byte) 0xFF, (byte) 0xCA, (byte) 0x00, (byte) 0x00, (byte) 0x04};
        CommandAPDU readUID = new CommandAPDU(c);
        resp = cardChannel.transmit(readUID);
        aux = (arrayByteToHex(resp.getData()));
    } catch (Exception e) {}
    return aux;
}
```

Figura 12: Método para obtenção do UID.

Este método transmite através do comando CommandAPDU(c) uma sequência de bytes c = {(byte) 0xFF, (byte) 0xCA, (byte) 0x00, (byte) 0x00, (byte) 0x04}; que é o comando APDU para a obtenção do UID. O UID é retornado pelo comando resp = cardChannel.transmit(readUID); que é recuperado pelo comando resp.getData().

4.3.3. Aplicação de Manipulação e Funcionalidades da API Desenvolvida

Para operar os cartões foram desenvolvidas uma API e uma Aplicação Gráfica. A API foi planejada para executar as seguintes funções:

- Ler UID - onde é enviado um comando APDU correspondente e retornado o UID;
- Ler ATR - onde é enviado um comando APDU correspondente e retornada a ATR;
- Logar Chave - onde é enviada a chave em hexadecimal e o comando APDU solicitando a autenticação;
- Ler informações de uma posição da memória específica - onde é enviada a posição que se pretende ler e o comando APDU de leitura, retornando a informação lida;

- Gravar informações numa posição específica de memória - onde é enviada a informação a ser gravada, a posição da memória onde será gravada e o comando APDU para gravação;

A interface gráfica para o envio de comandos para a API e o retorno das informações é ilustrada na Figura 13, onde é exibida a tela construída para a manipulação da API. A interface apresenta quatro botões que, da direita para a esquerda, permitem: obter a ATR do cartão, obter o UID do cartão, ler as informações do cartão e escrever dados no cartão. Na imagem da interface, também é demonstrado o retorno das solicitações feitas ao clicar nos botões.

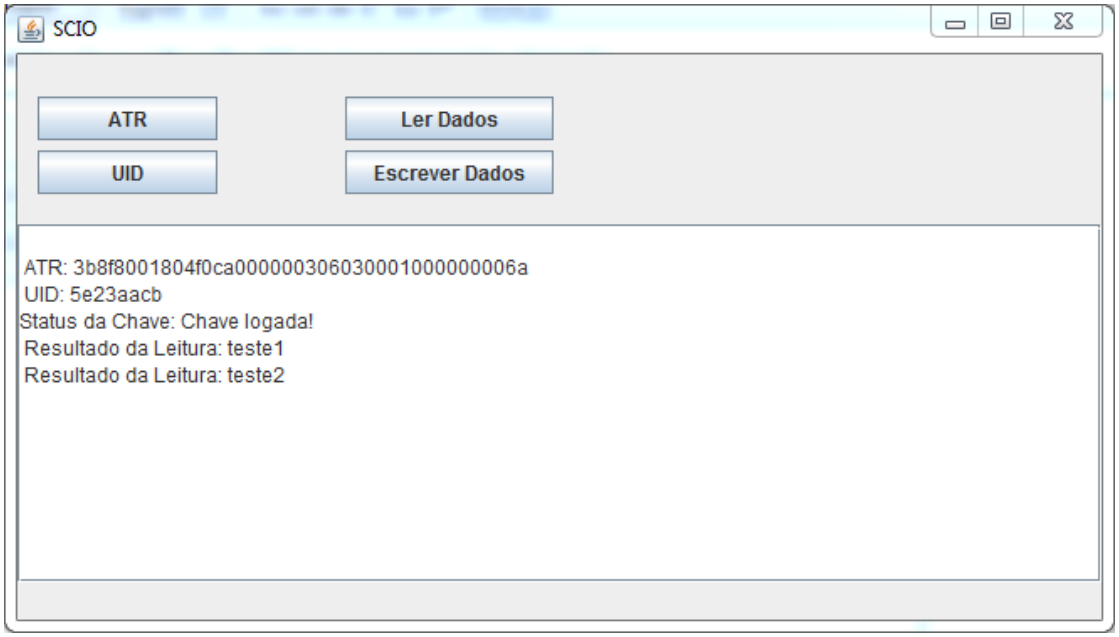

Figura 13: Tela de manipulação da API.

4.4. Testes da API de Leitura e Escrita em *Smart Cards*

Os testes iniciais desta aplicação foram feitos *in loco* no laboratório do grupo de pesquisa GTSeg. Nele, foram feitas diversas leituras e escritas nos cartões com os modelos de leitoras ACR 120, SCL010 e RC700, todas com suporte ao padrão PC/SC. A aplicação também foi testada na central do sistema SIM de transporte público da cidade de Santa Maria. Na central, foram feitas cargas e descargas de créditos do sistema SIM e carga e descarga de informações do sistema de identificação eletrônica desenvolvido neste trabalho. No processo

de teste no sistema SIM, foram utilizados dois cartões do tipo Cartão Cidadão da tecnologia *Mifare* 1K.

Os testes foram executados gravando-se as informações em diferentes setores dos cartões de transporte público e em cartões virgens com capacidades de 1k e 4k adquiridos para a pesquisa. Durante este processo, foram definidos quais setores seriam necessários para alocar as informações do sistema de identificação eletrônica da UFSM e para o correto funcionamento do sistema de transporte urbano da cidade de Santa Maria. A importância deste processo foi definir quais setores serão necessários para armazenar cada aplicação (SIM e UFSM) e as suas posições memória interna. Com isso, garante-se que uma aplicação ou uma ação deliberada acesse e modifique as informações de outras aplicações contidas no mesmo *smart card*, uma vez que os setores ocupados por cada aplicação estarão protegidos por diferentes chaves criptográficas.

De acordo com a quantidade de informação a ser armazenada nos cartões, um setor com quatro blocos de 16 bytes é suficiente para o sistema de identificação eletrônica da UFSM. A aplicação de transporte urbano, por sua vez, ocupa sete setores do *smart card*. Assim, chegou-se a conclusão que, para o modelo de dados atual, um *smart card* do tipo *Mifare* de 1K é suficiente para abrigar as duas aplicações. Caso seja necessário mais espaço para a API, pode-se utilizar cartões do tipo *Mifare* 4k.

4.5. Vulnerabilidades do Sistema de Identificação Eletrônica sem o Modelo de Proteção de Trilhas de Auditoria

O sistema de identificação eletrônica com *smart cards*, apresentado anteriormente neste capítulo, provê uma maior segurança contra clonagens e violações de informações de identificação contidas no cartão de identificação, se comparado ao sistema com código de barras, além disso, possibilita maior flexibilidade na utilização do mesmo dispositivo para diferentes sistemas como: pagamentos, identificação de acesso e vale transporte. No entanto, suas características de segurança só se aplicam no momento da identificação ou do pagamento, o que, certamente, ajuda a evitar fraudes, mas não impede ataques internos de manipulação indevida de informações e créditos.

As informações decorrentes das operações entre os terminais do sistema e os servidores (transações de atualização de cadastro, crédito e débito), assim como as informações armazenadas nas bases de dados (dados de identificação e dados contábeis) não

se beneficiam da segurança do sistema de *smart card*. Portanto, além de proteger os processos de identificação nos terminais é também preciso garantir a segurança das informações de registro de trilhas de auditoria relacionada às operações de manipulação de informações. Uma das formas de se coibir violações, em especial, as violações realizadas por pessoal interno é fazer uso do modelo de proteção de trilhas de auditoria desenvolvido e apresentado no capítulo 3 deste trabalho.

4.6. Levantamento de Requisitos para a Validação do Modelo de Proteção de Trilhas de Auditoria no Sistema de Pagamentos do RU da UFSM

Nesta etapa, fez-se uma série de reuniões com os analistas do Centro de Processamento de Dados e gestores do Restaurante Universitário (RU), estas reuniões tiveram como objetivo fazer a modelagem do problema, previamente apresentada na seção 4.2.2, que. é sintetizado no caso de uso da figura 8, os principais processos realizados por usuários e o sistema do Restaurante Universitário da UFSM (RU). Nesta seção, são detalhados os processos que envolvem o crédito e o débito no sistema de identificação eletrônica do sistema de refeições do RU. O levantamento destas informações tem como objetivo permitir a implementação do Modelo de Proteção de Trilhas de Auditoria no sistema RU. Os detalhes estão representados no diagrama de sequência da figura 14, cujo objetivo é fornecer um maior entendimento dos padrões operacionais e informacionais do sistema de crédito e débito do RU.

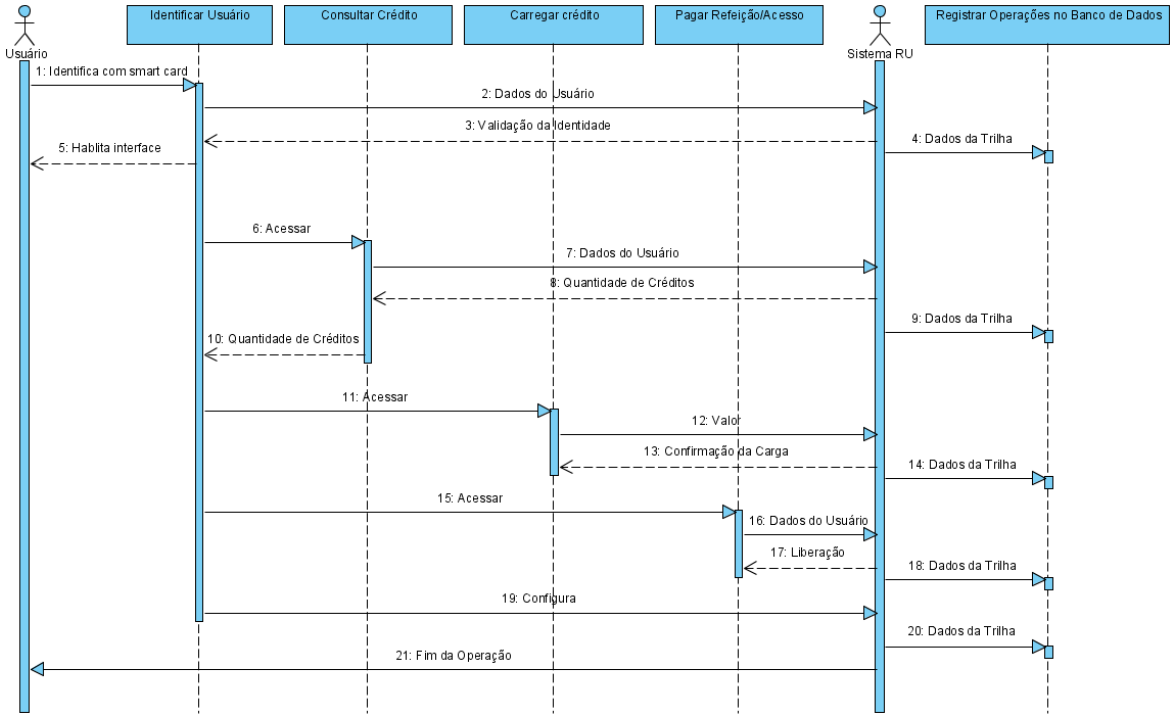

Figura 14: Diagrama de sequência do processo de carga de créditos e pagamento de refeições do sistema RU com *smart card*.

O diagrama representa as principais operações feitas entre os usuários e o sistema. Como pode ser observado, todo o usuário (aluno, servidor ou visitante) pode comprar créditos para fazer refeições, pagar refeições e consultar os seus créditos. O administrador do sistema, que também é um servidor da instituição e usuário, é responsável pela sua configuração, ou seja, é o responsável pelo sistema e pela sua manutenção. Primeiramente, qualquer usuário precisa ter a sua identidade confirmada para ter acesso a qualquer serviço do sistema RU. Após a sua correta identificação, o usuário pode escolher quais operações deseja realizar. O primeiro processo descrito é o da consulta de créditos, que permite o usuário verificar o número de créditos disponíveis para fazer as suas refeições. O segundo processo que se acha à disposição no sistema é o de compra de créditos que possibilita, mediante pagamento, que o usuário valide créditos no sistema RU para fazer refeições. O terceiro processo é o de pagamento de refeições e de liberação de acesso ao Restaurante Universitário, sendo que estas duas atividades ocorrem simultaneamente quando o usuário acessa às instalações do RU. Por fim, o usuário, com perfil de administrador, pode fazer configurações e modificações no sistema RU. A execução de todos os processos é registrada na base de dados.

