UNIVERSIDADE CATÓLICA DE SANTOS

PROGRAMA DE MESTRADO EM INFORMÁTICA

TRANSFERÊNCIA DE PROPRIEDADE DE ETIQUETAS RFID E UTILIZAÇÃO DE MATRIZES MDS 16 X 16 NA CIFRA DE BLOCO AES

Elcio Abrahão

Dissertação apresentada à Reitoria de Pós-Graduação Stricto Sensu da Universidade Católica de Santos, como parte dos requisitos para obtenção do título de Mestre em Ciências no Programa de Mestrado em Informática, Área de Sistemas Distribuídos

Prof. Dr. Jorge Nakahara Jr

Orientador

Unisantos

Santos, SP - Brasil

2007

Dados Internacionais de Catalogação-na-Publicação (CIP)

Divisão Biblioteca Unisantos

Abrahão, Elcio

Transferência de Propriedade de Etiquetas RFID e Utilização de Matrizes MDS 16 x 16 na cifra de bloco AES / Elcio Abrahão.

Santos, 2007.

105f.

Dissertação de Mestrado – Curso de Mestrado em Informática. Área de Sistemas Distribuídos – Universidade Católica de Santos, 2007. Orientador: Prof. Dr. Jorge Nakahara Jr. .

1. RFID. 2. MDS. 3. AES. I. Campus Vila Mathias. Universidade Católica de Santos. Divisão de Ciência da Computação. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

ABRAHÃO, Elcio. **Transferência de Propriedade de Etiquetas RFID e Utilização de Matrizes MDS 16 x 16 na cifra de bloco AES**. 2007. 105f. Dissertação de Mestrado – Universidade Católica de Santos, Santos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Elcio Abrahão

TÍTULO DO TRABALHO: Transferência de Propriedade de Etiquetas RFID e Utilização de Matrizes MDS 16 x 16 na cifra de bloco AES.

TIPO DO TRABALHO/ANO: Dissertação / 2007

Elcio Abrahão

Av.Itaborai, 1321

CEP 04.135-001 – São Paulo–SP

TRANSFERÊNCIA DE PROPRIEDADE DE ETIQUETAS RFID E UTILIZAÇÃO DE MATRIZES MDS 16 X 16 NA CIFRA DE BLOCO AES

Elcio Abrahão

Composição da Banca Examinadora:

Prof. Dr.	Getulio K. Akabane	Membro Externo	-	UNISANTOS
Prof. Dr.	Jorge Nakahara Jr	Orientador	-	UNISANTOS
Prof. Dr.	Hermes Senger	Membro Titular	-	UNISANTOS
Prof. Dr.	Auri Marcelo Rizzo Vincenzi	Suplente	-	UNISANTOS

UNISANTOS

A Valéria por todo suporte e incentivo durante esta jornada, que não é a primeira e com certeza não será a última... Ao meu orientador Jorge Nakahara Jr, cujo rigor, determinação e tenacidade são exemplo de profissional e amigo.

Agradecimentos

Primeiramente, gostaria de agradecer aos professores da Unisantos pela dedicação e atenção dispensada nestes últimos anos,

Aos meus pais e minha irmã pela insistência em continuar achando que todo trabalho que faço é o mais bonito...

A Grass Roots Bittime America pelo incentivo e suporte durante o último ano,

Ao Prof. Ms. Celso Poderoso,
coordenador da graduação da FIAP pelo incentivo e suporte durante o curso.

Aos meus alunos da graduação da FIAP, cuja sede de saber foi incentivo para este trabalho,

E finalmente para Deus, pela saúde, sabedoria e força para vencer os desafios e dificuldades que acompanham todas as grandes conquistas.

"You must keep sending work out; you must never let a manuscript do nothing but eat its head off in a drawer. You send that work out again and again, while youŕe working on another one. If you have talent, you will receive some measure of success - but only if you persist."

— Isaac Asimov

Resumo

Neste trabalho foram desenvolvidas duas frentes de pesquisa. Na primeira foram estudados protocolos de comunicação para etiquetas identificadas por rádio freqüência (RFID). As também chamadas etiquetas inteligentes são capazes de armazenar informações e fazer algumas operações computacionais elementares. Estas informações podem ser lidas através de um aparelho leitor sem necessidade de contato físico ou visada, como nas etiquetas de código de barras. A comunicação é feita remotamente por rádio freqüência e esta característica permite que a etiqueta seja suscetível a uma série de ataques que tentam fraudar ou obter informações do usuário sem sua autorização. Um ponto importante na utilização dessas etiquetas é a transferência de propriedade de um usuário para outro. A fim de garantir a segurança e privacidade do usuário nesse momento foi desenvolvida uma nova funcionalidade em um protocolo já existente de comunicação entre etiqueta e leitor a fim de permitir que a etiqueta fosse transferida de dono sem riscos à segurança e à privacidade. O protocolo modificado foi chamado ROTEP. Na segunda frente de pesquisa foi proposta uma nova e mais rápida camada de

difusão para a cifra de bloco AES. Essa nova camada substitui as camadas de rotação de bytes e transformação linear sobre colunas por uma nova matriz MDS e involutória. O objetivo é prover difusão completa em uma única iteração, melhorando significativamente a segurança da cifra. Os elementos da nova matriz MDS possuem baixo peso Hamming para proporcionar boa performance tanto na operação de ciframento quanto na de deciframento. Foi utilizada uma matriz de Cauchy ao invés de uma construção circular como na matriz original do AES, pois uma matriz circular não pode ser ao mesmo tempo MDS e involutória. O algoritmo modificado foi chamado MDS-AES.

Abstract

This work has two parts. In chapter 2 we study an improvement on an RFID protocol to allow ownership transfer and in chapter 3 we propose a new MDS involutory matrix to the AES block cipher. First part: RFID (Radio Frequency IDentification) is a technology that will become pervasive in the near future. Smart Labels endowed with microchips will substitute optical bar codes with many advantages. It is important that this technology can be utilized without bringing security and privacy problems to the users across all of the supply chain. One of the most important aspects of RFID security is the communication protocol between the reader and the smart tag. An ideal secure protocol must identify each tag without allowing any type of attack against the tag owner.

One recently proposed secure communication protocol (Dimitriou, SecureComm 2005) is based on mutual authentication where a secret identification number is shared between the reader database and the tag. This protocol is shown to be secure against all major attacks and to demand little RAM memory and computational operations. Nonetheless, it does not allow the reutilization of the tag,

since it does not offer a secure mechanism to allow the ownership transfer of the tag. The first contribution of this work is an improvement of Dimitriou's protocol allowing the ownership transfer of the tags across the supply chain without exposing the system to the most common attacks. The improved protocol was called RFID Ownership Transfer Enable Protocol (ROTEP).

Second part: This part proposes a new, large diffusion layer for the AES block cipher. This new layer replaces the ShiftRows and MixColumns operations by a new involutory matrix in every round. The objective is to provide complete diffusion in a single round, thus sharply improving the overall cipher security. Moreover, the new matrix elements have low Hamming-weight in order to provide equally good performance for both the encryption and decryption operations. We use the Cauchy matrix construction instead of circulant matrices such as in the AES. The reason is that circulant matrices cannot be simultaneously MDS and involutory. The new cipher was called MDS-AES.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AES	Advanced Encryption Standard
AK_i	AddRoudKey
EAS	Electronic Article Surveillance
EPC	Electronic Product Code
CP	Chosen Plaintext
DC	Differential Cryptanalysis
DES	Data Encryption Standard
GMC	Gerador de Matrizes de Cauchy
HF	High Frequency
IBM	International Business Machines
IFF	Identify Friend or Foe
KP	Known Plaintext
LC	Linear Cryptanalysis
MAC	Message Authentication Code
MDS	Maximum Distance Separable
MC	MixColumns

MITM	Meet-in-the-middle
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
NSA	National Security Agency
RAM	Random Access Memory
RFID	Radio Frequency IDentification
ROM	Read-Only Memory
ROTEP	RFID Ownership Transfer Enable Protocol
SB	SubBytes
SKU	Stock Keeping Unit
SPN	Substitution Permutation Network
SR	ShiftRows
TC	Trusted Center
TM	Transfer Mode
UHF	Ultra High Frequency
WORM	Write Once - Read Many

Lista de Símbolos

T	Uma etiqueta RFID
R	Leitor de Etiquetas
ID_i	Identificação de uma etiqueta RFID na i-ésima comunicação
M_1, M_2	Concatenação das mensagens M_1 e M_2
$h(M)$	Aplicação de uma função de hash h em M
$h_k(M)$	Aplicação de uma função MAC h_k em M
N	Número pseudo-aleatório único

1 Introdução

1.1 Objetivo

O objetivo principal deste projeto de mestrado é a seguranąa da informação. Proteger informações de valor para empresas e indivíduos tem sido alvo de inúmeras pesquisas no meio computacional devido a crescente demanda do tráfego, principalmente em redes como a Internet. Apesar de não se relacionarem diretamente, os dois temas deste trabalho de mestrado tem grande importância no contexto onde estão inseridos e aplicações práticas imediatas. Os objetivos de cada tema são:

a) desenvolver uma variante de protocolo de comunicação para etiquetas identificadas por rádio freqüência (RFID) que permita a transferência de propriedade da etiqueta de um usuário para outro sem comprometer a segurança e privacidade do usuário.

b) modificar a cifra de bloco AES introduzindo uma matriz de Cauchy, MDS e involutória a fim de melhorar a segurança do algoritmo, garantindo difusão com-

pleta em uma única iteração.

O item (a) será discutido no Seção 1.2 e o item (b) na Seção 1.3.

1.2 O Protocolo RFID: motivação

Identificação por rádio freqüência é uma tecnologia emergente que movimentará um mercado de US$ 2.77 bilhões até o final de 2006 e com estimativa de US$ 26.23 bilhões em 2016, conforme [45]. Essa tecnologia consiste na utilização de rádio freqüência para comunicação remota e tipicamente é usada para identificar produtos (mercadorias) durante toda a cadeia de suprimentos (desde a fabricação do produto, passando pelos atacadistas, varejistas até o consumidor final). A tecnologia RFID é composta por três elementos principais:

- Etiqueta: pequeno chip dotado de antena com capacidade limitada de memória e processamento.
- Leitor: aparelho capaz de emitir ondas de rádio freqüência e ler a resposta das etiquetas, repassando a informação obtida para um banco de dados.
- Banco de Dados: software gerenciador de banco de dados responsável por manter as informações sobre cada produto relacionado a uma etiqueta e vice-versa.

As etiquetas inteligentes, como também são chamadas as etiquetas RFID, irão substituir com vantagens o código de barras em um futuro próximo. Para isso basta que o custo por etiqueta passiva (aquela que não possui fonte de energia própria e é ativada pelo campo eletromagnético do leitor) atinja o patamar de US$ 0,05 por unidade segundo a EPC Global [21], o que viabilizaria a utilização em massa. Uma das grandes empresas multinacionais pioneiras na utilização de RFID é o Wal-Mart [52], cujo projeto rastreia mercadorias nos galpões de estoque e distribuição. A maioria dos projetos em andamento se limitam a utillizar uma etiqueta para cada palete de carga ou caixa com múltiplas unidades de produto em estoque Stock Keeping Unit (SKU). Nesse volume, a utilização de etiquetas RFID já é vantajosa economicamente em relação ao código de barras convencional [21]. Existem várias combinações de hardware e software nos elementos desta tecnologia. As etiquetas são classificadas quanto a sua fonte de energia (própria ou externa), quanto a freqüência de rádio utilizada, quanto à distância entre leitor e etiqueta para comunicação e quanto ao ambiente em que é utilizada (mais detalhes no capítulo 2). A ausência da necessidade de visada [1] para leitura das etiquetas, uma de suas principais vantagens, traz consigo algumas preocupações em relação à segurança do sistema e privacidade do usuário. Se considerarmos uma configuração simplificada do conjunto de elementos (banco de dados - leitor - etiqueta) (Fig. 1.1), utilizando um protocolo de comunicação simples, qualquer

[1] O leitor de etiquetas de códigos de barra emite um feixe de raios laser que refletindo na etiqueta permite a identificação do código pelo sensor do leitor. Portanto é necessário que o espaço entre o laser a etiqueta esteja livre para que haja a leitura.

entidade não autorizada pode atacar o sistema de diversas maneiras a fim de obter vantagens, fraudar o proprietário dos produtos ou mesmo obter informações sobre o produto que possam revelar algum segredo de seu usuário. Protocolos inseguros também propiciam a espionagem industrial e a coleta de informações de valor comercial por entidades que podem vender esta informação a potenciais concorrentes. Exemplos típicos são etiquetas em remédios que possam identificar a doença de seu proprietário ou tentativas de se desativar etiquetas ou trocar sua identificação em produtos de alto valor agregado a fim de substituí-los por preços menores ou modelos diferentes. Outra característica inerente à essa tecnologia é que durante todo o ciclo de vida do produto este tipicamente mudará de proprietário várias vezes. Se todo o fabricante do produto utilizasse etiquetas RFID somente em seu estabelecimento e não permitisse sua reutilização por outros proprietários, o custo do produto seria maior pois cada novo proprietário teria que anexar sua própria etiqueta. A primeira contribuição deste trabalho é a criação de uma nova funcionalidade para um protocolo de comunicação entre leitor e etiquetas proposto por Dimitriou [20] a fim de permitir a reutilização de uma única etiqueta durante todo o ciclo de vida de um produto, levando-se em consideração a segurança do protocolo e a privacidade do usuário durante toda a sua vida útil. O objetivo não é implementar o protocolo, limitando-se ao estudo teórico de seu funcionamento, embora, nenhuma solução abordada impeça uma implementação prática. O novo protocolo foi denominado ROTEP (RFID Ownership Transfer Enable Protocol).

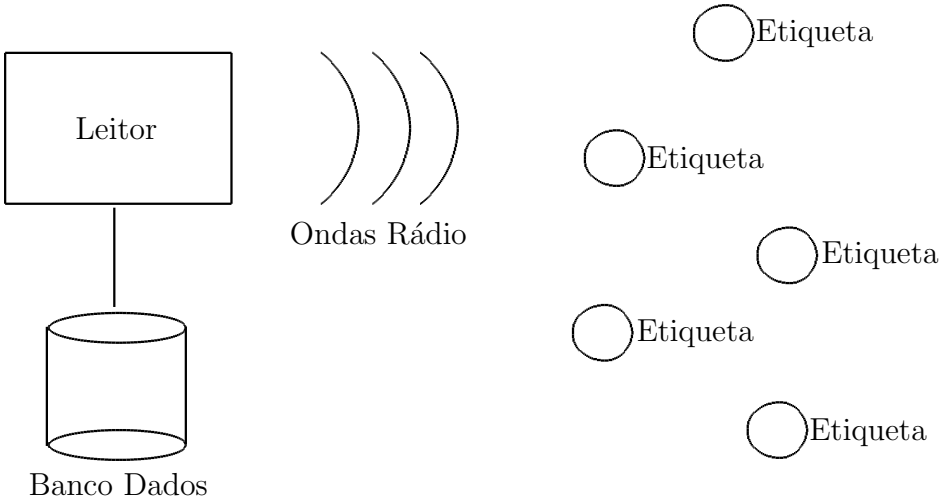

FIGURA 1.1 – Esquema básico de RFID. Fonte: [4]

1.3 Matriz MDS na cifra de bloco AES: motivação

Criptologia é a ciência que engloba a criptografia e a criptoanálise Criptografia é o estudo da escrita secreta e criptoanálise é o estudo das metodologias para decifrar a escrita secreta.

Segundo David Kahn [30] desde os tempos do imperador romano Júlio César, os governantes percebem as vantagens fornecidas pela criptologia. Com a criptografia conseguem manter protegidas suas informações sensíveis e com a criptoanálise buscam as informações de seus adversários. Durante a Segunda Guerra Mundial, novamente a criptografia e a criptoanálise desempenharam papel decisivo no de-

senrolar dos confrontos. Surpreendidos inicialmente pelo ataque a Pearl Harbor, os americanos não foram capazes de prever a audaciosa operação conduzida pelos japoneses, apesar de terem interceptado e decifrado mensagens diplomáticas daquele governo indicando um possível ataque. Contudo, a situação foi revertida pelos americanos ao longo do conflito. Além de quebrarem as cifras japonesas Red e Purple, conseguiram manter sua principal cifra, a Sigaba, intacta. Mas é do velho continente que vem o maior exemplo de sucesso nesta área. Além de utilizarem sua cifra Typex com êxito, os ingleses protagonizaram uma das maiores operações de quebra de cifra de que se tem notícia. Herdando informações iniciais dos poloneses e contando com a genialidade de pessoas como Alan Turing, foram capazes de decifrar a máquina alemã Enigma. A guerra se transformava em um jogo de cartas marcadas, com os ingleses podendo prever grande parte das jogadas dos alemães. Uma aliança entre americanos e ingleses, para a troca de informações oriundas de suas operações de criptoanálise permitiu, segundo historiadores, que a guerra fosse abreviada em dois a três anos.

No final da Segunda Guerra Mundial também surgiram os conceitos de confusão e difusão. Segundo Shannon [47] a propriedade de confusão significa que a relação entre os símbolos do criptograma e os símbolos da mensagem e da chave é complexa. O intuito é eliminar dependências simples (lineares) entre o criptograma, a mensagem e a chave, dificultando a criptoanálise. Este é um critério qualitativo. A propriedade de confusão é tipicamente implementada através de substituições (não-lineares). Já a propriedade de difusão dita que as propriedades estatísticas

da mensagem são dissipadas no criptograma. Isso significa, por exemplo, que redundâncias (do idioma) da mensagem não são preservadas no criptograma. Logo, cada símbolo do criptograma depende de cada símbolo da mensagem e da chave, e mesmo pequenas alterações na mensagem ou na chave alteram aleatoriamente cada símbolo do criptograma. Esse é um critério quantitativo. A difusão é tipicamente implementada através de transposições. Os americanos criarm em 1952 a sua Agência de Segurança Nacional (NSA, sigla em inglês) [49], a fim de centralizar e liderar todos os esforços governamentais em criptologia. A agência se transformaria no maior empregador de matemáticos e lingüistas, o centro com a maior capacidade computacional do planeta, com a maior quantidade de supercomputadores em um mesmo lugar.

A partir de 1970, o extraordinário avanço das telecomunicações e da informática abria novos horizontes para diversos setores da economia. Redes de computadores eram criadas para atender aos setores bancário e financeiro, demandando por uma solução que atendesse aos problemas de segurança da informação que surgiam com as novas tecnologias. A criptografia emerge como solução para o problema. Estudos e pesquisas nesta área proliferaram-se pelos grandes centros universitários. É sob este clima que os Estados Unidos resolvem adotar um algoritmo criptográfico para ser utilizado como padrão comercial, um sistema que pudesse prover um alto nível de segurança às emergentes necessidades de sistemas de telecomunicações e informática. Devido ao aumento do tráfego de informações digitais o governo americano previu a necessidade de um algoritmo criptográfico

para proteger esse grande volume de informações. O DES (Data Encryption Standard) foi escolhido através de um concurso inciado pelo NBS (National Bureau of Standards). O DES foi projetodo por pesquisadores da IBM e passou a vigorar como padrão de criptografia simétrica para aplicações comerciais e governamentais. A partir de 1997 o NBS, agora como NIST (National Institute of Standards and Technology) iniciou o processo para seleção do novo AES (Advanced Encryption Standard) com 15 algoritmos candidatos e o selecionado foi o Rijndael [2]. A nova cifra de bloco está em vigor desde setembro de 2000 e é o atual algoritmo de chave secreta padrão para proteção de dados nos EUA.

Preocupações com a segurança nacional e por fim o advento da Internet, do comércio eletrônico e do e-mail só vieram a aumentar a preocupação com o estudo de algoritmos ainda mais seguros. A contribuição deste trabalho é propor uma melhoria na segurança do algoritmo AES tornando-a ainda mais resistente aos ataques conhecidos, utilizando-se uma matriz MDS de dimensão 16 x 16. Embora não apresente uma implementação do algoritmo, este trabalho fornecerá todas as informações necessárias para se elaborar implementações em software ou hardware. Esse novo algoritmo foi chamado MDS-AES.

1.4 Organização do trabalho

1.4.1 Transferência de Propriedade de Etiquetas RFID

O capítulo 2 está organizado da seguinte forma: A seção 2.1 contém uma explanação inicial sobre a tecnologia RFID, a seção 2.2 contém definições sobre a terminologia de RFID, na seção 2.3 explicaremos sobre a padronização das etiquetas, na seção 2.4 uma introdução sobre segurança e privacidade, nas seções 2.5 a 2.7 serão discutidos tipos de ataques a RFID, as seções 2.8 a 2.9 explicarão o funcionamento de protocolos de comunicação para RFID, a seção 2.10 explica sobre o problema de transferência de propriedade, a seção 2.11 apresenta a contribuição de nosso trabalho, as seções 2.13 e 2.14 fazem a análise de segurança do protocolo original e do protocolo sugerido por nós e por fim a conclusão na seção 2.15.

1.4.2 Nova matriz MDS involutória para cifra de bloco AES

O capítulo 3 está organizado da seguinte forma: A seção 3.1 contém uma introdução sobre o AES, na seção 3.2 são apresentados os conceitos matemáticos preliminares necessários ao desenvolvimento das demais seções. A seção 3.3 apresenta a nova matriz MDS involutória para o AES, substituindo as camadas SR

e MC. A seção 3.4 apresenta uma análise de segurança da MDS-AES e a seção 3.5 a conclusão para este capítulo.