De acordo com o diagrama, é possível identificar as operações que executam modificações nas informações financeiras ou alterações no sistema, a saber:

- *compra de créditos* – é preciso ter a trilha de quem fez a carga de créditos, a data/hora da operação, qual o valor da operação e qual foi o caixa em que a operação foi feita;
- *fazer refeição/obter acesso* –é necessário ter as informações de quem consumiu os créditos, data/hora da operação, valor descontado e terminal acessado; e
- *gestão do sistema* –é essencial saber quais as alterações foram feitas no sistema, data/hora das alterações e qual a identidade do usuário que fez as alteração.

Com a identificação das operações que provocam alterações nos valores dos créditos ou no sistema, é possível verificar as informações críticas e definir como a trilha de auditoria deve ser construída.

Para o sucesso do processo de auditoria, os dados armazenados nas trilhas devem dar condições de identificar o tempo e o lugar onde as ações ocorreram, bem como quem as fez e o que foi realizado em cada uma. Com base neste princípio e nas informações coletadas durante o estudo do SRU, foram definidas as informações essenciais para a correta identificação de quem fez a operação, quando fez, onde fez e o que fez.

Como resultado, especificou-se a seguinte tupla para registro de proteção de trilha de auditoria:

[Tipo de vínculo], [número de matrícula], [operação realizada], [data/hora], [local] e [valor]

O registro de proteção é gerado a cada ocorrência de compra e/ou pagamento de créditos e nos processos de modificação do sistema. Cada registro é armazenado no *log* do sistema, formando uma trilha de auditoria. No mesmo instante da geração, o registro deve ser enviado pelo sistema de proteção de trilhas para armazenamento independente e auditável.

4.7. Descrição do processo de validação do Modelo de Proteção das Trilhas de Auditoria

Para validar o modelo de proteção de trilhas de auditoria proposto, o modelo foi implementado junto ao Sistema de Informações para o Ensino (SIE) da UFSM, a fim de possibilitar a coleta de informações do sistema de compra e pagamentos do Restaurante

Universitário da instituição. Em função de ser uma etapa de testes, o modelo não foi implantado diretamente no sistema principal, mas sobre uma réplica da base de dados do sistema principal. O trabalho foi realizado com o auxílio dos analistas do CPD, tendo sido implantado um módulo para a coleta e criptografia/descriptografia das trilhas. Para o processo de validação, foi desenvolvida uma interface para a execução de testes. A interface é apresentada na Figura 15, que enfoca o momento em que está sendo feito o processo de criptografia das informações.

As trilhas criptografadas pela aplicação de cifragem do modelo foram armazenadas em uma base independente no servidor da instituição. No período dos testes, observou-se um fluxo médio de 33.281 operações de débito (pagamento de refeições) e 5.885 operações de crédito (compra de créditos) por semana. Durante o processamento destas trilhas, não foram verificados atrasos de processamento ou congelamento do sistema. Conforme demonstrado na seção 3.6, o tempo necessário para o processo de encriptação não é significativo e não provoca atrasos na identificação dos usuários, uma vez que há um consumo maior de tempo na fila do *buffet* (em torno de 5 minutos por pessoa). O tempo para quem vai se servir é maior do que no processo de identificação com o Modelo de Proteção de Trilhas de Auditoria (15 segundos).

Nos testes, a chave simétrica foi armazenada no CPD, ficando sob a responsabilidade do analista encarregado pelo sistema de pagamentos, e a chave assimétrica privada foi gerada em um arquivo binário, ficando a disposição da equipe de auditoria. No caso desta validação o segredo da chave assimétrica privada ficou sobre responsabilidade do diretor do Restaurante Universitário para o seu armazenamento, em função da instituição não ter uma equipe de auditoria para sistemas de informação. Para instituições que possuem equipe de auditoria a chave simétrica deve ficar com um auditor da equipe.

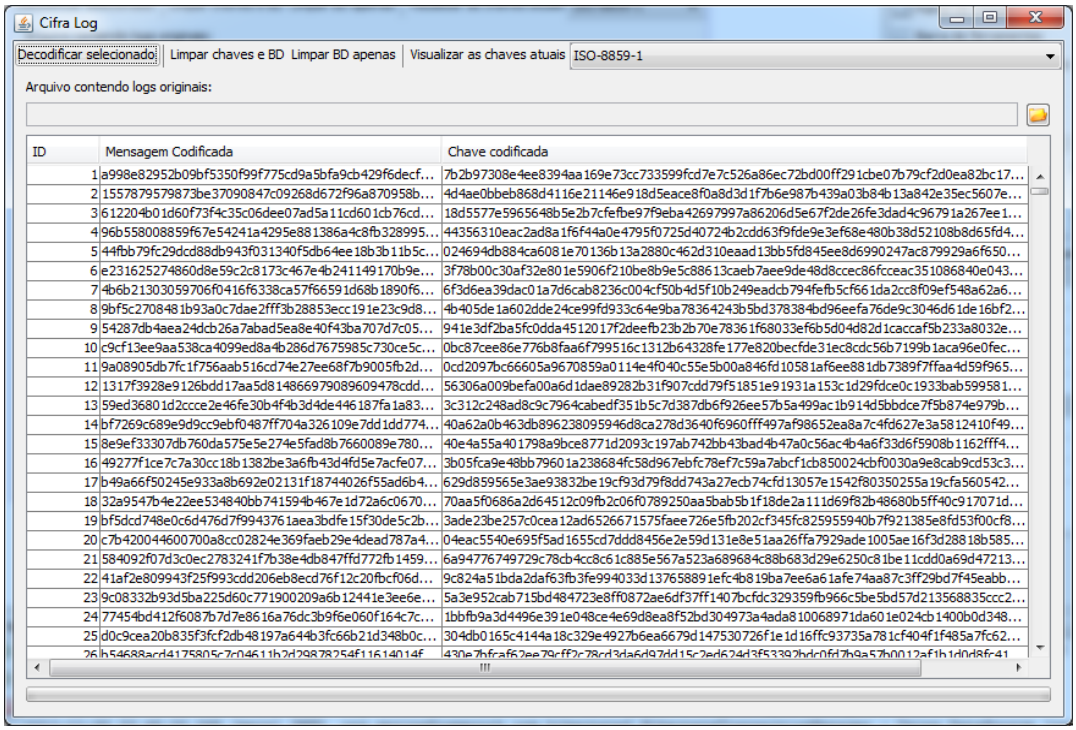

Figura 15: Interface da aplicação de proteção de trilhas.

Como alternativa de armazenamento da chave assimétrica, é possível usar o armazenamento da chave assimétrica privada em um *smart card*, utilizando a API de leitura e escrita em *smart cards*, apresentada na seção 4.3. Esta solução pode diminuir as chances de extravio ou cópia da chave assimétrica privada.

A seguir, são descritas algumas situações de possível violação avaliadas durante o processo de validação.

Durante o processo de auditoria, o sistema de trilhas do sistema SRU pode ser confrontado com o sistema desenvolvido neste trabalho, que permite garantir a autenticidade do registro nativo e eliminar a possibilidade do infrator apagar os seus rastros, já que, para alterar o *log* independente, é preciso acessar outro sistema onde ele está criptografado com um par de chaves. Desse modo, se houver alguma tentativa de violação por parte dos operadores de caixa, as informações referentes à violação (vínculo e matrícula de quem foi beneficiado, local e data de onde a operação foi feita) ficarão a disposição para o processo de auditoria.

Outra situação levantada foi a de que uma pessoa, com permissão de acesso à base de dados e ao sistema, queira modificar algum registro por negligência ou com o objetivo de apagar algum registro de uma operação ilegal. Neste caso, ele poderá ter sucesso em alterar o

sistema de *logs* do sistema principal, neste caso, o SRU. No entanto, ele também terá que obter acesso à base do sistema independente. Se isto ocorrer, a única violação possível é a de apagar toda a base de dados o que, no caso, chamaria a atenção e deixá-lo-ia exposto a investigações. Não haveria a possibilidade de o invasor alterar um dos registros do *log* independente sem obter as duas chaves, lembrando que uma das chaves ficaria junto com a base sob a responsabilidade do administrador e a outra, com o diretor do Restaurante Universitário em função da instituição não ter uma equipe de auditoria. Esta situação também é válida para o diretor do RU que precisa obter acesso ao sistema e dispor das duas chaves para conseguir o acesso às trilhas de auditoria.

Uma terceira situação levantada foi a de que se um analista modificar diretamente as tabelas do banco de dados de forma que a transação não passe pelo gerador de *logs*, haveria a possibilidade de fazer a violação sem deixar registros. Contudo, o modelo implementado no RU contempla o armazenamento das informações de crédito e débito. Sendo assim, mesmo que creditado um valor sem registros do processo, o registro de crédito e débito do sistema de proteção de trilhas apresentaria uma inconsistência entre valor depositado e valor gasto, o que também poderia ser identificado durante a auditoria.

A situação que foi identificada como crítica para o sistema é a de que o analista que implementar a solução pode criar regras no código fonte da aplicação para que algum usuário não tenha determinadas ações registradas no sistema de proteção de trilhas. Este tipo de situação não é facilmente detectada sem que as fontes do sistema sejam auditados. Uma forma de evitar este tipo de situação é que o código fonte também esteja protegido e que exista uma documentação consistente de quem desenvolveu e fez as manutenções na fonte da aplicação.

Auditorias rotineiras também são importantes para a manutenção da integridade da aplicação e do modelo de proteção de trilhas de auditoria.

4.8. Considerações Finais

Neste capítulo, foi apresentada a modelagem do sistema de identificação e pagamentos, assim como a estrutura operacional do Restaurante Universitário da UFSM. Estas informações serviram como base para o desenvolvimento da API de leitura e escrita em *smart cards* que será utilizado pela instituição como mecanismo de identificação eletrônica. A API foi testada em cartões virgens e nos cartões do transporte urbano da cidade de Santa Maria, para a comprovação de sua usabilidade e compatibilidade. O mecanismo de

identificação eletrônica com *smart card* demonstrou-se menos suscetível a fraudes do que os sistemas que adotam, por exemplo, código de barras e funcionou corretamente com o sistema de transporte urbano.

O modelo de trilhas de auditoria foi ajustado conforme as necessidades do sistema do Restaurante Universitário da UFSM e validado com o mesmo. Os testes envolveram a utilização de dados da réplica do banco de dados no período de uma semana. Os testes efetuados demonstraram a aplicabilidade e a regularidade do funcionamento da aplicação usada no modelo e também como o modelo encaixa-se na estrutura operacional da instituição.

Outro ponto a ser observado é que mesmo tendo sido desenvolvido para o sistema de identificação eletrônica da UFSM, o Modelo de trilhas de auditoria e a API de leitura e escrita em *smart cards* pode ser empregado em outros tipos de sistemas de identificação ou que tenham que tratar de informações críticas que precisam de mecanismos de auditoria confiáveis.

5. CONCLUSÃO

A informação é um ativo, com um papel cada vez mais estratégico dentro das instituições. Falhas e desastres em sistemas de informação podem levar a grandes prejuízos e até a falência de uma empresa. A segurança da informação corresponde tanto à proteção dos sistemas de informação contra a negação de serviço a usuários autorizados, bem como a proteção contra a intrusão e a modificação não autorizada de dados ou informações. Sistemas como os de identificação e micropagamentos, como aqueles adotados por instituições de ensino, necessitam de mecanismos de auditoria confiáveis que permitam identificar violações, em especial, as cometidas por colaboradores responsáveis pelo sistema.

Uma das atividades fundamentais para que se identifiquem violações e perdas futuras sejam evitadas é a de auditoria. Mecanismos de trilhas de auditoria, também chamadas de *logs* de auditoria, são formados por registros das atividades de usuários e administradores.

As trilhas de auditoria auxiliam as empresas a manter um controle histórico sobre alterações nas informações, mas não eliminam a vulnerabilidade de administradores de sistemas manipularem estas trilhas para eliminar rastros de modificações maliciosas. Um caso concreto é a manipulação de dados financeiros em instituições do setor educacional, onde o uso de *smart cards* para realização de micropagamentos é uma tendência e a gestão das informações financeiras torna-se vulnerável e pode impactar no orçamento da instituição.

Em virtude das constatações, este trabalho apresentou um modelo de proteção de trilhas (*logs*) que pode ser utilizado como solução para o problema do tratamento e da proteção das trilhas de auditoria. O modelo é baseado em criptografia dos dados e em divisão de responsabilidades na guarda das chaves criptográficas dos registros, possibilitando condições de garantir a legitimidade das informações em sistemas de identificação e pagamento. O modelo desenvolvido foi validado junto ao Centro de Processamento de Dados da Universidade Federal de Santa Maria com o intuito de ser implantado no sistema principal de refeições do Restaurante Universitário da instituição. O sistema possibilita que o Diretor do restaurante, e também gestor financeiro, possa ter garantias sobre a integridade do sistema de trilhas de auditoria sobre pagamentos recebidos e refeições servidas.