2 Transferência de Propriedade de Etiquetas RFID

2.1 Introdução

Identificação por rádio freqüência cuja sigla em inglês é RFID (Radio Frequency IDentification), é uma tecnologia já presente em nosso dia a dia, trazendo uma série de benefícios como: aumento de produtividade, diminuição de custos de logística e transporte, segurança em aeroportos e outras inúmeras aplicações práticas. Segundo Landt [32], a história do RFID remonta ao tempo da criação do universo a 14 bilhões de anos pois nessa época tudo se resumia a radiação eletromagnética que é justamente a fonte de energia do RFID. Entre os anos de 1600 a 1800 cientistas vieram a entender melhor a eletricidade, magnetismo e ótica acompanhado pelo crescimento das observações baseadas em princípios matemáticos e físicos. Ainda no século 18 vários outros cientistas avançaram no estudo da energia eletromagnética, começando com os experimentos de eletricidade de

Benjamin Franklin e em 1846 pela proposição feita por Michael Faraday de que a luz e as ondas de rádio são formas de energia eletromagnética. Em 1864, James Clerk Maxwell, publicou sua teoria dos campos eletromagnéticos, concluindo que as energias elétrica e magnética viajam em ondas transversais na velocidade da luz. Em 1887, Heinrich Rudolf Hertz confirmou a teoria de Maxwell produzindo e estudando ondas magnéticas (ondas de rádio) e concluindo que elas poderiam ser refletidas, refratadas e polarizadas assim como a luz. Hertz recebeu o crédito pela primeira transmissão e recepção de ondas de rádio, seguido rapidamente por Aleksandr Popov na Rússia. Já no século 20, em 1906 Ernest F. W. Alexanderson demonstrou o primeiro sinal de rádio gerado continuamente o que foi fundamental para as rádio transmissões modernas. Em 1922 surgiu o radar, que no decorrer da Segunda Guerra Mundial foi aperfeiçoado no Projeto Manhattan no Laboratório Científico de Los Alamos, e foi essencial para o sucesso dos aliados. O radar envia ondas de rádio para detectar a localização de objetos pela reflexão das ondas no objeto. RFID é uma combinação da tecnologia de rádio e do radar, porém ainda segundo [32], não há na literatura especializada uma referência clara de sua invenção. Em 1950 outras tecnologias relativas ao RFID foram desenvolvidas, como o sistema IFF (Identify Friend or Foe), sigla em inglês para Identificação de Amigo ou Inimigo utilizado nas aeronaves de combate. Entre os anos de 1960 e 1980 tecnologias emergentes como o transistor, circuitos integrados, o microprocessador e as redes de comunicação revolucionaram a maneira de fazer negócios. No início dos anos 60, foram fundadas empresas como a Sensormatic, Checkpoint e Knogo,

que desenvolveram o EAS, sigla em inglês para equipamento de vigilância eletrônica de artigos. Estes sistemas consistiam em etiquetas que utilizavam somente um bit de informação e somente a presença ou ausência da etiqueta poderia ser detectada. Como eram muito baratas de ser produzidas estas etiquetas serviram perfeitamente para proteção contra furto. Nos anos 70 desenvolvedores, inventores, companhias, universidades e outros institutos de pesquisa começaram um trabalho intenso com RFID. Nesta época surgiram aplicações como coleta eletrônica de pedágio, identificação de gado, monitoramento de veículos e automação industrial entre outros. Na década de 80 a tecnologia foi largamente implantada em estradas na Europa, nas linhas de montagem e em rebanhos de gado nos Estados Unidos, em aeroportos como em Dallas e portos marítimos em Nova York. De 1990 até os dias atuais as etiquetas RFID têm se disseminado por novas áreas ligadas a logística e rastreamento remoto. Sua utilização é comum nas autoestradas, aeroportos, supermercados, estacionamentos, portos, transportadoras e os mais variados setores principalmente nos Estados Unidos e Europa. Neste momento os esforços estão voltados para a regulamentação de padrões da tecnologia e para o aprimoramento do hardware, procurando principalmente a redução dos custos de produção e da tecnologia de software, em busca de melhores algoritmos para processamento das informações e protocolos de comunicação que propiciem a segurança e privacidade dos usuários.

2.2 Definições

Segundo [4], Identificação por Rádio Freqüência é um método para armazenar e recuperar remotamente dados utilizando mecanismos chamados **etiquetas RFID**. Em sua versão mais simples, uma etiqueta RFID é um componente eletrônico formado por uma antena e um microprocessador capaz de trocar informações com um aparelho leitor via ondas de rádio. Este microprocessador é dotado de memória para armazenar informações e uma pequena capacidade de processamento. A vida útil de uma etiqueta de RFID em uma cadeia de suprimentos se inicia no momento em que a etiqueta é fabricada. Abaixo segue uma cronologia da etiqueta durante toda a sua vida útil:

1. Fabricação da Etiqueta: além da montagem dos componentes eletrônicos, a etiqueta recebe o software responsável pela operação do hardware e um número identificador único para esta etiqueta (ID).

2. O fabricante da etiqueta vende um lote de etiquetas a uma empresa que a utilizará para identificar um produto. Com o lote de etiquetas segue um banco de dados com as informações referentes aquele lote de etiquetas.

3. A nova empresa proprietária das etiquetas fixa as etiquetas nos produtos e atualiza seu banco de dados associando o ID da etiqueta ao produto em questão.

4. A partir deste momento a etiqueta está apta a identificar o produto por

rádio freqüência.

5. Leitores de RFID, devidamente autorizados pelo empresa proprietária das etiquetas poderão ler as informações na etiqueta. Isto pode incluir toda a cadeia de logística desde transportadoras até armazens no atacado e varejo.

6. Um eventual novo proprietário, que adquira o produto identificado pela etiqueta, tem duas alternativas: a) inutilizar a etiqueta fisicamente ou através de um comando específico; b) reutilizar a etiqueta.

A contribuição deste trabalho se localiza nos momentos 4, 5 e 6 da vida útil da etiqueta RFID, ou seja, no momento em que a etiqueta é utilizada para identificar o respectivo produto.

2.3 Classificação e Padronização

Etiquetas podem ser classificadas de acordo com sua fonte de energia como **passivas**, **semi-passivas** ou **ativas**, ou por sua freqüência de transmissão (Alta Freqüência (HF) ou Ultra Alta Freqüência (UHF)). As etiquetas passivas não possuem fonte de energia própria, obtendo a energia necessária a partir do campo eletromagnético gerado pelo leitor. Este tipo de etiqueta pode responder ao leitor a uma distância média de um metro e tem pequena capacidade de memória e processamento. Segundo [3] o preço atual por unidade é US$ 0,20. As etiquetas semi-passivas têm sua própria fonte de energia, alcance entre 1 e 30 metros, custo

em torno de US$ 1,00 cada e responde somente quando contactada pelo leitor. As etiquetas ativas, além de possuir sua própria fonte de energia, têm alcance entre 30 e 100 metros, custam entre US$ 1,00 e US$ 10,00 e além de responder ao leitor também podem iniciar uma comunicação. O Laboratório AutoID [6], classifica as etiquetas RFID da seguinte maneira:

- Classe I - entre 1000 e 4000 portas lógicas e entre 100 e 200 bits de memória somente de leitura (ROM), ou memória de gravação única e leitura múltipla (WORM)
- Classe II - possuem um pouco mais de 4000 portas lógicas e memória de leitura aleatória (RAM)

Para utilização em larga escala, as etiquetas devem ter um baixo custo de produção o que resulta em limitações na capacidade de memória e processamento. Estas restrições forçam a utilização de soluções de baixo consumo de energia, com memória limitada. A utilização de etiquetas RFID na cadeia de suprimentos motivou a criação do EPC Global, um consórcio de empresas cujo objetivo é etiquetar cada unidade de cada item de produto com etiquetas cujo custo de produção não ultrapasse os US$ 0,05 [51]. O EPC Global também especifica um código eletrônico de produto para identificação única de cada instância de cada objeto já etiquetado.

2.4 Segurança e Privacidade

As implementações iniciais de etiquetas RFID forneciam um número de identificação (ID) como único identificador da etiqueta. Este código único permite que a etiqueta seja rastreada geograficamente, e além disso, produtos com esta etiqueta podem ser identificados remotamente. Este ID sempre é fornecido a qualquer leitor que o solicite. O notório aumento do número de etiquetas RFID tem levado a uma maior preocupação relacionada à segurança e à privacidade. Segundo [5] os principais tipos de ataques a sistemas de RFID são:

- Negação de serviço
- Impersonificação
- Vazamento de Informação
- Rastreamento Malicioso

2.5 Negação de serviço

Neste tipo de ataque um adversário pode interferir na comunicação entre o leitor e a etiqueta de duas maneiras: 1. utilizando aparelhagem eletrônica capaz de interferir nas ondas de rádio geradas pelo leitor/etiqueta, impedindo a comunicação entre ambos; 2. sobrecarregar o leitor ou etiqueta com leituras válidas ou inválidas em uma taxa superior a suportada pelo hardware impedindo

ou adiando a resposta do leitor/etiqueta verdadeiro. Em ambos os casos um adversário seria capaz de furtar um produto, passando pelo leitor verdadeiro sem ser detectado devido à negação de serviço.

2.6 Impersonificação

Qualquer adversário diferente da etiqueta que utilize um protocolo de comunicação se fazendo passar pela etiqueta pode fazer com que o leitor complete a consulta e aceite o adversário como se fosse a etiqueta verdadeira. Impersonificar uma etiqueta RFID implica em se comportar como uma etiqueta real perante um leitor. Este conceito está diretamente relacionado com identificação. Para prevenir este tipo de ataque deve-se garantir a autenticação da etiqueta perante o leitor e vice-versa. Um tipo de ataque clássico ocorre quando um adversário se coloca entre o leitor e a etiqueta no campo de leitura (Fig. 2.1)

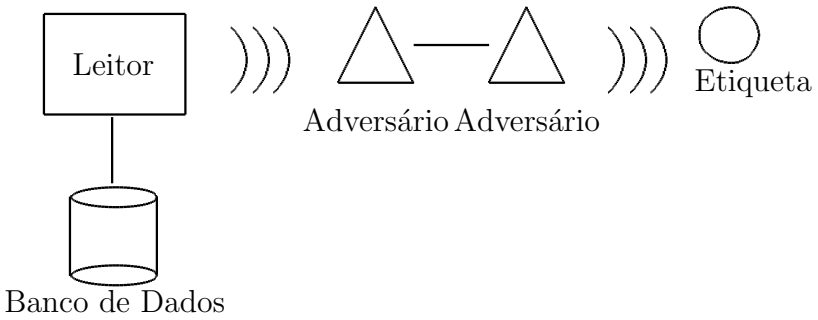

FIGURA 2.1 – O adversário se interpõem entre o leitor e a etiqueta interceptando a consulta à etiqueta. Fonte: [4]

2.7 Vazamento de Informação

Este tipo de problema ocorre quando a informação enviada pela etiqueta revela informações intrínsecas ao produto a que está associada. Livros em bibliotecas, produtos farmacêuticos, passaportes e cartões de identificação podem revelar informações sobre si mesmos ou sobre os proprietários do referido item. Portanto, etiquetas que não forneçam um protocolo para proteger estas informações podem ser uma fonte fácil de dados, colocando em risco a segurança e privacidade do usuário.

2.8 Rastreamento Malicioso

Dada uma série de leituras entre as etiquetas e os leitores, um adversário pode ser capaz de encontrar algum tipo de correlação entre várias leituras de uma mesma etiqueta ou conjunto de etiquetas. Sendo assim seria possível que o patrão rastreie seus empregados, que uma criança seja rastreada em um parque ou que tropas militares sejam rastreadas no campo de batalha. Pelo tamanho das etiquetas, algumas com dimensões milimétricas, elas podem ser escondidas no produto, o que impede que o usuário tenha prévio conhecimento de que esteja sendo rastreado.