Futuramente, o modelo deverá ser integrado ao sistema de identificação e micropagamentos com *smart cards*, cujo API de leitura e escrita também foi modelado e desenvolvido neste trabalho, demonstrando a flexibilidade no uso da solução.

5.1. Trabalhos Futuros

Sugere-se como trabalhos futuros para o modelo de proteção de trilhas de auditoria, o estudo e o desenvolvimento de sistemas autônomos para verificação de trilhas de auditoria que possam representar violações no sistema monitorado. Também sugere-se seu uso em sistemas com informações confidencias ou críticas, tais como nas áreas de saúde e defesa, para o monitoramento de acessos em informações confidenciais e proteção de suas trilhas. Como continuidade no desenvolvimento da API propõem-se o seu uso no gerenciamento de patrimônio com RFID.

REFERÊNCIAS

ALBUQUERQUE, R.; RIBEIRO, B. **Segurança no Desenvolvimento de Software** – Como desenvolver sistemas seguros e avaliar a segurança de aplicações desenvolvidas com base na ISO 15.408. Editora Campus. Rio de Janeiro, 2002.

ALVARENGA, G. L . **Criptografia Clássica e Moderna**. 2ª. Edição Copyright © 2010/2011 Luiz Gonzaga de Alvarenga.

ALVES, M. R.; ZAMBALDE, L. A. **Segurança da Informação**. 1 ed. Lavras. UFLA /FAEPE, 2007.

AMARO, G. **Criptografia Simétrica e Criptografia de Chaves Privadas**: Vantagens e desvantagens. Publicado por Revista de Negócios e Tecnologia da Informação. Vol. 2. Curitiba, 2007.

ANDERSON, R. **Security Engineering**. A Guide to Building Dependable Distributed Systems. Second Edition. Wiley Publishing, Inc. 2008.

ABNT NBR ISO/IEC 17799: **Tecnologia da Informação**. Código de Prática para Gestão da Segurança da Informação. Associação Brasileira de Normas Técnicas. Rio de Janeiro, 2005.

BELLARE, M.; YEE, B. S. **Forward integrity for secure audit logs**. Technical report. Computer Science and Engineering Department, U. California at San Diego, 1997.

_____. **Forward-security in private-key cryptography**. In Proceedings of the RSA Conference Cryptography Track, 2003.

BOSWORTH, S.; KABAY, M.E. **Computer Security Handbook Fourth Edition**. John Wiley & Sons, Inc, 2002 Canada. ISBN 0-471-41258-9. Pg 28 a 846.

BROCARDO, L. M.; DE ROLT, R. C.; FERNANDES, R. **Introdução À Certificação Digital da Criptografia Ao Carimbo de Tempo**. 1 ed. Florianópolis, 2006.

CHEN, Z. **Java Card Technology for Samart Cards**. Architecture and Programmer's Guide. Addison-Wesley. 2000. P3.

DevMedia. **Metadados** – O significado da informação no ambiente de Business Intelligence. Encontrado em: http://www.devmedia.com.br/articles/viewcomp.asp? comp=741 na data de 07/12/2011.

DIAS, C. **Segurança e Auditoria da Tecnologia da Informação**. Axcel Books. Rio de Janeiro, 2000.

ELMASRI, R.; NAVATHE B. S. **Fundamentals of Database Systems 4th ed**. Copyright © 2004 by Pearson Education, Inc. Pg 735.

FABCHER, C.H. **In your pocket: smartcards**. Spectrum, IEEE, Feb 1997.

FERREIRA, N. F. F.; ARAUJO, M. T. **Política de Segurança Da Informação Guia Prático para Elaboração e Implementação**. Editora: Ciencia Moderna, ISBN: 8573935030, Publicação: 2006. Publicação: 2006 Pagina 77 a 83.

FSU. **Florida State University**. Encontrado em: http://www.fsucard.fsu.edu/technology_center.htm na data de 07/07/2011.

FREITAS, M. E. A. **Gestão de Riscos Aplicada a Sistemas de Informação**: Segurança Estratégica da Informação. Biblioteca Digital da Câmara dos Deputados, 2009. Encontrado em: http://bd.camara.gov.br/bd/bitstream/handle/bdcamara /3564/gestao_riscos_freitas.pdf?sequence=4 na data de 04/05/2011.

FLUHRER, S; MANTIN, I.; SHAMIR, A. **Weaknesses in the key scheduling algorithm of rc4**. In RC4, Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography, 2001. páginas 1–24.

GARCIA, F.D.; VAN ROSSUM, P.; VERDULT, R.; SCHREUR, R.W. **Wirelessly Pickpocketing a Mifare Classic Card**. Security and Privacy, 2009 30th IEEE Symposium on 17-20 May 2009.

GEMALTO. **Milestone**: Four Million Students Now Use University Smart Cards Developed by Gemalto and Santander Universities Global. Encontrado em: Division http://www.gemalto.com/php/pr_view.php%3Fid%3D658, na data de 13/05/2011.

HAWTHORN, P. B.; CLIFTON, C.; WAGNER, D.; BELLOVIN, S. M.; WRIGHT, R. N.; ROSENTHAL, A.; POORE, R. S.; CONEY, L.; GELLMAN, R.; HOCHHEISER, H.

Statewide databases of registered voters: a study of accuracy, privacy, usability, security, and reliability issues. Communications of the ACM, 49(4):26–28, 2006.

HENDRY, M. **Multi-application Smart Cards**: Technology and Applications. New York: Cambridge University Press, 2007. 245 p.

IBM. **Recommendation for the Triple Data Encryption Algorithm (TDEA).** Block Cipher, 2004. Encontrado em: <http://csrc.nist.gov/publications/nistpubs/80067/ SP80067. pdf>, na data de: 03 de Abril de 2011

ISO/IEC 11770-1:1996 – **Information technology** – Security techniques – Key management –Part 1: Framework.

KRAUSE, M.; TIPTON, H. F. **Handbook of Information Security Management**. Auerbach, 1999.

LAUREANO, M. A. P.; MORAES, E. P. S. **Segurança como Estratégia de Gestão da Informação**. Revista Economia & Tecnologia – ISSN 1415-451X Vol. 8 – Fascículo 3 P. 38-44 – Ano. 2005.

PETERSON, N. J. Z.; BURNS R.; ATENIESE, G.; BONO S. **Design and Implementation of Verifiable Audit Trails for a Versioning File System**. Proceeding FAST '07 Proceedings of the 5th USENIX conference on File and Storage Technologies ©2007.

PC/SC Workgroup. **PC/SC Technical Workgroup**. Encontrado em http://www.pcscworkgroup.com/ na data de 05/07/2010.

RANKL, W; EFFING, W. **Smart card Handbook**. Third Edition. John Wiley & Sons. 2004 p18.

REZENDE, A. D.; e ABREU, F. A . **Tecnologia da Informação Aplicada a Sistemas de Informação Empresariais**. São Paulo: Atlas, 2000.

RU. **Restaurante Universitário da UFSM**. Encontrado em: http://w3.ufsm.br/ru/index.php na data de 17/3/2011.

SÊMOLA, M. **Gestão da Segurança da Informação** – Uma visão Executiva. Editora Campus. Rio de Janeiro, 2003.

SCHNEIDER, B.; KELSEY, J. **Cryptographic support for secure logs on untrusted machines**. Proceedings of the 7th USENIX Security Symposium, 1998.

SCHNEIDER, B.; KELSEY, J. **Secure audit logs to support computer forensics**. ACM Trans. Inf. Syst. Secur., 2:159–176, 1999.

SIMON, F.; DOS SANTOS, L., A.; HARA S. C. **Um Sistema de Auditoria baseado na Análise de Registros de Log**. Departamento de Informática Universidade Federal do Paraná (UFPR). Escola Regional de Banco de Dados (ERBD'2008), Florianopolis-SC, April 2008.

TAHERDOOST, H.; ZAMANI, M.; NAMAYANDEH, M. **Study of smart card technology and probe user awareness about it**: A case study of Middle Eastern students. Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on 8-11 Aug. 2009.

TANENBAUM, A. S. **Redes de Computadores**. 4 edition, 2003.

TIPTON, H. F; KRAUSE, M. **Information Security Management Handbook**. New York, USA: Auerbach Publications, v.2, 2008.

TRINTA, M. F. N.; MACÊDO, C. R. **Um Estudo sobre Criptografia e Assinatura Digital.** Departamento de Informática Universidade Federal de Pernambuco Setembro, 1998. Encontrado em http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm, em 06/5/2011.

UNICAMP. **Cartão Universitário Inteligente**. 2011. Disponível em: http://www.smartcard.unicamp.br/smartcard/index.html, encontrado em: 05/05/2011.

____.**Cartão Universitário Inteligente Visão Versão <1.0>**.Encontrado em: http://www.unicamp.br/cgi/zope/database/pdf/SmartCardVisao.pdf. Encontrado em: 05/05/2011.

UNISINOS. **Cartão Unisinos**: tudo em um. O Jornal da Unisinos. 02/07/2010. Encontrado em: http://www.juonline.com.br/index.php/universidade/02.07.2010/cartao-unisinos-tudo-em-um/2372, em: 10/08/2010.

UCHÔA, J. Q. – **Segurança em Redes e Criptografia**; UFLA; Lavras – MG; 2003.

WU, Z. Z; B.; WANG, W. **Tamper resistance protection of logs based on forward-secure**. In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, volume 8, pages 90 –94, 2010.

XU, W.; CHADWICK, D. W.; OTENKO, S. **A PKI Based Secure Audit Web Server**. ed. IASTED Communications, Network and Information and CNIS., Phoenix, USA, 2005.

YU, D. C. N.; TAN, C. **Design and Implementation of Mobile Security Access System (MSAS) Based on SSL VPN**. First International Workshop on Education Technology and Computer Science. On page(s): 152 – 155, 2009.

Apêndice A – Artigo Publicado no XI SEPROSUL – Novembro de 2011

Método para a Manutenção Segura de Trilhas de Auditoria

Maintenance Method for Secure Audit Trail

Método para el mantenimiento de Pista de Auditoría Segura

Ernâni Teixeira Liberali, Universidade Federal de Santa Maria (UFSM)
e.liberalli@gmail.com

Giani Petri, Universidade Federal de Santa Maria (UFSM)
gianipetri@gmail.com

Raul Ceretta Nunes, Universidade Federal de Santa Maria (UFSM)
ceretta@inf.ufsm.br

Resumo

O processo de gestão da segurança da informação depende de um processo de auditoria interna. Em sistemas automatizados, a auditoria faz uso das trilhas de auditoria para a identificação de violações no tratamento das informações. As trilhas de auditoria são registros que armazenam informações confidenciais sobre acesso e operações de usuários e administradores. Manter os registros íntegros é de fundamental importância para um processo de auditoria ou ainda para identificar tentativas de fraudes. Este trabalho apresenta um método para manutenção segura de trilhas de auditoria, utilizando técnicas criptográficas juntamente com um processo de distribuição e armazenamento seguro das chaves usadas na encriptação, as quais são mantidas sob responsabilidade de diferentes administradores.

Abstract

The process of managing information security depends on an internal audit process. In automated systems, the audit makes use of audit trails to identify violations in the handling of information. Audit trails are records that contain confidential information about operations and access for users and administrators. Maintain records of integrity is of fundamental importance to an audit process or to identify fraud attempts. This paper presents a method for safe maintenance of audit trails, using cryptographic techniques along with a distribution process and safe storage of keys used for encryption, which are kept under the responsibility of different managers.

Resumen

El proceso de gestión de seguridad de la información depende de un proceso de auditoría interna. En los sistemas automatizados, la auditoría hace uso de las pistas de auditoría para

identificar violaciónes en el manejo de la información. Pistas de auditoría son los registros que contienen información confidencial sobre las operaciones y el acceso de los usuarios y administradores. Mantener registros de la integridad es fundamental para un proceso de auditoría o para identificar los intentos de fraude. Este trabajo presenta un método para el mantenimiento seguro de pistas de auditoría, con técnicas criptográficas, junto con un proceso de distribución y el almacenamiento seguro de claves utilizadas para el cifrado, que se mantienen bajo la responsabilidad de los diferentes administradores.

Palavras-chave: Trilhas de Auditoria; Manutenção segura; Distribuição de chaves.

Keyword: Audit Logs, Maintenance Safe, Key distribution.

Palabras clave: Registros de auditoría, mantenimiento seguro, con distribución de llaves.

1. Introdução

A informação é um ativo que, como qualquer outro ativo importante para os negócios, tem valor para as organizações e, consequentemente, necessita ser adequadamente protegida (NBR 17799, 2003). Falhas e desastres em sistemas de informação podem levar a grandes prejuízos e até a falência de uma empresa. Cartões de crédito, Internet e a globalização potencializaram as relações comerciais entre empresas por meio eletrônico, de forma que a segurança não é apenas uma preocupação da empresa, mas também do cliente ou do consumidor (FREITAS, 2009).

A segurança da informação corresponde tanto à proteção dos sistemas de informação contra a negação de serviço a usuários autorizados, bem como a proteção contra a intrusão e a modificação não autorizada de dados ou informações (armazenados, em processamento ou em trânsito). Sua realização abrange a segurança dos recursos humanos, da documentação, dos materiais, das áreas e instalações, das comunicações e recursos computacionais, assim como ações destinadas a prevenir, detectar, deter e documentar eventuais ameaças a seu desenvolvimento (NBR 17799, 2003; DIAS, 2000; KRAUSE e TIPTON, 1999).

Cuidados especiais devem ser aplicados na gestão da segurança da informação para verificar a integridade e para garantir que dados sensíveis estejam adequadamente protegidos. Uma das atividades fundamentais para que se evitem perdas é a de auditoria. A tarefa de auditoria inclui recomendar ações para eliminar ou minimizar as perdas, através da identificação de vulnerabilidades e riscos, determinar se os controles de segurança adequados estão em vigor, garantir que os dispositivos de auditoria e segurança sejam válidos e verificar

se os controles, as trilhas de auditoria e as medidas de segurança estão funcionando de forma eficaz (BOSWORTH; KABAY, 2002).

Em sistemas automatizados, o processo de auditoria envolve a coleta, a manutenção e a análise das trilhas de auditoria, também chamadas de *logs* de auditoria, que são formadas por registros das atividades de usuários e administradores. Em muitas aplicações, o controle de acesso e informações relacionadas com as operações realizadas devem ser mantidos em arquivos de *log*s seguros (armazenamento seguro das trilhas) para detecção de intrusão e violações ou para fins de auditoria do sistema (XU *et al*. 2005). Nos arquivos de *log*, geralmente, é armazenada uma grande quantidade de informações confidenciais. Portanto, é importante garantir que, caso ocorra alguma violação do sistema, os *logs* não sejam comprometidos e a violação possa ser detectada posteriormente (XU *et al*. 2005). Para não ser descoberto, o primeiro alvo de um atacante experiente costuma ser o sistema de **logs** de auditoria. O seu objetivo é apagar os rastros do ataque para escapar da detecção, bem como para manter o método de ataque em segredo, fazendo com que as falhas de segurança exploradas não sejam identificadas pelos administradores do sistema (BELLARE; YEEY, 1997). Como consequência, há necessidade de proteção aos registros de *logs* utilizando técnicas que permitam garantir a segurança das trilhas de auditoria para protegê-las de adulterações e danos causados por qualquer usuário ou administrador, quer seja de forma intencional ou não.

Este trabalho apresenta um método para a manutenção segura de registros de *logs* de auditoria utilizando algoritmos criptográficos juntamente com um processo seguro de distribuição e armazenamento das chaves utilizadas no processo de encriptação dos *logs.* O método é composto por dois algoritmos, um assimétrico e outro simétrico. O registro de *log* é cifrado com a chave simétrica e, então, ela é cifrada com uma das chaves assimétricas. No entanto, o processo de criptografar uma informação, para uma trilha de auditoria, gera um problema referente à distribuição e ao armazenamento das chaves utilizadas na cifragem. O método proposto distribui as chaves através de um canal de comunicação confiável e armazena-as em bases de dados independentes, sob diferentes responsabilidades: de diferentes administradores, um administrador e um auditor. O resultado é um registro de *log* ilegível e que afiança maior segurança ao processo de auditoria.

O restante do trabalho está organizado como segue. Na seção 2 (dois), apresenta-se a metodologia utilizada na pesquisa. A seção 3 (três), aborda alguns conceitos fundamentais

sobre segurança da informação em um sistema de informações e a seção 4 (quatro) apresenta o conceito de trilhas de auditoria (*logs*). A seção 5 descreve detalhes do método de gestão de trilhas de auditoria proposto., enquanto a seção 6 (seis) faz a descrição dos testes realizados para a avaliação de desempenho do método proposto. A seção 7 (sete), por sua vez, aborda os principais mecanismos para integridade de *logs* existentes na literatura. E, por fim, a seção 8 (oito) apresenta as conclusões do trabalho.

2. Metodologia

A metodologia utilizada, no presente trabalho, inicia-se com a realização de uma pesquisa bibliográfica para a identificação das técnicas e modelos utilizados para a proteção das trilhas de auditoria. A partir do estudo do estado da arte, identificaram-se as características dos métodos adotados para a proteção de trilhas de auditoria. Na sequencia, realiza-se uma pesquisa aplicada, com a implementação de alguns algoritmos criptográficos para verificar a viabilidade de utilizar a criptografia sem comprometer o desempenho do sistema conforme exposto na seção 6.2. Dessa forma, identifica-se o modelo mais adequado para o processo de cifragem do *log* e geração das chaves. Diante disso, propõe-se uma técnica para a distribuição e o armazenamento das chaves, conforme apresentado na seção 5.1. Por fim, realiza-se um comparativo do método deste trabalho com as principais técnicas estudadas de forma qualitativa, para verificar a sua contribuição em relação aos outros métodos.

3. A Segurança da Informação em um Sistema de Informações

É evidente que os negócios estão cada vez mais dependentes das tecnologias e estas precisam proporcionar confidencialidade, integridade e disponibilidade. Segundo Albuquerque (2002) e Krause (1999), há três princípios básicos para garantir a segurança da informação, principalmente no que se refere a sistemas que envolvam dados pessoais e financeiros, tais como:

- Confidencialidade: a informação somente pode ser acessada por pessoas explicitamente autorizadas. É a proteção de sistemas de informação para impedir que pessoas não autorizadas tenham acesso.
- Disponibilidade: a informação deve estar disponível no momento em que a mesma for necessária.
- Integridade: a informação deve ser recuperada em sua forma original (no momento em que foi armazenada). É a proteção dos dados ou informações contra modificações intencionais ou acidentais não autorizadas.

Rezende, Abreu (2000) e Sêmola (2003) defendem ainda que, para que uma informação seja considerada segura, o sistema que o administra deve respeitar os seguintes critérios:

- Autenticidade: garante que a informação ou o usuário é autêntico.
- Não repúdio: não é possível negar (no sentido de dizer que não foi feito) uma operação ou um serviço que modificou ou criou uma informação; não é possível negar o envio ou a recepção de uma informação ou dado.
- Legalidade: garante a legalidade (jurídica) da informação; a aderência de um sistema à legislação; e as características das informações que possuem valor legal dentro de um processo de comunicação, onde todos os ativos estão de acordo com as cláusulas contratuais pactuadas ou a legislação nacional ou internacional vigente.
- Privacidade: foge do aspecto de confidencialidade, pois uma informação pode ser considerada confidencial, mas não privada. Uma informação privada deve poder ser vista / lida / alterada somente pelo seu dono. Garante, ainda, que a informação não será disponibilizada para outras pessoas (neste caso, é atribuído o caráter de confidencialidade à informação). É a capacidade de um usuário realizar ações em um sistema sem que seja identificado.
- Auditoria: rastreabilidade dos diversos passos de um negócio ou processo, identificando os participantes, os locais e os horários de cada etapa. A auditoria aumenta a credibilidade da empresa e é responsável pela adequação da empresa às políticas legais e internas.

No contexto da auditoria de sistemas, é fundamental que haja um planejamento a fim de que se possa contemplar o máximo de requisitos para análise, ou seja, é necessário observar alguns requisitos relevantes para a efetiva realização de uma auditoria bem sucedida. Sendo assim, há vários modelos específicos com técnicas a serem seguidas. Dentre os itens que se pode destacar no processo de planejamento da auditoria, estão:

- conhecimento do ambiente computacional;
- determinação dos pontos de controle;
- estabelecimento dos objetivos de validação e avaliação dos pontos de controle, como (a) técnicas de auditoria, (b) prazos de execução da validação, (c) custos incorridos com a validação, (d) nível de tecnologia exigida do auditor e (d) natureza da fraqueza do controle interno passível de ser alcançada;
- verificação da sensibilidade de cada ponto de controle: matriz ponto de controle, parâmetro, voto, fraqueza, técnica de auditoria a aplicar.
- hierarquização dos pontos de controle e
- documentação e registro do processo de planejamento da auditoria.

4. Trilhas de auditoria (logs)

A auditoria procura identificar e evitar ações suspeitas e fraudulentas por parte do usuário, coletando dados sobre as suas atividades em registros de *log*. As informações coletadas são analisadas a fim de descobrir problemas de segurança e sua origem (SIMON *et al*. 2008). A auditoria digital permite a verificação do conteúdo de um sistema de arquivo em um determinado período no passado. A auditoria é um protocolo de contestação, resposta entre o auditor e o sistema de arquivo a ser auditado (PETERSON *et al,* 2007). A principal funcionalidade de um serviço de auditoria é oferecer armazenamento seguro e permanente dos registros de *log*, para que eles possam detectar quando uma falha de segurança ocorreu (XU et al. 2005).

A necessidade de identificar quais foram às ações e determinar os padrões suspeitos são importantes requisitos para a segurança do sistema. Além disso, a auditoria deve ser realizada de maneira independente e transparente, de forma que todas as informações relevantes devem ser catalogadas (HAWTHORN *et al*. 2006). Uma trilha de auditoria, que também pode ser chamada de *log*, é usada para assegurar o fluxo preciso das transações em um sistema. Cada detalhe de uma fonte, entrada de um determinado documento ou transação deve ser feito com base em um relatório ou arquivo.

A técnica de rastreamento pode ser aplicada em uma única transação para teste rápido; no entanto, para garantir que o controle funcione consistentemente, o teste deve cobrir grandes volumes de dados em diferentes períodos de tempo (BOSWORTH; KABAY, 2002). As trilhas de auditoria devem ser construídas como parte normal dos sistemas de controles internos. Alguns sistemas podem ser adquiridos com o recurso de registro de auditoria automatizados.

O *log* de sistema inclui uma entrada para cada operação aplicada ao banco de dados. Estas entradas, registradas no *log*, podem ser necessárias para a recuperação de uma falha de operação ou falha do sistema (ELMASRI; NAVATHE, 2004). Podem-se expandir as entradas de *log* para que elas também incluam o número da conta do usuário e terminal *on-line* para que seja aplicado a cada operação registrada no *log*. Se qualquer adulteração com o banco de dados é suspeita, uma auditoria do banco de dados é realizada, o que consiste em analisar o *log* para examinar todos os acessos e as operações aplicadas ao banco de dados durante um determinado período de tempo (ELMASRI; NAVATHE, 2004). Quando uma operação ilegal

ou não autorizada é encontrada, o administrador do sistema pode determinar o número da conta usada para executar esta operação. Auditorias de banco de dados são particularmente importantes para bancos de dados sensíveis, que são atualizadas por muitas transações e usuários, como um banco de dados bancário, que é atualizado por vários terminais de atendimento (ELMASRI; NAVATHE, 2004).

Para preparar para uma auditoria futura, um sistema de arquivos gera metadados referentes ao seu conteúdo atual de autenticação e salva-os em uma base de dados ou em um sistema de arquivos. Para realizar uma auditoria, o auditor acessa os metadados, faz contestações ao sistema de arquivos e confronta as informações obtidas com as representadas nos metadados (PETERSON *et al*. 2007).

O sistema deve estar preparado para resistir a ataques com criação de históricos e versões falsas que passam pelo processo de auditoria (PETERSON *et al.* 2007). Esta classe de ataque inclui a criação de versões falsas do arquivo de dados que coincide com os metadados publicados, mas difere dos dados utilizados na sua criação. Também abrange a criação de históricos falsos; a inserção ou a exclusão de versões em uma sequência sem detecção.

Diante disso, destaca-se a importância de um processo de auditoria como uma atividade que objetiva garantir a segurança e visa à continuidade do negócio.

5. Descrição do Método Proposto

Esta seção propõe um método para garantir a segurança na manutenção de registros de *logs*, o qual permite evitar que adulterações sejam realizadas nas trilhas de auditoria. A utilização de técnicas criptográficas aplicadas nos registros auditados, juntamente com um processo seguro para o armazenamento e distribuição das chaves usadas para cifragem e decifragem, é o ponto central da proposta.