2.9 Protocolo de Comunicação para RFID

Alguns mecanismos de segurança tem sido estudados e melhorados para prevenir ou diminuir a possibilidade de fraudes ou ataques às etiquetas RFID. Esses mecanismos utilizam protocolos criptográficos como em [22] e [28], e protocolos de comunicação específicos para prevenir o acesso às informações da etiqueta por adversários não autorizados. Devido a natureza do sistema de RFID, o protocolo de comunicação passa a ter vital importância para a segurança da tecnologia. Ainda segundo [5], um protocolo básico de comunicação é aquele que implementa somente a Identificação da etiqueta. Entende-se por Identificação que o leitor obtém uma identificação da etiqueta, porém sem prova de que esta identificação é correta. (Fig. 2.2)

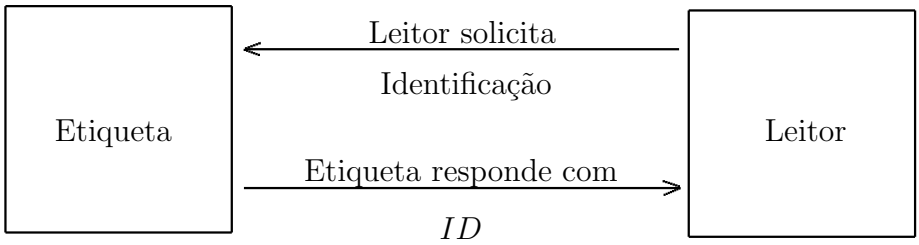

FIGURA 2.2 – Esquema típico de identificação, onde a etiqueta fornece um ID fixo para todas as consultas realizadas pelo leitor. Fonte: [1]

O próximo passo é a autenticação: o leitor obtém a identidade de uma etiqueta e uma prova que esta identidade é correta. A prova consiste em que ambos, leitor e etiqueta compartilham uma função de hash h secreta, que utilizam para verificar a autenticidade das mensagens. Ambos aplicam a mesma função sobre os mesmo

valores. Caso os resultados confiram há a autenticação. Um esquema muito utilizado é a autenticação por desafio-resposta, onde o leitor gera um número pseudo aleatório N e a envia para a etiqueta. A etiqueta responde com dois números: ID (identificação da etiqueta) e $h(ID, N)$ (número gerado por uma função de ID e N). (Fig. 2.3).

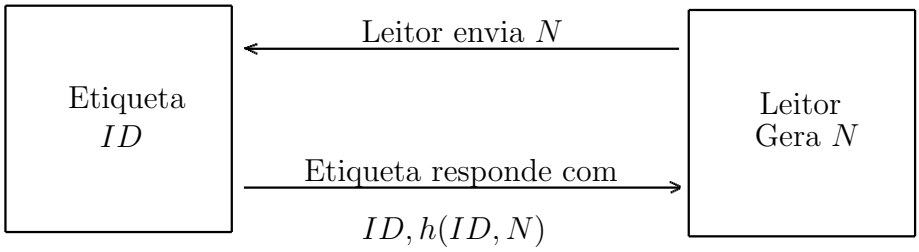

FIGURA 2.3 – Autenticação no esquema de desafio-resposta. Fonte: [1]

2.10 Um protocolo simples para RFID

Em alguns trabalhos anteriores sobre RFID foram pesquisados diversos protocolos para etiquetas de baixo custo (baixa capacidade de processamento e memória), como o esquema Hash Lock citado em [21]. Este esquema é de baixo custo pois tudo o que ele necessita é de uma função de hash h. Cada etiqueta verifica o leitor da seguinte maneira: o leitor possui uma chave k para cada etiqueta, e cada etiqueta possui o resultante metaID, onde metaID $= h(k)$. A etiqueta recebe uma solicitação de acesso ao ID e envia o metaID como resposta. O leitor envia uma chave que é relacionada ao metaID recebido pela etiqueta. A etiqueta então

calcula a função de hash da chave recebida e compara se o resultado da função de hash corresponde ao metaID armazenado na etiqueta. Somente se ambos os resultados conferem a etiqueta envia seu próprio ID para o leitor.

Ohkubo *et al.* [42] propôs uma nova versão do Hash Lock que requer que a etiqueta possua uma função de hash h e um gerador de números pseudo-aleatórios. Cada etiqueta calcula a função de hash baseada em um número pseudo-aleatório (r) e na identificação da etiqueta (id), ou seja, $c = h(id|r)$. A etiqueta , então, envia c e r para o leitor. O leitor envia os dados ao banco de dados. O banco de dados calcula a função de hash para o r recebido e para o id armazenado no banco de dados. O banco de dados identifica, então o id relacionado com o c recebido e envia o correspondente id para o leitor.

Juels [28] propôs um mecanismo de desafio-resposta com aplicação de pseudônimos para melhorar a privacidade das etiquetas RFID. Este protocolo é baseado na rotação de pseudônimos que chamamos de α_1, α_2 ..., α_k, para cada etiqueta. Estes pseudônimos, porém, não funcionam como autenticação para a etiqueta. Uma etiqueta só se autentica ao leitor após o leitor ter se autenticado perante a etiqueta. O leitor (verificador) se autentica perante etiqueta fornecendo uma chave β_i. Esta chave β_i é única para um dado pseudônimo α_i. Uma vez que o leitor está autenticado perante a etiqueta, a etiqueta autentica a si mesma informando a chave de autenticação γ_i. Assim como β_i, esta chave de autenticação γ_i é única para o identificador α_i. Em outras palavras, o protocolo proposto é do

tipo desafio-resposta, porém intercalado com rotação de pseudônimos.

Henrici e Muller em [27] propuseram um protocolo que tenta distinguir os IDs das etiquetas utilizando uma função de hash baseada em um número de transação (TID). Na fase inicial deste protocolo a etiqueta não expoẽm outra informação a não ser o hash de seu ID, que é utilizado para identificar e acessar a etiqueta. Quando consultada a etiqueta incrementa em uma unidade seu número de transação (TID) e envia a seguinte informação: $h(ID)$, identificador do banco de dados ($DBID$), hash $h(TID \circ ID)$ e a diferença entre a transação corrente e a última transação completada com sucesso ($\Delta TID = TID - LST$). Nessa mensagem, $h(ID)$ identifica e etiqueta no banco de dados $DBID$. $h(TID \circ ID)$ tem o propósito de combater os ataques de clonagem: ela muda a cada tentativa de leitura e é verificada pelo receptor legítimo. $\Delta TID = TID - LST$ é utilizado pelo receptor legítimo para recalcular o TID da etiqueta obtendo o ($TID*$) corrente utilizado pela etiqueta. Como esse número é apenas a diferença entre duas transações nenhuma informação útil a um atacante é transmitida em aberto. No banco de dados o registro com $HID = h(ID)$ é selecionado, a LST e ΔTID são somados para se obter a transação corrente. $h(TID \circ ID)$ é calculado e se o valor conferir com o valor recebido ($h(TID \circ ID)$) então o $TID*$ é armazenado no banco de dados e uma nova $ID*$ é gerada a partir de um número aleatório (RND): $ID* = RND \circ ID$. O banco de dados é atualizado com o novo $ID*$, HID e $h(ID)$ e o antigo TID e substituído pelo atual $TID*$. Agora a mensagem

contendo RND e o $h(RND \circ TID* \circ ID)$ é criado e enviado à etiqueta. A etiqueta calcula o hash e se o valor não conferir a mensagem é descartada. Caso contrário a etiqueta atualiza sua ID com $RND \circ ID$ e seu LST com o valor de TID. Agora a etiqueta possui um novo ID que será transformado em $h(ID)$ para a próxima consulta. Todos estes protocolos possuem fraquezas que podem expor as etiquetas a algum tipo de ataque, além de não prover nenhum mecanismo para transferência de propriedade das etiquetas. Dimitriou [20] propôs um protocolo leve [1] para RFID de baixo custo, baseado no mecanismo de desafio-resposta. Este protocolo permite tanto a autenticação do leitor com a etiqueta quanto da etiqueta para o leitor, prevenindo o ataque por clonagem e a utilização de leitores não autorizados. Segundo Dimitriou este protocolo se mostrou resistente à maioria dos ataques conhecidos. Este protocolo é baseado em um segredo compartilhado entre a etiqueta e o leitor (presume-se que há uma ligação segura entre o leitor e o banco de dados do sistema), e na troca da identificação da etiqueta a cada sessão de autenticação, tornando impossível que um adversário possa distinguir entre os IDs verdadeiros e números gerados pseudo-aleatoriamente em cada sessão. Na figura 2.4 a função de hash h do valor M, denominada $h(M)$, e a MAC de N_T, N_R é denominada $h_{ID_i}(N_T, N_R)$. A notação da Tabela 2.1 será utilizada para explicação detalhada.

[1] Protocolo que utiliza poucas operações matemáticas podendo ser implementados em etiquetas de RFID com pouca capacidade de processamento.

TABELA 2.1 – Notação para o protocolo RFID.

Notação	Significado
T	Etiqueta RFID
R	Leitor de RFID consultando a etiqueta
ID_i	Identificação da etiqueta durante a i-ésima consulta
M_1, M_2	Concatenação das mensagens M_1 e M_2
$h(M)$	Aplicação da função de hash h em M
$h_k(M)$	Aplicação da função de hash com chave k (MAC) em M
N	Número gerado aleatoriamente

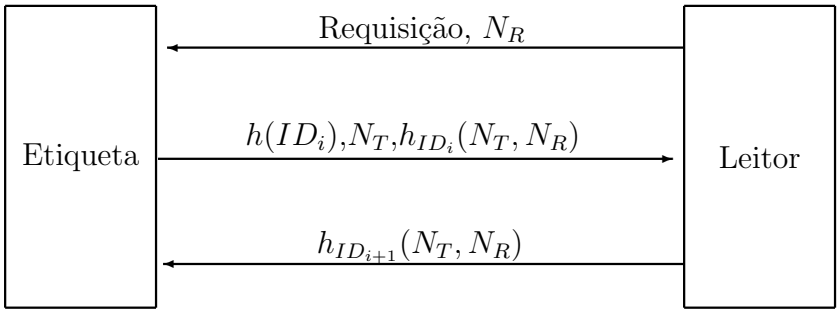

FIGURA 2.4 – Esquema do protocolo original (ciclo original de comando). Fonte: [1]

2.11 Transferência de Propriedade

Um dos pontos mais importantes apresentados pelo protocolo de Dimitriou é o ID_i secreto, compartilhado entre o leitor (Banco de Dados) e a etiqueta. Durante o processo de leitura podemos desejar que uma única parte gerencie o acesso a várias etiquetas. Molnar em [37] assume a presença de uma central confiável, chamada Trusted Center (TC). De acordo com Molnar o TC atua como uma terceira entidade que gerencia as políticas de privacidade associadas às etiquetas. Apesar do fabricante de etiquetas RFID poder atuar como TC na prática isso não é requerido. Etiquetas RFID podem ser emitidas sem armazenar segredo

algum. Quando a etiqueta for utilizada pela primeira vez o TC pode então escrever os segredos relevantes. Pode-se construir etiquetas que somente podem ser escritas desta maneira uma vez, e então não permitir reescrita ou leitura posterior dos segredos. Se um leitor, A, deseja determinar o ID da etiqueta, ela deverá perguntar ao TC. O TC pode então, determinar se A está autorizada a ver a informação baseada na política de privacidade da etiqueta, armazenada no banco de dados. Se decidirmos transferir a propriedade da etiqueta de A para B, nós devemos assegurar que após a transferência, A não será capaz de ler as etiquetas. Desta maneira, A delega a B a propriedade da etiqueta e B se torna o novo TC. A contribuição desta parte deste trabalho é uma melhoria no protocolo de comunicação sugerido por Dimitriou adicionando um mecanismo seguro para transferência de propriedade de um TC para outro na cadeia de suprimentos.