O método trabalha com um sistema de criptografia híbrido, unificando técnicas de criptografia assimétrica e simétrica para a proteção dos registros de *logs*. O processo para a segurança das trilhas inicia com a obtenção do registro de *log* recebido da aplicação auditada. O método adota dois tipos de chaves. A simétrica é usada para encriptação e decriptação dos registros de *logs* e a chave assimétrica é utilizada para encriptar a chave simétrica. Durante o processo de cifragem dos *logs*, as chaves são geradas de forma aleatória. A chave assimétrica pública é utilizada para a encriptação da chave simétrica e, então, é descartada. A chave

privada é utilizada para a recuperação da chave simétrica e é armazenada em um local seguro, assim como a chave simétrica.

Com esta abordagem, ao final do processo de encriptação, têm-se dois segredos, em que um é a cifra da chave simétrica e o outro é a chave privada que será necessária para a recuperação da chave simétrica e, consequentemente ,a recuperação das trilhas encriptadas. Estes segredos são distribuídos através de um canal de comunicação confiável e são armazenados em diferentes bases de dados, conforme é descrito na seção 5.1. A seção 5.2, por sua vez, explica como é realizado o processo de cifragem.

5.1. Distribuição e Armazenamento das Chaves

A proteção de *logs* de auditoria é de extrema importância. Assim sendo, utilizar métodos eficientes para evitar que adulterações sejam realizadas é imprescindível. No entanto, a segurança de uma informação confidencial, como os registros de *log*, não depende somente da eficiência dos mecanismos de criptografia utilizados, uma vez que ela também depende da forma com que as chaves utilizadas para a proteção das informações são geridas, isto é, manipuladas, distribuídas e armazenadas. Além disso, a responsabilidade dos administradores dos sistemas em manter os mecanismos de transmissão e armazenamento das chaves, por seu turno, é outro fator de vital importância para manter a integridade dos registros de auditoria. Para maior segurança do sistema, em cada entrada de um registro de *log*, é gerada uma chave nova para a encriptação dos mesmos. Esta característica reduz a possibilidade de um atacante obter uma das chaves criptográficas e conseguir comprometer toda a trilha de auditoria.

A distribuição das chaves criptográficas é realizada através de um canal direto de comunicação baseado no protocolo TCP *(Transmission Control Protocol)* com o uso do protocolo SSL (*Secure Sockets Layer*), para transmiti-las de forma segura. Este modo de transmissão de dados permite garantir a autenticação mútua entre servidor e cliente, a integridade e a confidencialidade da mensagem no meio de comunicação que é utilizado para transportar os segredos criptográficos e as cifras dos *logs* (XU *et al*. 2005). O SSL é um protocolo de segurança criado para prover autenticação e cifragem sobre redes TCP/IP, inclusive, na internet. O SSL foi projetado para executar sobre protocolos de transporte confiáveis. No método proposto, utiliza-se o SSL sobre o protocolo TCP, uma vez que se trata do principal protocolo de transporte da Internet (YU *et al*. 2009). No protocolo SSL, clientes e servidores podem autenticar- se e, em continuidade, trocar dados cifrados entre si.

O SSL possui algumas limitações de segurança devido às suas características e aos propósitos fundamentais. Entre as suas limitações, está o fato de não garantir a proteção dos dados locais (XU *et al*. 2005), haja vista que somente garante a segurança dos dados durante a sua transmissão entre duas aplicações. Para assegurar a proteção dos dados locais, fato que o protocolo SSL não garante, as chaves geradas durante o processo de encriptação são armazenadas em locais seguros, diferentes, sob a responsabilidade de distintos administradores.

Propõe-se que a cifra da chave simétrica fique em uma base de dados que esteja sob a responsabilidade de um administrador, enquanto a chave assimétrica privada fique armazenada em outra base de dados sob a tutela de um auditor externo. As atividades dos usuários são processadas pelo CPD que fica a cargo de um administrador que é o super-usuário do sistema. As transações são interceptadas durante a comunicação com o banco de dados gerando um registro de *log* externo ou utilizado pelo sistema. O registro de *log* capturado é cifrado com a chave simétrica, gerada pela aplicação criptográfica. Após ser utilizada na cifragem do *log*, a chave simétrica é cifrada com a chave assimétrica pública, também gerada pela aplicação criptográfica. Na sequência, o *log* cifrado e a chave simétrica podem ser armazenados no banco de dados. Por fim, a chave assimétrica privada necessária para a recuperação da chave simétrica que, por sua vez, é necessária para a recuperação do *log*, é armazenada em uma base independente sob a responsabilidade de um auditor externo.

Este processo está representado na figura 1 (um). Este método torna necessário o uso de duas chaves para o acesso ao conteúdo do *log* o que permite um intertravamento das chaves para o acesso às informações. O *log* cifrado é gerado de forma independente ao gerado pelo sistema do CPD. O processo de uso das chaves é descrito pela figura 2 (dois) na seção 5.2. O referido processo provê maior segurança para o sistema, visto que, para obter acesso às informações das trilhas, é necessário que as duas bases de dados sejam acessadas para a obtenção das chaves e, posteriormente, executar o processo de decifragem dos registros de *log*. Isto garante que, mesmo as ações feitas por um atacante com o *status* de super-usuário que acesse e altere os registros de *log* originais da base de dados, não será possivel conseguir alterar o segundo registro de *log*, pois, para isso, ele terá que obter a chave local e a chave do auditor externo.

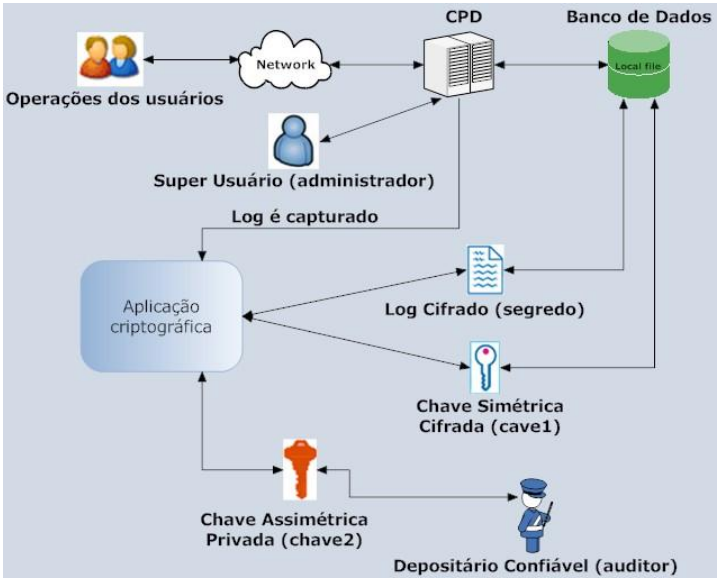

FIGURA 1: Processo de codificação do log e distribuição das chaves.

5.2. Cifragem

O processo de cifragem consiste em transformar um registro de *log* enviado pela aplicação auditada em uma cifra ilegível. A cifragem dos registros de *logs* é realizada com uma chave simétrica gerada em tempo de execução. Em seguida, a chave simétrica é encriptada com uma chave assimétrica pública para, então, serem distribuídas e armazenadas conforme descrito na seção 5.1.

Seja o registro de *log* representado por log, a chave simétrica gerada pelo método representado por SK, o processo de encriptação representado por E, obtém-se o *log* cifrado a partir de,

$$C_{log} = E_{SK}(log).$$

Obtendo a cifra do *log* (C_{log}) com o uso da chave simétrica SK, realiza-se o processo para o armazenamento seguro da chave simétrica gerada anteriormente. Esse processo pode ser representado por

$$C_{SK} = E_{PbK}(SK),$$

onde a chave simétrica SK é cifrada com a chave assimétrica pública E_{PbK} . Dessa forma, obtém-se uma cifra da chave simétrica (C_{SK}). Após esse processo, a chave assimétrica

privada e a cifra da chave simétrica C_{SK}, são armazenadas em base de dados distintas. Este processo de cifragem é representado na figura 2 (dois).

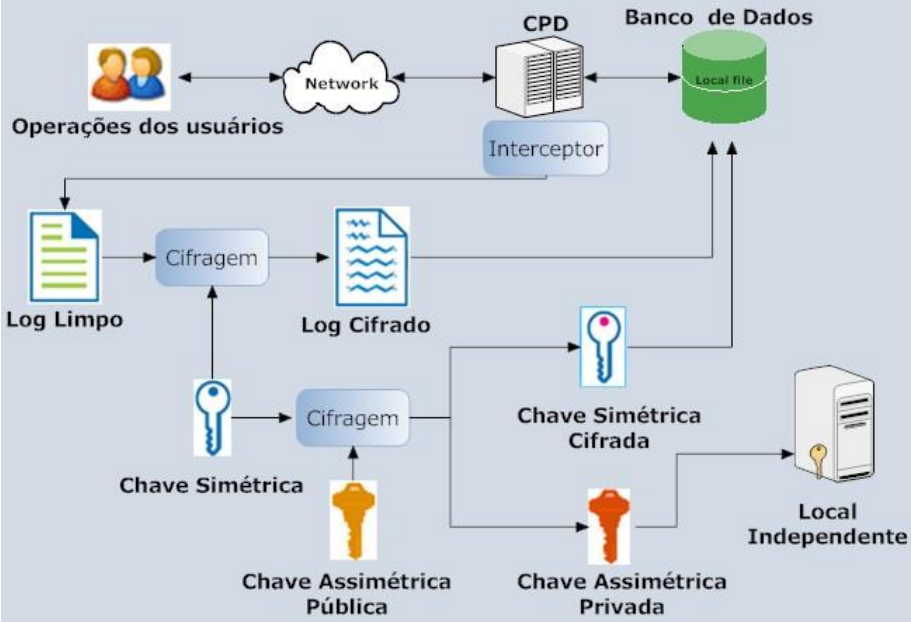

FIGURA 2: Processo de cifragem.

5.3. Decifragem

A decifragem consiste na utilização de chaves criptográficas para descriptografar os registros de *log* cifrados. Para a execução desse processo, inicialmente, é necessário recuperar a chave assimétrica privada da base de dados para, com ela, decifrar a chave simétrica que foi utilizada para criptografar o registro do *log* e que está armazenada em outra base de dados.

Seja a cifra da chave simétrica C_{SK}, a chave assimétrica privada gerada pelo método representada por PvK, o processo de decifragem representado por D. Obtém-se a chave simétrica (SK) a partir de,

$$SK = D_{PvK}\,(C_{SK}).$$

Obtendo a chave simétrica, ela é utilizada para decifrar o *log* encriptado. Esse processo pode ser representado por

$$log = D_{SK}(C_{log}),$$

onde C_{log} corresponde a cifra do registro de *log* e o processo de decifragem com a chave simétrica recuperada representada através de D_{SK}. Desse modo, obtém-se o registro de auditoria original (log).

6. Avaliação de Desempenho

Esta seção apresenta um comparativo entre os tempos de execução de diferentes algoritmos de criptografia (relação na seção 6.1) e uma discussão dos resultados obtidos com a aplicação do método proposto (seção 6.2). O objetivo é comparar o desempenho do processo de encriptação dos algoritmos e verificar se o método proposto possui um desempenho competitivo. Para a implementação dos algoritmos e do método proposto, utilizou-se a linguagem de programação Java, juntamente com o ambiente de desenvolvimento integrado NetBeans.

6.1. Características dos Algoritmos Criptográficos Avaliados

Os algoritmos implementados dividem-se em dois grupos: os algoritmos que utilizam apenas técnicas de criptografia simétrica, sendo eles: AES, RC2, RC4, DES e Triple DES; e os algoritmos híbridos que são formados pelo algoritmo assimétrico RSA, juntamente com os algoritmos simétricos citados anteriormente.

Nas implementações realizadas, o algoritmo AES utiliza um bloco de tamanho fixo em 128 *bits* com uma chave de mesmo tamanho, e o algoritmo DES, uma chave de 56 *bits*. Por sua vez, o algoritmo 3DES utiliza três chaves com tamanho de 64 bits, oferecendo maior segurança, porém é mais lento que o DES original. Os algoritmos RC2 e RC4 também utilizam chaves com tamanho de 128 *bits*.

6.2. Resultados

Assim como os algoritmos estão divididos em dois grupos, os resultados também são apresentados respeitando essa divisão. Os testes realizados consideram os tempos de execução somente do processo de encriptação dos registros de *log*. O equipamento empregado nos testes foi um computador com processador Intel Core I7 com 4GB de memória DDR3 PC 1333MHz e sistema operacional Windows 7 *Ultimate*, utilizando SDK JDK6 sem a execução paralela de outras aplicações.

A realização dos testes deu-se com três tomadas de tempo para cada algoritmo. Cada teste executado possui um fluxo de 1000 registros encriptados, gerando uma média aritmética após as tomadas de tempo.

A Tabela 1 mostra os resultados dos testes utilizando os algoritmos simétricos, juntamente com o algoritmo assimétrico RSA. Estes testes comparam o desempenho com a unificação de diferentes técnicas de criptografia, levando em consideração os aspectos de segurança de cada algoritmo. Os tempos de execução estão representados em segundos.

TABELA 1. Resultados dos testes com algoritmos simétricos junto com o algoritmo RSA.

RSA+AES	**RSA+RC2**	**RSA+RC4**	**RSA+DES**	**RSA+3DES**
59,826s	59,259s	58,978s	59,306s	59,186s

A Tabela 2 expõe os resultados dos testes utilizando os algoritmos simétricos. Os testes somente com esses algoritmos mostram o desempenho das criptografias com as diferentes características de segurança que cada algoritmo possui. Os tempos de execução estão representados em segundos e demonstram que os algoritmos RC2 e RC4 são mais rápidos.

TABELA 2. Resultados dos testes com algoritmos simétricos.

AES	**RC2**	**RC4**	**DES**	**3DES**
0,483s	0,416s	0,379s	0,452s	0,484s

Os tempos de execução demonstrados nas Tabelas 1 e 2 correspondem ao processo de encriptação dos registros de *logs*. Uma comparação simplificada entre os tempos das duas tabelas explicita que a aplicação de um método que utiliza somente um algoritmo simétrico possui um maior desempenho. Porém, o uso de um algoritmo simétrico, usando uma única chave para cifrar e decifrar os dados torna o processo mais vulnerável, posto queao obter acesso à chave simétrica, os registros são facilmente decifrados.

Os tempos resultantes dos testes com algoritmos simétricos mostram que os algoritmos RC2 e RC4 tiveram maior desempenho no processo de cifragem. Porém, Fluhrer *et al*. (2001) afirmam que algumas chaves do algoritmo RC4 são fracas, tornando-o inseguro. O algoritmo DES, em sua forma original, já não é mais seguro (TANENBAUM, 2003). No entanto, a sua forma modificada, o 3DES, mantém um bom nível de segurança, porém é mais lento.

Conforme mostra a Tabela 2 não houve uma expressiva variação de tempo dos algoritmos testados. Com isso, optou-se pela utilização do algoritmo AES no método proposto na seção 2 por suas características de segurança e pela sua ampla utilização prática.

O RSA é considerado um algoritmo robusto, pois sobreviveu a todas as tentativas de rompimento por um longo tempo (TANENBAUM, 2003). A sua utilização juntamente com algoritmos simétricos torna o processo de encriptação mais seguro. Os resultados dos testes da Tabela 1 evidencia que não há uma significativa variação de tempo entre os conjuntos de algoritmos testados, ou seja, a utilização do algoritmo RSA com as características do algoritmo simétrico.

7. Trabalhos Relacionados

A segurança nos registros de *logs* é imprescindível para os modelos organizacionais atuais. Esta seção apresenta alguns trabalhos existentes que buscam garantir a segurança em trilhas de auditoria.

Uma das abordagens para a proteção de registros de *log* é a utilização de um *host* remoto que armazena as informações evitando a violação por *hackers* (WU *et al*. 2010). Nesse método, uma cópia idêntica do *log* é direcionada ao *host* remoto. Outra técnica utilizada é a distribuição das trilhas de auditoria para múltiplas máquinas replicadas na rede, sendo que, para a violação da integridade, os atacantes precisam comprometer mais de uma máquina (WU *et al*. 2010). Contudo, estas abordagens fazem uso de vários recursos de *hardware*, necessitando de um excessivo uso da rede e ainda possuem um alto custo para implementação, não atentando para atacantes internos, tais como administradores maliciosos.

Bellare e Yee (1997 e 2003) definiram integridade em um sistema de *logs* de auditoria propondo uma nova propriedade de segurança denominada *forward integrity*. Esta propriedade é baseada no uso de MAC (*Message Authentication Code*) e não utiliza a replicação em *host* remoto. Prevenindo a alteração ou a inserção de informações pelo

atacante, mesmo quando os registros de *log* ficaram disponíveis, caso ele consiga controle total do sistema. Para garantir proteção, vários registros de *log* são indexados e identificados de forma independente em um intervalo de tempo. No final do intervalo de tempo, outro registro de *log*, definido como entrada especial, contendo o número de registros de *log* no período atual, é criado para indicar o fim desse intervalo de tempo. Tornando a manipulação dos registros de *logs* restrita ao intervalo de tempo corrente, a técnica impede que um atacante, que obtenha a chave, possa adulterar registros gerados em intervalos anteriores. Porém, de posse da chave MAC, o atacante poderá forjar novas entradas de *logs*, sendo-lhe possível comprometer processos de auditoria.

Uma proposta que incrementa o uso de MAC (SCHNEIER e KELSEY, 1998 e 1999) é usar cadeias de *hash* unidirecional para interligar os registros da trilha de auditoria, após cada entrada de *log*. Ao contrário da proposta de Bellare e Yee que indexa e identifica os registros de *log* em um intervalo de tempo, Schneier e Kelsey interligam os *logs* em uma cadeia de *hash* segura, após cada registro de entrada. Cada registro de *log* possui uma ligação, realizada para autenticar todas as entradas prévias. No trabalho de Schneier e Kelsey, após a criação e o encerramento de um registro de *log*, eles são enviados para uma máquina confiável, mantendo, assim, a necessidade de investimento adicional. Schneier e Kelsey desenvolveram também um mecanismo criptográfico para proteger os *logs* de leituras não autorizadas, o qual é baseado em criptografia simétrica, com um servidor central de segurança, que armazena as chaves criptográficas. Dessa forma, caso um atacante invada a máquina com as chaves ou o segredo da chave esteja com apenas uma pessoa que também tenha acesso ao sistema, ele conseguirá acesso aos registros de *logs*, podendo adulterá-los.

Holt (2006) propôs o modelo *Logcrypt*, baseado no modelo de Schneier e Kelsey (1998), adicionando a configuração de chaves públicas. A criptografia de chave pública permite criar assinaturas com uma chave e verificá-las com outra diferente. Estas assinaturas podem ser usadas no lugar de MAC para propiciar a verificação de registro sem a possibilidade de modificá-lo, bem como permitir a publicação da chave inicial usada para criar o *log*, pois só a chave pública é necessária para a verificação. Em seu trabalho, Holt não aborda a forma como é feita a distribuição das chaves utilizadas para cifrar os registros de *log*.

Em (WU *et al*. 2010), um novo esquema de proteger os *logs* com criptografia e assinatura digital foi proposto. O esquema indica o uso de assinatura digital assimétrica baseado em *forward-secure*, mesmo que a chave privada do assinante seja exposta, o atacante

não consegue obter as mensagens cifradas em um tempo anterior, por isso podendo cumprir a criptografia *forward-secure*. Esse modelo tem a capacidade de separar a assinatura da verificação e separar a criptografia da descriptografia.

O método proposto, neste artigo, é composto por um sistema de criptografia híbrido. O uso de algoritmos assimétricos com algoritmos simétricos trata a vulnerabilidade destacada no trabalho de (SCHNEIER e KELSEY, 1998 e 1999), que utilizam um mecanismo baseado em criptografia simétrica. Além disso, o método gera as chaves criptográficas em tempo de execução para cifrar cada registro de *log* e distribui-as através de um canal de comunicação confiável, baseado no protocolo SSL que permite criar um canal de comunicação seguro para a transmissão dos segredos, armazenando-as em bases de dados diferentes e independentes sob a responsabilidade de diferentes autoridades que são o administrador do sistema e um auditor externo, assim sendo, dificultando o acesso a todas as chaves utilizadas no processo de encriptação e prevenindo que novos registros de *logs* sejam forjados, conforme a vulnerabilidade identificada no trabalho de Bellare e Yee (1997 e 2003). O método proposto, neste trabalho, renova as chaves criptográficas a cada registro de *log*. Dessa forma, o processo de criação das novas chaves ficaria bastante custoso ao comparar com a proposta de Wu (2010), o qual trabalha com assinatura digital.

8. Conclusão

A informação tornou-se um dos ativos mais importantes para a manutenção do negócio, diante disso é fundamental que se tenham políticas e mecanismos de gestão da segurança destas informações. Uma das atividades que auxilia na detecção e na prevenção de falhas é a atividade de auditoria, a qual faz uso de trilhas de auditoria que são constituídas de registros de *log*, utilizados por muitas organizações para registrar as atividades e o comportamento das transações em seus sistemas de informações. Com isso, os registros de *logs* tornam-se um mecanismo de vital importância no processo de identificação de fraudes e violações em sistemas que trabalham com informações sigilosas.

Atualmente, existem várias técnicas para garantir a segurança de registros de *log*, cada uma com suas características. O presente trabalho apresenta um método que garante a manutenção segura dos *logs* de auditoria, utilizando técnicas criptográficas juntamente com um processo para a distribuição e o armazenamento seguro das chaves utilizadas.

Com uma análise dos resultados, conclui-se que o uso de dois algoritmos criptográficos, o AES e o RSA, não implica maiores custos computacionais, o que permite trabalhar com duas chaves para a cifragem e decifragem do *log*. O uso de duas chaves provê maior segurança, pois uma das chaves fica sob a responsabilidade do administrador local e a outra sob a tutela de um auditor externo.

Com o uso de algoritmos de criptografia, tem-se uma vulnerabilidade no processo de distribuição das chaves criptográficas utilizadas, sendo que, na transmissão das chaves, um atacante pode interceptar o meio de comunicação e obter as chaves para decifrar os registros de *log*. No método proposto neste trabalho, a distribuição das chaves criptográficas é realizada através de um canal de comunicação formado pelo protocolo SSL dessa forma, aumentando a segurança no processo de distribuição das chaves. Por fim, conclui-se que este método provê uma maior segurança global para as trilhas de auditoria, pois, além de implementar algoritmos criptográficos comprovadamente eficientes pela literatura, ele trabalha com a divisão de responsabilidades para o acesso às trilha de auditoria. Neste caso, mesmo que o atacante tenha o perfil de administrador do sistema, ele não conseguirá alterar os registros gerados pelo modelo proposto neste trabalho. Outro ponto relevante é que, durante uma auditoria no sistema de *log* nativo, ele poderá ter os seus registros confrontados com os gerados pela aplicação apresentada neste trabalho.

Referências Bibliográficas

ALBUQUERQUE, Ricardo e RIBEIRO, Bruno. *Segurança no Desenvolvimento de Software – Como desenvolver sistemas seguros e avaliar a segurança de aplicações desenvolvidas com base na ISO 15.408.* Editora Campus. Rio de Janeiro, 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR ISO/IEC 17799*: Tecnologia da Informação. Código de Prática para Gestão da Segurança da Informação. Associação Brasileira de Normas Técnicas. Rio de Janeiro, 2003.

BELLARE, M. and YEE, B. S. *Forward integrity for secure audit logs*. Technical report. Computer Science and Engineering Department, U. California at San Diego, 1997.

BELLARE, M. and YEE, B. *Forward-security in private-key cryptography*. In Proceedings of the RSA Conference Cryptography Track, 2003.

BOSWORTH, S.; KABAY, M.E. *Computer Security Handbook Fourth Edition.* John Wiley & Sons, Inc, 2002 Canada. ISBN 0-471-41258-9. Pg 28 a 846.

DIAS, Cláudia. *Segurança e Auditoria da Tecnologia da Informação*. Axcel Books. Rio de Janeiro, 2000.

ELMASRI, Ramez; NAVATHE B. Shamkant. *Fundamentals of Database Systems 4th ed.* Copyright © 2004 by Pearson Education, Inc. Pg 735.

FREITAS, M., Eduardo Antônio. *Gestão de Riscos Aplicada a Sistemas de Informação: Segurança Estratégica da Informação.* Biblioteca Digital da Câmara dos Deputados, 2009. Encontrado em: http://bd.camara.gov.br/bd/bitstream/handle/bdcamara_/3564/gestao_riscos_freitas.pdf?sequence=4 na data de 04/05/2011.

FLUHRER, S; MANTIN, I., and SHAMIR, A. *Weaknesses in the key scheduling algorithm of rc4*. In RC4, Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography, 2001. páginas 1–24.