2.12 Novo Protocolo

Há várias aplicações práticas para RFID na cadeia de suprimentos. Uma das mais importantes e talvez a mais imediata é o rastreamento de produtos através da cadeia de suprimentos. Uma vez que um fabricante anexa uma etiqueta RFID a seu produto, ele deveria ter certeza de que todas as medidas de segurança e privacidade foram devidamente implementadas. Etiquetas desprotegidas podem ter seus produtos escaneados por qualquer indivíduo que possua leitores comuns. Além disso o ID da etiqueta está diretamente relacionado com o EPC Electronic

Product Code do produto e a lista de EPC é disponível publicamente. Assumindo-se que o protocolo proposto em [20] leva em consideração os itens de segurança e privacidade (análise de segurança detalhada na seção 2.14), o presente trabalho propõe um mecanismo para garantir a transferência segura de propriedade sem riscos a privacidade. Esta contribuição é uma melhoria do protocolo desenvolvido para etiquetas de baixo custo que se beneficia do mecanismo de autenticação mútua para permitir ao TC atualizar a política de privacidade associada a etiqueta. O novo protocolo foi chamado de ROTEP, sigla para RFID Ownership Transfer Enable Protocol. A solução é baseada no valor de ID_i compartilhado entre o leitor (Banco de Dados) e a etiqueta. Supondo que o proprietário atual (chamado A) deseja transferir a propriedade da etiqueta (T) para outro proprietário (chamado B), será necessário que:

1. A transfira as informações do banco de dados sobre o produto associado a T para o novo proprietário B (assume-se que esta transferência é feita por um meio seguro).

2. A aciona o modo de transferência (TM) de T. (somente A conhece o comando para acionar o TM).

3. A faça a transferência física de T para B.

Para que T saia do TM, é preciso que esta receba o comando de saída que somente o novo proprietário B conhece, caso contrário T responderá a todas as consultas com um número aleatório qualquer, sem utilidade para identificar a etiqueta. Para sair do TM será necessário:

5. B envia para T o comando de saída do TM.

6. B troca de maneira segura um novo valor secreto para ID_{i+1} conhecido somente por T e B.

7. B passa a ser a única entidade capaz de se comunicar com T.

O mecanismo proposto para colocar a etiqueta em modo de transferência (TM) está na Fig. 2.5 e é uma variação do protocolo descrito na Fig. 2.4. Na Fig. 2.6 vemos o mecanismo para saída do TM e geração do novo segredo entre a etiqueta e B.

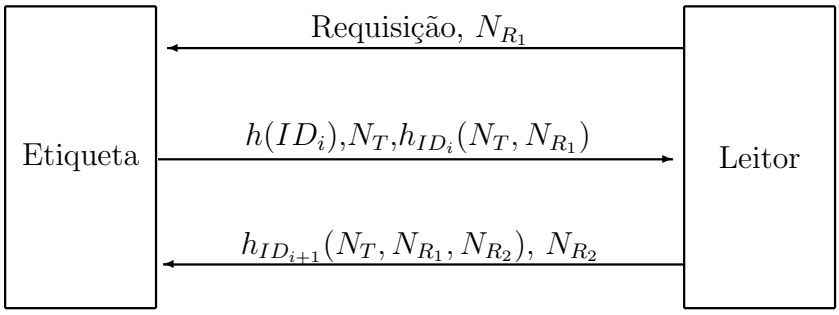

FIGURA 2.5 – Ciclo de comunicação para o modo de transferência. Um número pseudo-aleatório único N_{R_2} gerado pelo leitor coloca a etiqueta em modo de transferência de propriedade. Fonte: [1]

Protoloco para colocar a etiqueta em modo de transferência:

1. O leitor transmite a requisição e N_{R_1}.

2. A etiqueta gera o novo identificador N_T e devolve $h(ID_i)$, N_T e $h_{ID_i}(N_T, N_{R_1})$ para o leitor. O banco de dados associado ao leitor autentica a etiqueta e gera uma nova identidade , ID_{i+1}.

3. O banco de dados constrói a mensagem $h_{ID_{i+1}}(N_T, N_{R_1}, N_{R_2})$, utilizando a nova chave ID_{i+1}, e a envia ao leitor que reencaminha para a etiqueta. Como a etiqueta e o leitor compartilham o algoritmo para geração de ID_{i+1}, a etiqueta gera uma nova chave ID_{i+1} e computa o valor de $h_{ID_{i+1}}$ (N_T, N_{R_1}, N_{R_2}). Se o valor recebido é o mesmo que o computado pela etiqueta, esta aceita a resposta como autêntica e então coloca a etiqueta em modo de transferência (TM), apagando a chave antiga ID_i e os números N_T, N_{R_1} da memória. A etiqueta utiliza ID_{i+1} como seu novo identificador. Caso contrário a etiqueta rejeita a resposta e mantém a chave antiga ID_i e o protocolo no estado original.

4. Se o comando para entrada no TM é completado com sucesso a etiqueta entra em modo de transferência e de agora em diante irá responder a qualquer solicitação de comunicação com um número pseudo-aleatório que será diferente a cada consulta.

Uma vez em TM a etiqueta está pronta para ter sua propriedade transferida de A para B. Para que o novo proprietário B possa sair do modo de segurança

este utiliza a mensagem previamente fornecida por A: $h_{ID_{i+1}}$ (N_{R_2}). Porém o antigo proprietário, que conhece $h_{ID_{i+1}}$ (N_{R_2}), poderia de maneira maliciosa, reassumir o controle da etiqueta, caso mantivéssemos o segredo compartilhado pelo par etiqueta-A com etiqueta-B (ID_i). Para evitar que o antigo proprietário A tenha acesso à etiqueta depois de transferida para B modificaremos o segredo compartilhado (ID_i) entre B e a etiqueta de forma segura e a partir deste momento somente B e sua etiqueta poderão comunicar-se seguramente. Para gerar este novo segredo ID_i utilizamos o protocolo de Diffie-Hellman [36]. Através deste protocolo é possível que duas entidades, T (etiqueta) e B (novo proprietário) compartilhem um segredo(ID_{i+2}), publicamente sem risco de violação do segredo. Isso é possível devido a complexidade do problema do logaritmo discreto. Fig. 2.6.

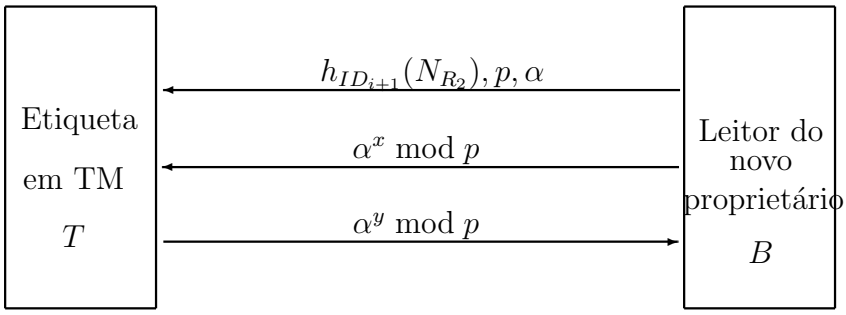

FIGURA 2.6 – A etiqueta T recebe o comando para saída do TM, o número primo p e o gerador α. Em seguida T e B trocam as mensagens que permitem o cálculo do novo ID_i^*. Fonte: [1]

Protocolo para tirar a etiqueta do modo de transferência:

1. O novo proprietário B comanda a saída do modo de transferência transmitinto a etiqueta T o comando: $h_{ID_{i+1}}$ (N_{R_2}), p, α. Onde $h_{ID_{i+1}}$ (N_{R_2})

é o comando de saída propriamente dito, p é um número primo e α é um gerador de $Z_p^*(2 <= \alpha <= p-2)$.

2. A etiqueta gera um novo valor ID_{i+1} e computa $h_{ID_{i+1}}$ (N_{R_2}), se o valor não coincidir a etiqueta responde com um número aleatório e o processo termina aqui, caso contrário T armazena em sua memória p e α para uso posterior e continua em 3.

3. T gera aleatoriamente um número secreto x, onde $1 <= x <= p-2$ e transmite para B: $\alpha^x \bmod p$.

4. B gera um número secreto aleatório y, onde $1 <= y <= p-2$ e transmite para T: $\alpha^y \bmod p$.

5. B recebe a mensagem de T item (3) e computa $ID_i^* = (\alpha^x)^y \bmod p$.

6. T recebe a mensagem de B item (4) e computa $ID_i^* = (\alpha^y)^x \bmod p$.

7. Tanto T quando B passam a considerar como novo segredo compartilhado ID_i^*. Daí em diante o protocolo volta a seu ciclo normal de operação.

A principal diferença entre o protocolo original e o ROTEP (com TM) é a geração do N_{R_2} pelo leitor que é enviado à etiqueta em dois formatos: sem ciframento, e processado por uma função de hash h. Se a etiqueta computar um MAC com estes três valores N_T, N_{R_1} e N_{R_2} e o resultado coincide com o valor recebido, o modo de transferência é iniciado.

Em um ciclo normal de comunicação, a cada consulta à etiqueta, esta responderá conforme o protocolo original. Caso o TC utilize o comando para TM, a etiqueta entra em TM. Neste momento podemos transferir a etiqueta para seu novo proprietário que deverá ativar o comando para saída do TM e obter o novo segredo ID_i^* para voltar a utilizar a etiqueta normalmente (Fig. 2.7).

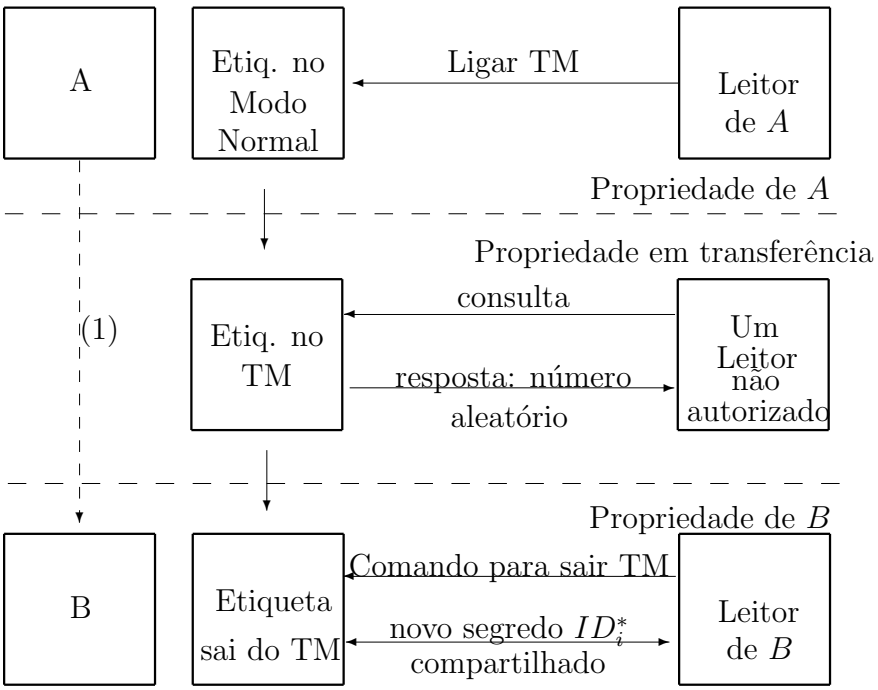

FIGURA 2.7 – Três diferentes estados para as etiquetas: Propriedade de A, em transferência de propriedade e propriedade de B. (1) indica a direção do fluxo de transferência. Fonte: [1]

Recomenda-se a utilização de uma senha para evitar acesso não autorizado ao modo de transferência, assim como a utilização de um ambiente seguro para transferir as informações sobre a etiqueta entre A e B. Uma vez que A não terá acesso ao comando de B para gerar o novo ID_i^*, garantido pelo protocolo

de Diffie-Hellman [36], ela não será mais capaz de consultar a etiqueta. Se B precisar transferir a propriedade da etiqueta novamente, basta repetir o processo.