HAWTHORN, P, B; CLIFTON, C; WAGNER, D; BELLOVIN, S. M; WRIGHT, R. N; ROSENTHAL, A; POORE, R. S; CONEY, L; GELLMAN, R; e HOCHHEISER, H. *Statewide databases of registered voters: a study of accuracy, privacy, usability, security, and reliability issues*. Communications of the ACM, 49(4):26–28, 2006.

KRAUSE, Micki e TIPTON, Harold F. *Handbook of Information Security Management.* Auerbach, 1999.

PETERSON, N. J. Zachary; BURNS Randal; ATENIESE, Giuseppe; BONO Stephen. Design and Implementation of Verifiable Audit Trails for a Versioning File System. Proceeding FAST '07 Proceedings of the 5th USENIX conference on File and Storage Technologies ©2007.

REZENDE, Denis Alcides e ABREU, Aline França. Tecnologia da Informação Aplicada a Sistemas de Informação Empresariais. São Paulo: Atlas, 2000.

SÊMOLA, Marcos. Gestão da Segurança da Informação – Uma visão Executiva. Editora Campus. Rio de Janeiro, 2003.

SCHNEIDER, B. and KELSEY, J. Cryptographic support for secure logs on untrusted machines. Proceedings of the 7th USENIX Security Symposium, 1998.

SCHNEIDER, B. and KELSEY, J. *Secure audit logs to support computer forensics. ACM Trans. Inf. Syst. Secur*., 2:159–176, 1999.

SIMON, Fernando; DOS SANTOS, L., Aldri; HARA S. Carmem. *Um Sistema de Auditoria baseado na Análise de Registros de Log*. Departamento de Informática Universidade Federal do Paraná (UFPR). Escola Regional de Banco de Dados (ERBD'2008), Florianopolis-SC, April 2008.

TANENBAUM, A. S. *Redes de Computadores*. 4 edition, 2003.

TIPTON, Harold, F; KRAUSE, Micki. *Information Security Management Handbook.* New York, USA: Auerbach Publications, v.2, 2008.

WU, Z., ZhuGe, B., and WANG, W. *Tamper resistance protection of logs based on forward-secure.* In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, volume 8, pages 90 –94, 2010.

XU, Wensheng; CHADWICK, David W., and OTENKO, Sassa. *A PKI Based Secure Audit Web Server*. ed. IASTED Communications, Network and Information and CNIS. , Phoenix, USA, 2005.

YU, Dingguo., Chen, Nan and TAN, Chengxiang. *Design and Implementation of Mobile Security Access System (MSAS) Based on SSL VPN.* First International Workshop on Education Technology and Computer Science. On page(s): 152 – 155, 2009.

Apêndice B –Apresentação da Tecnologia dos *Smart Cards*

Este apêndice apresenta os elementos estudados no processo de escolha da tecnologia a ser utilizada no sistema de identificação eletrônica. Os itens estudados envolvem a normatização técnica sobre a tecnologia dos *smart cards*, o estudo sobre os principais modelos de dados usados na operação dos *smart cards*, as interfaces de comunicação e as suas características técnicas e os protocolos de comunicação.

1. Normas Técnicas

Com vistas a dar maior interoperabilidade e segurança aos *smart cards*, a ISO/IEC define padrões específicos para construir e operar um *smart card*. Esta padronização é importante, pois os *smart cards* são utilizados em diversos serviços em que a identificação do usuário e o sigilo das informações são fatores críticos.

1.1. ISO/IEC 7816

A norma ISO/IEC 7816 define as características dos cartões (*smart cards*) para utilização com dispositivos externos. Essas características visam a manter a interoperabilidade entre cartões de diferentes fabricantes, padronizando propriedades físicas e elétricas, características de comunicação, identificação de *chip* e dados contidos nele. Ela é composta por 15 partes que são:

- parte 1: Cartões com contatos - características físicas;
- parte 2: Cartões com contatos - dimensões e localização dos contatos;
- parte 3: Cartões com contatos - protocolos de interface e de transmissão elétrica;
- parte 4: Organização de segurança e comandos básicos para a troca de informações;
- parte 5: Cadastro de fornecedores de aplicativos;
- parte 6: Define o meio de transferência de dados;
- parte 7: Definição dos comandos *Structured Card Query Language* (SCQL);
- parte 8: Comandos para operação segura;
- parte 9: Comandos para gerenciamento de cartões;
- parte 10: Define os sinais eletrônicos e ATR (*answer to reset*) para cartões;
- parte 11: Verificação pessoal através de métodos biométricos;

- parte 15: criptografia de informações.

1.2. ISO / IEC 14443

A ISO / IEC 14443 é um padrão para *smart cards* que usa a interface sem contato, opera na faixa de 13,56 MHz em um raio de 10 centímetros do leitor antena. A norma é composta por quatro partes:

Parte 1: [ISO / IEC 14443-1:2000] define o tamanho e as características físicas do cartão. Ela também enumera as condições ambientais que o cartão deve ser capaz de suportar, sem danos permanentes à sua funcionalidade.

Parte 2: [ISO / IEC 14443-2:2001] estabelece a potência da radio frequência (RF) e a relação sinal ruído. Dois sistemas de sinalização, do tipo A e tipo B, também são definidos. A comunicação é no regime *half-duplex* com 106 *kbits* por segundo de taxa de dados em cada direção. Os dados transmitidos pela placa são modulados com carga de 847,5 kHz. Também define a alimentação do cartão que é alimentado por um campo de RF e não é necessário bateria.

Parte 3: [ISO / IEC 14443-3:2001] determina os protocolos de inicialização e anticolisão do Tipo A e Tipo B. Os comandos anticolisão, as respostas, o *frame* de dados e o calendário estão definidos na parte 3 (três). O esquema de inicialização e anticolisão foi concebido para permitir a construção de leitores multi-protocolo que possibilita a comunicação com os dois tipos de cartões.

Parte 4: [ISO / IEC 14443-4:2001] define os dados de alto nível dos protocolos de transmissão do tipo A e tipo B. Os protocolos descritos na Parte 4 (quatro) são elementos opcionais do padrão ISO / IEC 14443.

Os tipos A e B diferem na forma de comunicação dos circuitos.

2. Arquiteturas Operacionais

Existem normas e padronizações que definem a estrutura de dados da memória dos *smart cards* e dos recursos dos dispositivos. As arquiteturas são descritas a seguir.

2.1. EMV (*Europay*, Mastercard e Visa)

Entre os padrões de comunicação, destaca-se o padrão EMV [EMVCo, 2010], criado a partir de 1993 pela colaboração dos principais sistemas de pagamento mundiais (Europay, Mastercard e Visa). O padrão EMV delimita uma série de regras e padrões no que diz respeito às modalidades de operação dos cartões de crédito com *chip*, às suas características físicas e elétricas, à estrutura dos cartões de um ponto de vista da segurança, à interoperabilidade dos cartões nos terminais em nível global, dentre outras.

O padrão EMV define, também, as regras de interação entre os cartões e os terminais de pagamento, as quais estão baseadas no padrão ISO 7816. O padrão EMV determina os requisitos mínimos de segurança, mas deixa os circuitos livres para estabelecer parâmetros adicionais de segurança, o que levou ao desenvolvimento de diferentes sistemas. A Visa desenvolveu o VSDC (*Visa Smart Debit Card*), a Mastercard, o M/Chip e a JCB, o J/Chip. Todos esses sistemas são compatíveis com o padrão EMV, mas têm parâmetros adicionais de gestão do risco nas transações (EMVCo, 2010). Este modelo tem a vantagem de ser um dos mais seguros, mas é um sistema proprietário caro para o uso em empresas sem vínculo com instituições financeiras e operadoras de cartões.

2.2. PC/SC - Personal Computer/Smart Card

A especificação PC/SC (*Personal Computer/SmartCard*) propõe uma arquitetura para a utilização de *smart cards* em computadores pessoais, tendo sido desenvolvida pelo *PC/SC Workgroup*, fundado em 1996 por empresas líderes do mercado de computadores pessoais e de *smart cards*: Bull CP8, Gemplus, Hewlett-Packard, IBM, Microsoft, Schlumberger, Siemens Nixdorf, Sun Microsystems, Toshiba e Verifone. Define uma interface de baixo nível de dispositivos e API de aplicativos independentes permitindo que vários aplicativos compartilhem recursos do cartão. É baseada nas normas ISO/IEC 7816 e EMV e foi desenvolvida para facilitar o uso de cartões inteligentes em ambientes computacionais (PC/SC WORKGROUP, 2010). Este modelo apresenta a vantagem de sua compatibilidade com diversos dispositivos de diferentes fabricantes e tem suporte nativo nos principais sistemas operacionais.

2.3. NFC

O *Near Field Communication* (NFC) é um padrão definido pelo NFC Fórum, um consórcio global de companhias de *hardware*, *software*, cartões de crédito, bancos, provedores, entre outros. O NFC é uma tecnologia que permite conectividade sem fio de curto alcance. Produtos com tecnologia NFC simplificam a maneira como os dispositivos interagem entre si, acelerando a velocidade de conexão, recebimento e compartilhamento de informações, o que o torna uma forma rápida e segura de se fazer pagamentos (NFC BRASIL, 2011). Este modelo tem a desvantagem de ser proprietário.

3. Interfaces de Comunicação

Para o usuário do *smart card*, as partes visíveis de um módulo de *chip* são os contatos dourados. Estas áreas de contato representam a interface de comunicação do *smart card* com o mundo externo. A norma descreve a área de contato mínimo e a sua posição sobre o cartão inteligente e funções dos contatos. Podem existir interfaces de oito e seis contatos, dependendo do tamanho do *chip* usado (HAGHIRI; TARANTINO, 2002). *Chips* com um tamanho de até cerca de 5 mm² podem ser construídos em um módulo com seis áreas de contato. Para c*hips* com dimensões maiores, são usados módulos oito contatos. O tamanho máximo de chips é aproximadamente 32 mm². A seguir, na Figura 1 (um), um exemplo de dois *chips Infineon* onde, à esquerda, um módulo de *chip* com oito áreas de contato e a direita de um módulo com seis áreas de contato.

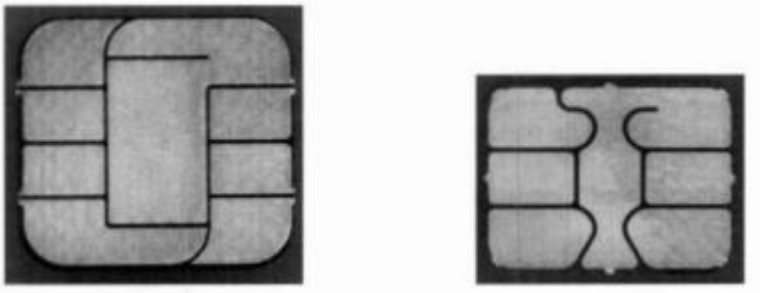

Figura 1: Exemplos de contatos (HAGHIRI; TARANTINO, 2002).

A norma ISO 7816 define a funcionalidade dos oito contatos dos *smart cards*, para garantir que os pontos de contato estejam na mesma posição em cada cartão inteligente conforme figura 2 (HAGHIRI; TARANTINO, 2002).

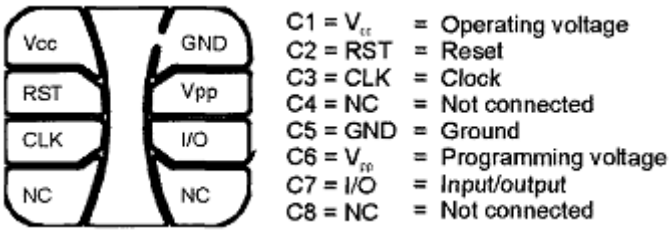

Figura 2: Funcionalidades dos contatos (HAGHIRI; TARANTINO, 2002).

Vcc: tensão que provê a alimentação do chip do cartão com valores aproximados entre 3 (três) e 5 (cinco) V.

RST: este ponto de contato tem como objetivo promover um *reset* a quente, ou seja, corre o envio de um sinal de *reset* ao *chip*. Já um *reset* a frio ocorre quando se desliga a fonte de alimentação, como quando se retira e insere um cartão na leitora.

CLK: caso os cartões inteligentes não tenham gerador de *clock* interno no *chip*, há a necessidade de um sinal de *clock* externo a partir do qual o *clock* interno é derivado.

GND: é o ponto de contato que serve de tensão de referência, normalmente esta tensão é de 0 (zero) V.

Vpp: é um ponto opcional e utilizado em cartões inteligentes antigos, era usado para programar a EEPROM, é mantido para compatibilidade.

I/O: é um ponto *half-duplex* para entrada ou saída de dados do cartão.

NC: Contatos disponíveis: reservados para uso futuro.

Nos chips atuais, a tensão de programação Vpp é produzida internamente no *chip*, portanto, o contato C6, atualmente, não é ocupado. Os módulos de *chips* com interface sem contato necessitam de dois contatos elétricos necessários para a ligação elétrica da antena com o módulo do *chip* (HAGHIRI; TARANTINO, 2002). A alimentação do circuito é feita por indução eletromagnética gerada pelo leitor ao aproximar o cartão. As áreas de contato, no entanto, não são determinadas pelo módulo do *chip*, portanto, não existe um padrão para o tamanho e a posição, conforme figura 3 (três). Para melhorar a comunicação alcance em torno de 10 cm, pode-se acoplar uma antena no corpo do cartão, figura 13.

Figura 3: Interface do chip tecnologia sem contato (HAGHIRI; TARANTINO, 2002).

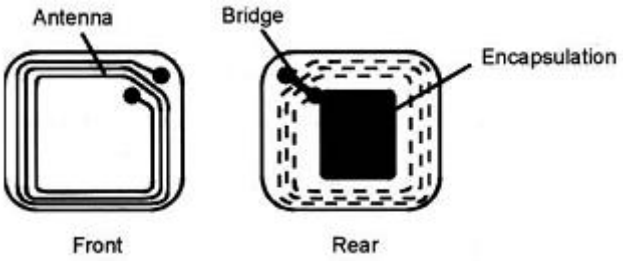

Figura 4: Esquema da antena (HAGHIRI; TARANTINO, 2002).

4. Leitores

Leitores de cartão inteligente são circuitos ou placas que permitem a comunicação com o computador. De forma muito ampla, existem duas famílias de leitores de *smart cards* as com e as sem contato. No entanto, características adicionais podem ser agregadas para dar mais segurança aos leitores como teclado para digitação de senha inclusos no próprio leitor e sistemas de validação entre leitor e cartão antes da transmissão de dados iniciar. Estes recursos ajudam a proteger informações críticas de segurança, especialmente PINs e senhas.

5. Base tecnológica para a Construção da API

Nesta seção, são apresentados os protocolos de comunicação e os elementos necessários, como as *strings* de comunicação ATR e UID, ao desenvolvimento da API, sendo que isso é necessário para a comunicação entre o computador e o cartão (*smart card*) do sistema de identificação eletrônica.

5.1. Protocolos de comunicação

Os protocolos de transmissão definem os processos de comunicação entre o terminal e o cartão inteligente. A compreensão destes protocolos foi necessária para o desenvolvimento

da API de comunicação descrita neste capítulo. A comunicação é feita no princípio como mestre-escravo, onde o terminal atua como mestre e o cartão como escravo, em que a comunicação é *half-duplex* (RANKL; EFFING, 2003). A seguir, na tabela 3 (três), mostra-se um exemplo de protocolos de comunicação:

Tabela 3 - Protocolos de transmissão (RANKL; EFFING, 2003).

Protocolo	Significado
T = 0	Assíncrona, half-duplex, orientado a byte, especificado na norma ISO / IEC 7816-3
T = 1	Assíncrona, half-duplex, orientado a bloco, especificados na norma ISO / IEC 7816-3
T = 2	Assíncrona, full-duplex, orientado a bloco, especificados na norma ISO / IEC 10536-4
T = 3	Full-duplex, ainda não especificado
T = 4	Assíncrona, half-duplex, orientado a byte, extensão de T = 0, ainda não especificado
T =5 ... T = 13	Reservado para uso futuro, ainda não especificado
T = 14	Para uso nacional ainda não padronizados pela ISSO
T = 15	Reservado para uso futuro e ainda não especificado

Apenas dois protocolos de transmissão são utilizados, atualmente, por normas e padrões internacionais (RANKL; EFFING, 2003). O T=0 foi o primeiro a ser usado e parte do princípio de envio *byte* a *byte*, tem a vantagem de consumir menos memória. É muito utilizado em cartões para comunicação móvel, sendo definido pela norma ISO / IEC 7816-3. Já o protocolo T=1 que envia blocos de *bytes* também é definido pela norma ISO / IEC 7816-3. Por ser orientada a bloco, toda vez que um bloco é enviado do terminal para o cartão a resposta dele é devolvida pelo cartão, mantendo uma alternância de envio e retorno de informação. É um protocolo mais robusto e possui mecanismos de detecção e reenvio de blocos que contem erros. É utilizado em sistemas de cartões de pagamento e identificação pelo fato deles exigirem um tráfego maior de informações (RANKL, 2007). Existe ainda o protocolo T=2, que está em fase de definição, baseado no T=1 e no protocolo USB que possui uma taxa de transmissão maior que o T=0 e T=1 podendo chegar a 1,5 Mbps (RANKL, 2007).

5.2. Protocolo APDU

Applications protocol data units (APDUs) são usados para trocar informações que trafegam entre o cartão inteligente e o terminal. O APDU é uma unidade de dados padronizada internacionalmente para a camada de aplicação, que é a camada sete no modelo

OSI. Nos cartões inteligentes, esta camada localiza-se acima da camada de protocolo de transmissão.

As unidades de dados dependentes do protocolo da camada de protocolo de transmissão são chamadas de “protocolo de transmissão de unidades dados” (TPDUs). O protocolo APDU pode ser dividido em dois formatos: C-APDU, que é o pacote de comando, e R-APDU, que é o pacote de resposta. Para cada APDU de comando, existe um APDU de resposta. Em termos simples, uma APDU é uma espécie de recipiente que contém um comando completo para o cartão ou uma resposta completa a partir do cartão. APDUs são transmitidos pelo protocolo de transmissão de forma transparente, o que significa sem alteração ou interpretação (RANKL; EFFING, 2003).

As APDUs seguem a norma ISO / IEC 7816-4 e devem ser independentes do protocolo de transmissão. Por conseguinte, o conteúdo e o formato de um APDU não devem mudar quando um protocolo de transmissão diferente é usado, o que se aplica, sobretudo, aos dois protocolos padrão, T = 0 e T = 1. Essa exigência de independência de protocolo afeta a estrutura do APDUs, já que deve ser possível transmiti-las de forma transparente, usando tanto o *byte-oriented* T = 0 protocolo e do bloco *T-oriented* = 1 protocolo

5.2.1. Comando APDU

O comando APDU é composto por um corpo e um cabeçalho (ISO 7816-4:2008). Na figura 14, é apresentada a estrutura de comando APDU.

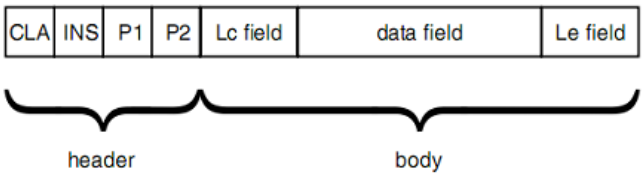

Figura 5: Estrutura de um comando APDU (ISO 7816-4:2008).

O *byte* CLA é o *byte* de classe e é usado para identificar a aplicação. É obrigatório o preenchimento desse campo, porém não precisa representar nenhum valor. O *byte* INS é o *byte* de instrução que identifica o comando envidado para o cartão. Esse *byte* sempre deve ter valor e é tratado como um código pré-definido que identifica uma solicitação de comando realizada pela aplicação. É obrigatório o preenchimento desse campo. Os *bytes* P1 e P2 são *bytes* de parâmetros e podem ser utilizados para prover informações sobre o comando

enviado. O preenchimento desse campo é obrigatório. O *byte* LC Field identifica o comprimento do corpo opcional de dados. O *Data Field* ,campo de preenchimento, é opcional. O *Data Field* e o campo de dados que será enviado para o cartão com o propósito de executar a operação solicitada no cabeçalho do comando. É um campo opcional. O *byte LE Field* identifica o tamanho de resposta esperado pelo *hos*t em relação ao comando enviado. É opcional o preenchimento desse campo.

5.2.2. Estrutura de uma resposta APDU

A resposta de APDU é enviada em resposta a uma APDU de comando e consiste em um corpo opcional e um “reboque” obrigatório, conforme mostrado na Figura 6 (seis).

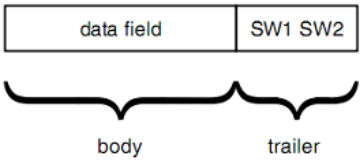

Figura 6: Estrutura de uma resposta APDU (ISO 7816-4:2008).

Data Field é o campo de dados que contém informações sobre a resposta, o seu tamanho é defino pelo LE da estrutura de comando. É um campo opcional. Os *bytes* SW1 e SW2 são *bytes* que identificam o código de retorno, os quais são predefinidos por algumas especificações e também podem ser pré-definidos pela aplicação. Por exemplo, o código 90 00 identifica que o comando foi completado com sucesso. É obrigatório o preenchimento desses campos.

O cartão deve sempre enviar um reboque em resposta a um comando. Os dois *bytes* SW1 e SW2, que também são chamados de "código de retorno", codificam a resposta ao comando. Por exemplo, o código de retorno "90 00" significa que o comando foi executado completamente e com sucesso. Há mais de 50 códigos diferentes (ISO 7816-4: 2008). O seu esquema de classificação básica é ilustrada na figura 6 (seis).

SW1 SW2	Significado
9000	**Processamento normal**
61XX	
62XX	**Processamento com avisos (*warnings*)**
63XX	
64XX	**Erros de execução**
65XX	
66XX	
6700	**Erros de verificação**
68XX	
69XX	
6AXX	
6B00	
6CXX	
6D00	
6E00	
6F00	

Figura 6: Esquema básico de códigos de retorno (ISO 7816-4:2008).

6. ATR

O *Answer to Reset* (ATR) é uma *string* de caracteres que é enviado pelo cartão. Esses caracteres contêm informações sobre o protocolo de transmissão e os dados do cartão, sendo enviados depois que o cartão foi alimentado com tensão elétrica no momento do contato como terminal. Caso o comando não seja enviado corretamente, o terminal rejeitará o cartão. A estrutura do comando ATR é definida pela norma ISO/IEC 7816-3 e possui cerca de 33 bytes de tamanho. É muito raro que um ATR tenha o comprimento máximo permitido (RANKL; EFFING, 2003), sendo que, na maioria das vezes, consiste em apenas alguns *bytes*. Em aplicações em que o cartão deve ser usado muito rapidamente, após a sequência de ativação, o ATR deve ser curto.

Um exemplo típico é o pagamento de um pedágio rodoviário com uma bolsa de cartão eletrônico inteligente. Mesmo que o veículo passe pelo pedágio rapidamente, deve ser possível identificar de forma confiável o cartão de débito, no pouco tempo disponível. O início da transmissão ATR deve ocorrer entre 400 e 40.000 ciclos de *clock* após o terminal emite o sinal de *reset*. Com um *clock* de 3,5712 MHz, o que corresponde a um intervalo de 112 ms a 11,20 ms, ao 4,9152 MHz, o intervalo é de 81,38 ms para 8,14 ms. Se o terminal não recebe o início da ATR dentro deste intervalo, ele repete o tempo de ativação (normalmente, até três vezes) para tentar detectar o ATR (RANKL; EFFING, 2003). Se todas

estas tentativas falharem, o terminal assume que o cartão está com defeito e reage em conformidade.

7. UID

Todos os cartões inteligentes compatíveis com a ISO 7816 são fornecidos com um número Identificador Único (UID), semelhante a um número de *chassis* de um veículo. Para fins de interoperabilidade, o UID de um cartão está aberto e disponível para ser lido por todos os leitores compatíveis.

REFERÊNCIAS

EMVCo, Europay, Mastercard e Visa, **"About EMV"**. Disponível em: http://www.emvco.com/about emv.aspx , visitada em 26/08/2010.

HAGHIRI, Y.; TARANTINO, T. **Smart card manufacturing**: a practical guide. J. Wiley, 2002.

ISO/IEC 7816. Características dos cartões (*smart cards*) para utilização com dispositivos externos.

ISO / IEC 14443. Padrão para *smart cards* com interface sem contato.

NFC BRASIL. **NFC Forum** encontrado em: http://www.nfcbrasil.com.br/?p=49 na data de 17/05/2011.

PC/SC Workgroup. **PC/SC Technical Workgroup**. Encontrado em http://www.pcscworkgroup.com/ na data de 05/07/2010.

RANKL, W; EFFING, W. **Smart card Handbook**. Third Edition. John Wiley & Sons. 2004 p18.