Quando a etiqueta está no modo de transferência, é possível que durante o transporte ou manuseio o produto que possui a etiqueta seja desviado de seu curso normal ou mesmo furtado. Neste caso a etiqueta seria inútil, pois somente iria responder as consultas com números pseudo-aleatórios sem significado real. Para garantir que a etiqueta não gere ruído de rádio freqüência gerando uma grande quantidade de números inúteis quando corretamente consultados pelo leitor, podemos utilizar o comando de **killing**, proposto por Garfinkel em [25], um mecanismo de auto-destruição que fará a etiqueta parar de funcionar após certo número de acessos falhos.

2.13 Custo Computacional

Um dos principais aspectos do protocolo proposto por Dimitriou é o segredo compartilhado entre o leitor (banco de dados) e a etiqueta. Em cada consulta o leitor deve computar uma função de hash $h(x)$ ou de MAC $h_k(x)$. Então, para propósito de comparação, vamos assumir que o custo computacional será somente a busca no banco de dados. O leitor depende da busca no banco de dados para encontrar um ID_i válido. Este esquema requer um trabalho linear $O(n)$. De acordo com Molnar [37], outros esquemas baseados em árvores balanceadas po-

dem diminuir o tempo de pesquisa no banco de dados para $O(\log n)$. Outras técnicas para reduzir o tempo de pesquisa no banco de dados é a delegação a um determinado leitor de um grupo de etiquetas para serem acessadas somente por aquele leitor que estão diretamente associados àquelas etiquetas. Se considerarmos D o número de etiquetas delegadas e n o número total de etiquetas é esperado que D seja muito menor que n e portanto o trabalho será $O(D)$.

2.14 Análise de Segurança do Protocolo Original

A segurança do protocolo original é baseada no ID secreto compartilhado entre o banco de dados do leitor e a etiqueta. No momento em que a etiqueta é confeccionada, o fabricante deve criar um ID inicial para a etiqueta e este ID deve ser armazenado no banco de dados. Cada companhia que utilizar estas etiquetas deverá ter acesso a este banco de dados.

Outra possibilidade é criar as etiquetas sem ID inicial para que o primeiro TC proprietário da etiqueta possa inicializá-la com um comando especial e a partir deste ponto trabalha com o protocolo de comunicação normal.

O protocolo utiliza um função de hash para enviar o ID para a etiqueta. Depois de receber a mensagem a etiqueta computa a mesma função de hash com seu próprio ID. Se o resultado calculado pela etiqueta confere com o resultado

recebido na mensagem então a etiqueta considera o leitor como confiável.

Após o leitor autenticar a etiqueta, esta utiliza o mesmo algoritmo para autenticar a si mesma perante o leitor. No próximo passo, ambos, leitor e etiqueta trocam seu ID corrente por um novo. A etiqueta e o leitor geram um novo ID utilizando o mesmo algoritimo em cada transação onde se autenticam com sucesso.

Um extensiva análise de segurança foi realizada neste protocolo de acordo com [20]. Uma vez que o comando que permite a transferêcia de propriedade não modifica o esquema do protocolo original de Dimitriou, nós vamos somente repetir aqui a análise realizada em [20]:

Ataque na etiqueta - o adversário não conhece o segredo compartilhado entre a etiqueta e o leitor, então ele não pode autenticar a si mesmo perante a etiqueta. Assumindo que o adversário conheça a função de hash utilizada, a complexidade da computação de uma busca exaustiva dependerá do tamanho da chave.

Ataque ao leitor - Este tipo de ataque será evitado com o esquema de autenticação mútua. O adversário deve se autenticar com sucesso perante a etiqueta antes de tentar se autenticar ao leitor, o que não é possível devido ao esquema de ID secreto compartilhado.

Ataque na comunicação entre a etiqueta e o leitor - o adversário deve interceptar e bloquear a comunicação. Além disso o adversário deve ser capaz de rastrear a

constante mudança no ID em cada transação de consulta.

Ataque a privacidade do usuário - o ID de cada etiqueta é diferente em cada sessão de leitura. Se o adversário tentar consultar uma etiqueta com uma seqüência errada, ele não receberá resposta alguma da etiqueta.

Privacidade de localização - Como o ID da etiqueta nunca é o mesmo, a sua localização geográfica não pode ser rastreada.

Ataque contra a chave - se a função de hash e o tamanho da chave são cuidadosamente escolhidos este ataque é computacionalmente intratável de acordo com [36].

Ataques contra a implementação - é importante evitar a utilização de $NR = 0$, e não permitir que o mesmo ID seja gerado em seqüência ($\mathrm{ID}_i = \mathrm{ID}_i + 1$).

2.15 Análise de Segurança do Novo Protocolo

As diferenças entre as duas implementações são:

1. O ROTEP possui um comando para entrada no TM.

2. O ROTEP possui um comando para saída do TM.

3. O ROTEP troca um novo segredo para ID_i utilizando o protocolo de Diffie-Hellman [36]

Analisando-se o item 1 acima, o ROTEP possui uma informação extra adicionada como última parte do ciclo de comunicação ($h_{ID_{i+1}}(N_T, N_{R_1}, N_{R_2})$). Esta informação extra permite que a etiqueta entre em modo de transferência (TM). Neste estado, a etiqueta irá responder a todas as consultas com números pseudo-aleatórios exceto ao comando para saída do modo de transferência. O comando para entrar no modo de transferência utiliza o mesmo esquema de autenticação mútua do protocolo original somente adicionando um terceiro número pseudo-aleatório, N_{R_2}, que irá identificar o ciclo de comunicação corrente como o comando para o modo de transferência de propriedade. Assim como no protocolo original, este número extra , N_{R_2}, é autenticado pela etiqueta utilizando o novo ID secreto, compartilhado entre o leitor e a etiqueta que nunca será exposto claramente (pela utilização de uma função de hash). Portanto a adição deste comando adicional não implica em vulnerabilidade para o novo protocolo.

Análise semelhante pode ser feita em relação ao item 2, pois o método é idêntico ao item 1.

Portanto para os itens 1 e 2, o ROTEP mantem a mesma resistência aos ataques do protocolo original.

Um ataque protocolo de Diffie-Hellman (item 3) é pouco provável, pois o adversário deverá lidar com a complexidade dos logaritmos discretos. Ver seção 2.11.

2.16 Conclusão

Segurança e privacidade são assuntos muito relevantes e devem ser considerado em qualquer sistema de informação. Para se preservar a segurança e privacidade de etiquetas RFID, protocolos de comunicação seguros devem ser utilizados para proteger informações sensíveis contra espionagem e ataques ativos. Reutilização de etiquetas através de toda a cadeia de suprimentos é um ponto importante na tecnologia de RFID e na economia de escala para diferentes proprietários de produtos como fabricantes, transportadores, intermediários, vendedores, donos de lojas e consumidores finais. O ROTEP permitirá a transferência de propriedade levando em consideração os aspectos de segurança e privaciadade.

Essa transição de propriedade poderá ser realizada pelo TC (com uma senha), portanto ninguém mais poderá comandar a etiqueta para entrar no modo de transferência. O TC pode decidir onde e quando comandar o modo de transferência. Os comandos extras e a utilização de Diffie-Hellman não modificam o resistência do protocolo original a ataques conhecidos mesmo quando a etiqueta está em modo de transferência. Esta melhoria é uma alternativa para aumentar o ciclo de vida da etiqueta, permitindo sua utilização através de toda a rede de suprimentos enquanto preserva a privacidade e segurança.

3 Nova matriz MDS involutória para cifra de bloco AES

3.1 Introdução

O algoritmo AES (Advanced Encryption Standard) é uma cifra do tipo SPN (Substitution Permutation Network) desenvolvida por J. Daemen e V. Rijmen para o processo de seleção do AES [2]. A cifra original é chamada Rijndael e foi selecionada entre 15 candidatos durante o processo de seleção para o AES, por iniciativa do NIST (National Institute of Standards and Technology) dos EUA em 1997. O AES se tornou o novo padrão mundial em criptografia simétrica, como sucessor do algoritmo DES (Data Encryption Standard). Em setembro de 2001, Rijndael foi oficialmente padronizado como FIPS PUB 197 [40]. Rijndael e o AES já foram implementados em diferentes linguagens de programação e fazem parte de inúmeros sistemas de software [48]. O AES é a menor instância da cifra Rijndael [19], pois o AES opera com blocos de texto de 128 bits e chaves de 128,

192 ou 256 bits, para as quais a cifra itera dez, doze ou quatorze rodadas, respectivamente. Há quatro transformações em uma rodada completa de Rijndael: SB (SubBytes), SR (ShiftRows), MC (MixColumns) e AK_i (AddRoundKey). O índice i representa o número da rodada. Uma rodada completa do Rijndael aplicada a um bloco de texto X pode ser denotada por $AK_i \circ MC \circ SR \circ SB(X) = AK_i$ (MC (SR (SB(X)))). Há uma transformação inicial, AK_0, somente na primeira rodada e a última rodada não inclui MC.

No AES os blocos de texto, chave e subchaves são representados por uma matriz 4xNb de bytes, chamada matriz de estados, onde Nb é o número de palavras de 32 bits no dado de entrada. Por exemplo, um bloco de 128 bits (texto legível ou cifrado) $A = (a_0, a_1, .., a_{15})$ pode ser representada de forma compacta como uma matriz de estados 4x4 como segue:

$$A = \begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}.$$

Pode-se separar o funcionamento do AES em dois processos pararelos: A) Encriptação/Decriptação e B) Processamento da Chave. O processo A é composto pelas operações:

1. SB - Cada byte da mensagem é substituido por outro determinado por uma

tabela de substituição de dimensão 8 x 8 chamada caixa-S.

2. SR -Cada matriz de estado tem a linha rotacionada (para a esquerda) em 0, 1, 2 e 3 bytes respectivamente.

3. MC - Cada coluna da matriz de estado é multiplicada por uma matriz MDS 4x4.

4. AK - ou-exclusive byte a byte entre a matriz de estados e a matriz na i-ésima sub-chave.

No processo B são realizadas as seguintes operações com a chave:

1. Uma sub-chave é gerada por uma operacção de ou-exclusivo entre cada coluna da matriz de estados da chave e uma tabela de substituição.

2. Novas sub-chaves serão geradas conforme o número de rodadas do algoritmo.

3. Cada sub-chave gerada para cada rodada é utilizada na na operação AK do processo A.

Este trabalho propoẽ uma nova camada de difusão para a cifra AES, que substitui as camadas SR e MC. Esta nova camada consiste em uma matriz MDS involutória 16×16, chamada $M_{16\times16}$. O novo algoritmo foi chamado MDS-AES. Portanto a nova estrutura de uma rodada completa do MDS-AES se torna $\mathrm{AK}_i \circ M_{16\times16} \circ \mathrm{SB}(X) = \mathrm{AK}_i\ (M_{16\times16}\ (\mathrm{SB}(X)))$. Esta nova estrutura tem

duas conseqüências: (i) difusão completa é alcançada em uma única rodada o que melhora a segurança da cifra porque o fator de difusão [18, 19] aumenta de 5 (no AES) para 17; porém, (ii) diminui a performance devido ao grande número de operações elementares da nova matriz 16×16.

O fator de Difusão é uma característica da cifra que é herdada das matrizes MDS. Considerando uma entrada que é uma palavra de 32 bits, com o menor número de bytes de diferença não nula, ou seja, 1 byte, teremos uma saída onde todos os bytes de diferença serão necessariamente não nulos. A soma dos bytes de diferença não nulos na entrada e na saída é pelo menos 5. Este é o Fator de Difusão do AES [18]. Para o MDS-AES o fator de difusão será 17, por que 1 byte não nulo na entrada vai gerar 16 bytes não nulos na saída.

3.2 Preliminares

Uma transformação involutória ou simplesmente uma involução f satisfaz, $f(x) = f^{-1}(x)$, para todo x no domínio de f. A utilização de involuções na criptologia data da época das cifras ATBASH, ALBAM e ATBAH [30], da máquina alemã Enigma [30] e mais recentemente das cifras de bloco Khazad [8] e Anubis [7].

Há várias razões para a utilização de transformações involutórias. Este mapeamento reduz tanto o custo da operação de encriptação quanto da decriptação. Este mapeamento implica que ambas possuem a mesma segurança criptográfica.

É importante enfatizar que fazer a camada de difusão do AES uma involução não transforma toda a cifra em uma involução (porque a caixa S-não é uma involução).

$Caixa\text{-}s$ é uma abreviatura de $caixa\ de\ substituição$. A caixa-s é uma função não linear que substitui um valor de n bits por um valor de m bits. n e m, são números inteiros maiores do que zero. No caso do AES, $n = m = 8$ bits e a caixa-s é bijetora.

3.2.1 Corpos Finitos

Um corpo é um anel comutativo (com elemento neutro) no qual todos os elementos não nulos possuem um inverso multiplicativo [36]. Neste trabalho nos concentramos no corpo finito GF(2^8) utilizado no AES e em cifras correlatas [7, 8]. Para o AES, GF(2^8) é representado como GF(2)/(m(x)), onde $m(x) = x^8+x^4+x^3+x+1$ é um polinômio irredutível sobre GF(2). Assumimos uma representação polinomial para cada elemento $a \in$ GF(2^8) no AES. Então: $a = (a_7, a_6, \ldots, a_1, a_0) = \sum_{i=0}^{7} a_i.x^i$, com $a_i \in GF(2)$ para $0 \leq i \leq 7$. Uma representação compacta de cada elemento $x \in$ GF(2^8) utiliza dígitos hexadecimais (representado com o sub-índice x), expressando os coeficientes da representação polinomial. Por exemplo, $x^7+x^5+x^3+x^2+1 = AD_x$, e $m(x) = 11B_x$.

Adição em GF(2^8) corresponde à operação ou-exclusivo, pois GF(2^8) tem característica dois. Multiplicação em GF(2^8) consiste numa multiplicação de polinômios módulo $m(x)$ [19]

3.2.2 Códigos Corretores de Erros

Códigos corretores de erros já foram sugeridos para o projeto de algoritmos de chave pública por McEliece [34]. A utilização de códigos corretores de erros, em algoritmos de chave secreta foi sugerida por Vaudenay em [50].

A distância Hamming entre dois vetores (ou palavras código) de um espaço vetorial n-dimensional $(GF(2^p))^n$ é o número de posições (entre n posições possíveis) nas quais os dois vetores diferem. O peso Hamming de um vetor (ou palavra código) $u \in (GF(2^p))^n$ é a distância Hamming entre u e o vetor nulo em $(GF(2^p))^n$, ou seja, o número de posições não nulos em u.

Um código linear $[n, k, d]$ sobre GF(2^p) é um subespaço k-dimensional do espaço vetorial $(GF(2^p))^n$, onde a distância Hamming entre dois n-elementos distintos é pelo menos d, e d é o maior número com essa propriedade. Uma matriz geradora G para um $[n, k, d]$-código linear C é uma matriz $k \times n$ cujas linhas formam uma base para C. Códigos $[n, k, d]$ lineares obedecem o limite de Singleton, $d \leq n - k + 1$. Um código que satisfaz estritamente o limite de Singleton, ou seja, $d = n - k + 1$, é chamado Maximum Distance Separable ou código MDS. Alternativamente, um $[n, k, d]$-código corretor de erros com matriz geradora $G = [I_{k\times k}|A]$, onde $I_{k\times k}$ é a matriz identidade $k \times k$, e A é a matriz $k \times (n - k)$, é MDS, somente e somente se cada submatriz quadrada formada por i linhas e i colunas, $1 \leq i \leq \min\{k, n - k\}$, de A for não singular. [35]

Matrizes MDS tem sido um componente fundamental no projeto de cifras de bloco como as SHARK [46], Square [17] e Rijndael [40] para garantir difusão rápida e efetiva em um pequeno número de rodadas. Uma maneira de se obter matrizes MDS é a utilização de matrizes circulantes, onde cada linha é uma instância rotacionada (por uma única unidade) das linhas da vizinhança (em uma mesma direção). Por exemplo, uma matriz circulante 4×4 foi utilizada na cifra de bloco AES [40]. Entretanto esta matriz não é involutória:

$$\begin{bmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{bmatrix} \cdot \begin{bmatrix} 02_x & 03_x & 01_x & 01_x \\ 01_x & 02_x & 03_x & 01_x \\ 01_x & 01_x & 02_x & 03_x \\ 03_x & 01_x & 01_x & 02_x \end{bmatrix} = \begin{bmatrix} 05_x & 00_x & 04_x & 00_x \\ 00_x & 05_x & 00_x & 04_x \\ 04_x & 00_x & 05_x & 00_x \\ 00_x & 04_x & 00_x & 05_x \end{bmatrix}.$$

O fato da matriz de difusão utilizada no AES não ser involutória não é devido a seus elementos. Considerando uma matriz circulante arbitrária 4×4 (com linhas rotacionadas para a direita imitando a matriz de MixColumns),

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{bmatrix}.$$

Se A fosse involutória, $A \cdot A = I_{4\times4}$, onde $I_{4\times4}$ corresponde a matriz identidade

4×4. Esta equação implica em duas restrições (onde $+$ é ou-exclusivo):

$$a_0^2 + a_2^2 = 1, \tag{3.1}$$

e

$$a_1^2 + a_3^2 = 0. \tag{3.2}$$

Se A fosse MDS, então o deteminante a seguir seria não nulo:

$$\begin{vmatrix} a_1 & a_3 \\ a_3 & a_1 \end{vmatrix} = a_1^2 + a_3^2 \neq 0. \tag{3.3}$$

A restrição (3.3) contradiz a (3.2). Resultado similar seria obtido se as linhas fossem rotacionadas para a direita. Portanto, concluimos que matrizes circulantes não podem ser ao mesmo tempo MDS e involutórias. Interpretação análoga se aplica a matriz circulantes maiores. Por essa razão, de agora em diante, consideraremos outras técnicas de construção de matrizes MDS.

Em [29], Junod e Vaudenay sugeriram algumas heurísticas para construir matrizes MDS com baixo custo de implementação. O objetivo deles foi projetar matrizes com um grande número de elementos iguais a 1 (e outros elementos de baixo peso Hamming), para minimizar o custo de implementação. Eles alegam que sua construção resulta em matrizes ótimas em termos de menor número de XOR, menor número acesso a tabela de dados e menor número de variáveis temporárias. No entanto as

matrizes citadas em [29] não são involutórias.

Outra técnica de construção de matrizes involutórias envolvem as chamadas matrizes de Cauchy [53, 13] utilizadas nas cifras de bloco Khazad e Anubis [7]. Nestas cifras, os elementos da matriz são cuidadosamente selecionados para minimizar o número de operações primitivas como ou-exclusivo, acesso a tabela de dados, e chamadas a xtime [19]. Neste trabalho procuramos por matrizes MDS involutórias maiores, cujos elementos possuam baixo peso Hamming. Em particular procuramos por matrizes 16×16, que provem completa difusão em uma única rodada. (Fig. 3.1).

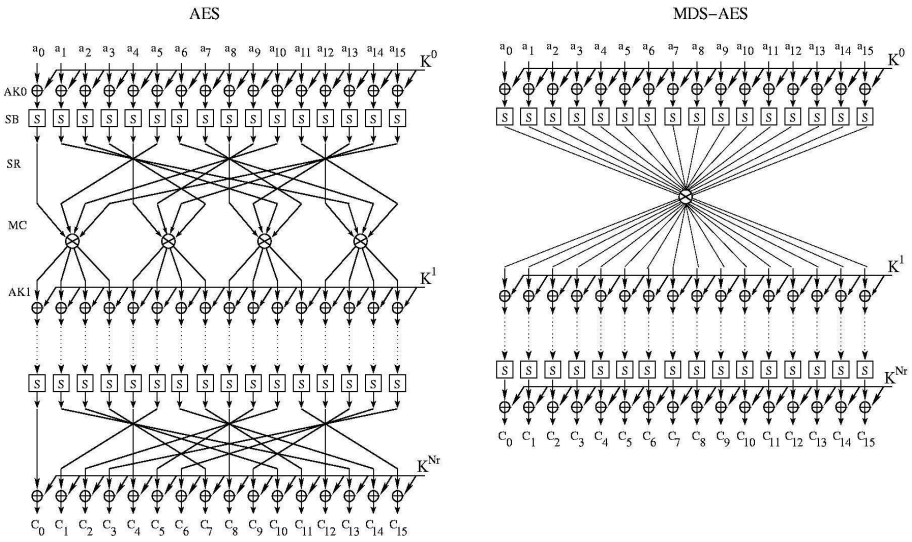

FIGURA 3.1 – Gráfico computacional do AES e MDS-AES. Fonte: [38]

Definição 1: [53] Seja $x_1, x_2, \ldots, x_m$, e $y_1, y_2 \ldots, y_n$ elementos no corpo $\mathbb{F}$, tal que

(1) $x_1, \ldots, x_m$ são distintos,

(2) $y_1, \ldots, y_n$ são distintos, e

(3) $x_i + y_j \neq 0$ para $1 \leq i \leq m$, $1 \leq j \leq n$.

Uma matriz de Cauchy $m \times n$ sobre um corpo finito $\mathbb{F}$ tem elementos $c_{i,j} = \frac{1}{x_i + y_j}$. O determinante de uma matriz de Cauchy é $\frac{\prod_{i<j}(x_i - x_j)(y_i - y_j)}{\prod_{ij}(x_i + y_j)}$. Então, por definição uma matriz quadrada de Cauchy é não singular.

Matrizes de Cauchy quadradas são unitárias ($A^{-1} = A^T$), e simétricas ($A = A^T$, onde A^T indica a matriz transposta A). Estas propriedades garantem que a matriz de Cauchy é involutória.

Neste trabalho, procuramos matrizes que satisfazem várias restrições simultaneamente:

(1) ser MDS,

(2) ser involutória,

(3) ser 16×16,

(4) cada elemento da matriz, em GF(2^8), deve possuir baixo peso Hamming,

(5) os bits menos significativos em cada elemento da matriz deve estar preferevelmente nas posições menos significativas.

A matriz MDS 16x16 involutória encontrada, substitui as camadas SR e MC da cifra original gerando um nova cifra. A nova cifra foi chamada MDS-AES (Fig. 3.1) Uma rodada completa do MDS-AES consiste em uma camada SB, seguida da nova

matriz MDS, e da camada AK_i. A última rodada consiste na camada SB seguida por AK_{Nr} (Fig. 3.1)

O critério (1) garante rápida difusão em um pequeno número de rodadas. A restrição (2) objetiva melhorar a difusão para o ciframento e deciframento.A restrição (3) é conseqüência do tamanho do bloco do AES 16 bytes. As últimas duas restrições tem por objetivo alta performance (em implementações de hardware e software). O peso Hamming de cada elemento da matriz influe diretamente no número de operações xor e xtime. Os bits mais significativos em cada elemento da matriz afetam o número de chamadas a xtime [19], o que significa o mesmo que multiplicação por 02_X em GF(2^8). Entretanto, devido ao tamanho destas matrizes, mais operações primitivas serão necessárias do que no AES. Por exemplo, a matriz 4×4 do AES consome dois xtime e quatro xors por linha de matriz, ou 8 xtime e 16 xors por matriz MC, ou 32 xtime e 64 xors por rodada. Portanto o preço por uma difusão mais rápida é uma menor performance. Porém, do ponto dde vista da segurança as vantagens do MDS-AES são significativas (Vide seção 3.4).

3.3 Uma Nova Matriz MDS Involutória

Estivemos pesquisando grandes matrizes MDS involutórias para susbtituir as camadas SR e MC de cada rodada do AES. Nossa técnica de pesquisa seguiu a estrutura das matrizes de Cauchy (Seção 3.2.2), onde todos os elementos em uma linha são

distintos. Este método de construção por si só garante que a matriz resultante é MDS e involutória.

Nosso algoritmo de pesquisa somente precisa selecionar os elementos da primeira linha da matriz MDS 16×16, pois a construção da matriz de Cauchy depende desta linha. Uma vez que esta linha foi determinada, as demais linhas são simples permutações da primeira linha. Nossa primeira escolha para um elemento nesta linha é o 01_X, porque é o menor elemento não nulo em GF(2^8). Mais tarde os elementos são selecionados em ordem crescente, tal que

(i) quaisquer dois elementos são distintos;

(ii) os pesos Hamming dos elementos de $M_{16\times16}$são os menores possíveis (no máximo 4);

(iii) o bit de maior significância em cada elemento está preferivelmente na posição de menor significância.

(iv) todos os elementos não são nulos.

Se os valores não obedecem as restrições acima o algoritmo procura o valor maior mais próximo e aplica o mesmo procedimento novamente até que o décimo quinto elemento seja determinado. O décimo sexto elemento (mais a direita) na primeira linha da matriz é o resultado de uma operação de ou exclusivo de todos os quinze valores anteriores e 1. Este último valor também deve ser diferente dos elementos anteriores.

Para automatizar este processo e propiciar diferentes restrições para o peso Hamming e número de operações elementares xtime e xor desenvolvemos um software na linguagem Java, chamado Gerador de Matrizes de Cauchy (GMC). Neste software os valores para o número de elementos na matriz assim como os valores máximos para o peso Hamming, xtime e xor, para cada elemento selecionado são parametrizáveis. Assim a partir da escolha dos valores o software determinará a escolha da primeira linha de elementos da matriz. A geração da primeira linha segue as restrições das matrizes de Cauchy e o software também faz simulação do custo operacional com cada um dos elementos selecionados a partir de 1 (menor peso Hamming), substituindo um elemento de maior custo por um de menor custo, caso este exista. O cálculo do último elemento da linha requer uma rotina especial para garantir que o resultado dos xor com todos os demais elementos da linha seja 1. Para isso o software mantém um registro do cálculo acumulado de xors e caso o resultado com o último elemento não seja 1 uma rotina descarta o último e o penúltimo elemento e seleciona outros candidatos. Como esta restrição é prioritária para que a matriz seja MDS involutória, o último elemento pode não cumprir a restrição para peso Hamming, daí todos os elementos possuírem o menor peso Hamming exceto o último elemento. O software gera os demais elementos da matriz conforme as regras de Cauchy. Essas permutações 2 a 2 restrigem o número de elementos da matriz a ser uma potência de 2.

A melhor matriz encontrada segundo estas restrições é a seguinte:

$$M_{16\times16} = \begin{bmatrix}
01_x & 03_x & 04_x & 05_x & 06_x & 07_x & 08_x & 09_x & 0a_x & 0b_x & 0c_x & 0d_x & 0e_x & 10_x & 02_x & 1e_x \\
03_x & 01_x & 05_x & 04_x & 07_x & 06_x & 09_x & 08_x & 0b_x & 0a_x & 0d_x & 0c_x & 10_x & 0e_x & 1e_x & 02_x \\
04_x & 05_x & 01_x & 03_x & 08_x & 09_x & 06_x & 07_x & 0c_x & 0d_x & 0a_x & 0b_x & 02_x & 1e_x & 0e_x & 10_x \\
05_x & 04_x & 03_x & 01_x & 09_x & 08_x & 07_x & 06_x & 0d_x & 0c_x & 0b_x & 0a_x & 1e_x & 02_x & 10_x & 0e_x \\
06_x & 07_x & 08_x & 09_x & 01_x & 03_x & 04_x & 05_x & 0e_x & 10_x & 02_x & 1e_x & 0a_x & 0b_x & 0c_x & 0d_x \\
07_x & 06_x & 09_x & 08_x & 03_x & 01_x & 05_x & 04_x & 10_x & 0e_x & 1e_x & 02_x & 0b_x & 0a_x & 0d_x & 0c_x \\
08_x & 09_x & 06_x & 07_x & 04_x & 05_x & 01_x & 03_x & 02_x & 1e_x & 0e_x & 10_x & 0c_x & 0d_x & 0a_x & 0b_x \\
09_x & 08_x & 07_x & 06_x & 05_x & 04_x & 03_x & 01_x & 1e_x & 02_x & 10_x & 0e_x & 0d_x & 0c_x & 0b_x & 0a_x \\
0a_x & 0b_x & 0c_x & 0d_x & 0e_x & 10_x & 02_x & 1e_x & 01_x & 03_x & 04_x & 05_x & 06_x & 07_x & 08_x & 09_x \\
0b_x & 0a_x & 0d_x & 0c_x & 10_x & 0e_x & 1e_x & 02_x & 03_x & 01_x & 05_x & 04_x & 07_x & 06_x & 09_x & 08_x \\
0c_x & 0d_x & 0a_x & 0b_x & 02_x & 1e_x & 0e_x & 10_x & 04_x & 05_x & 01_x & 03_x & 08_x & 09_x & 06_x & 07_x \\
0d_x & 0c_x & 0b_x & 0a_x & 1e_x & 02_x & 10_x & 0e_x & 05_x & 04_x & 03_x & 01_x & 09_x & 08_x & 07_x & 06_x \\
0e_x & 10_x & 02_x & 1e_x & 0a_x & 0b_x & 0c_x & 0d_x & 06_x & 07_x & 08_x & 09_x & 01_x & 03_x & 04_x & 05_x \\
10_x & 0e_x & 1e_x & 02_x & 0b_x & 0a_x & 0d_x & 0c_x & 07_x & 06_x & 09_x & 08_x & 03_x & 01_x & 05_x & 04_x \\
02_x & 1e_x & 0e_x & 10_x & 0c_x & 0d_x & 0a_x & 0b_x & 08_x & 09_x & 06_x & 07_x & 04_x & 05_x & 01_x & 03_x \\
1e_x & 02_x & 10_x & 0e_x & 0d_x & 0c_x & 0b_x & 0a_x & 09_x & 08_x & 07_x & 06_x & 05_x & 04_x & 03_x & 01_x
\end{bmatrix}.$$

Uma comparação entre a performance da matriz $M_{16\times16}$ e da matriz do AES, simplesmente conta o número de operações elementárias por rodada, como os xors, número de chamadas a xtime e número de consultas a tabelas de dados (se xtime

TABELA 3.1 – Comparação de performance em software (estimativa).

Cifra	# por operação **por rodada**		
	# xtime	# xor	total
AES	32	64	96
MDS-AES	688	272	960

TABELA 3.2 – Comparação entre o DES e o AES.

Cifra	DES	AES
Tamanho da Bloco	64 bits	128 bits
Tamanho da Chave	56 bits	128,192 ou 256 bits
Número Rodadas	16	10,12,14
Autores	IBM	J. Daemen e V. Rijmen
Lançamento Oficial	1977	1998
Projetada para	hardware	hardware e software
Estrutura	Feistel	SPN
Tempo de vida	23 anos (1977-2000)	30 a 50 anos (esperado)
Referência	[39]	[19, 18]

for armazenada em uma tabela). Para simplificar, assumimos que cada uma destas operações requer um único ciclo de máquina. Portanto, uma única rodada de MDS-AES custa o mesmo número de operações elementares que toda a computação da matriz do AES em 9 rodadas (com chave de 128 bits).

3.4 Análise de Segurança

O AES tem sido extensivamente analisado desde 1997. Entretanto os resultados conhecidos se aplicam somente as variantes com número reduzido de rodadas: análise diferencial (DC) e linear (LC) [14], ataque quadrado [18, 23], diferencial impossível

(ID) [9, 44, 43], colisão [26], ataque bumerangue [11] e assim por diante.

Uma característica comum explorada por todos estes ataques em versões reduzidas do AES é a difusão completa em 2 rodadas pela combinação das camadas SR e MC. MC opera sobre palavras de 32 bits. Isso significa que nem todo bit de saída depende de todos os bits da entrada (texto original e chave) depois de uma única rodada. No AES, por exemplo, todos os bits do texto original são difundidos completamente depois de duas rodadas [18, p.29]. Os bits da chave, entretanto, são difundidos completamente após um número de rodadas que depende do tamanho da chave. Para chaves de 128 bits, a difusão completa é obtida depois de duas rodadas, e é necessária uma rodada adicional para cada 32 bits de comprimento adicional da chave [18, p.29]. Esta decisão de projeto (difusão incompleta em uma única rodada) foi possivelmente baseada em um compromisso entre segurança e performance.

Se a difusão completa fosse atingida em uma única rodada do AES, então todos os ataques conhecidos contra o AES teriam muito menor impacto. Consequentemente, com difusão completa, o número de rodadas de uma cifra pode ser reduzido, compensando a perda de performance relacionada à matriz $M_{16\times16}$.

Como exemplo, invariantes diferenciais cobrindo 4 rodadas de AES, contém ao mínimo 25 caixas-s ativas [15], conforme [18] (linhas pontilhadas na Fig. 3.2(a), onde uma diferença não nula é representada por δ). A matriz MDS (3.3) faz com que o invariante diferencial contenha no mínimo 33 caixas-s ativas durante 3 rodadas, (Fig. 3.2(b)): dezesseis bytes de diferenças ativas na primeira rodada, um byte ativo

na segunda rodada e dezesseis bytes ativos na terceira rodada. Pela contagem do número de caixas-s, a correspondente probabilidade do invariante diferencial cai de $(2^{-6})^{25}$ (no AES) para $(2^{-6})^{33}$ (no MDS-AES). Este ataque implica que pelo menos 4 rodadas são necessárias para o MDS-AES. Razões similares se aplicam aos ataques lineares (convencionais) [33].

Outro importante ataque a ser considerado no MDS-AES é a técnica $multiset$ [17], pois este é o mais efetivo ataque conhecido contra versões de rodadas reduzidas do AES. Para maiores detalhes vide [38].

Um nítido aumento na segurança pode ser observado em relação ao ataque de colisão de Gilbert e Minier [26]. Este ataque é aplicavél a 7 rodadas do AES (exigindo todos os 2^{128} textos legíveis) e depende da difusão incompleta um uma rodada do AES. Portanto, este ataque, não se aplica ao MDS-AES, pois a difusão é completa em uma única rodada.

Considerando o ataque diferencial impossível (ID) [9, 44], qualquer diferencial truncado (com probabilidade um) envolvendo duas rodadas deve envolver no mínimo 17 caixas-s ativas. Utilizando a técnica miss-in-the-midle (MITM) [9] podemos concluir que qualquer par de diferenciais E_0 e E_1 para um ataque ID deve cobrir três rodadas, caso contrário, não haverá contradição entre E_0 e E_1. Isso significa que um invariante ID pode cobrir até três rodadas do MDS-AES (comparado com quatro no AES). Sendo assim, se quisermos aplicar um ataque ID para 4 rodadas do MDS-AES, uma chave completa (128 bits) deverá ser adivinhada (devido a matriz $M_{16\times16}$),

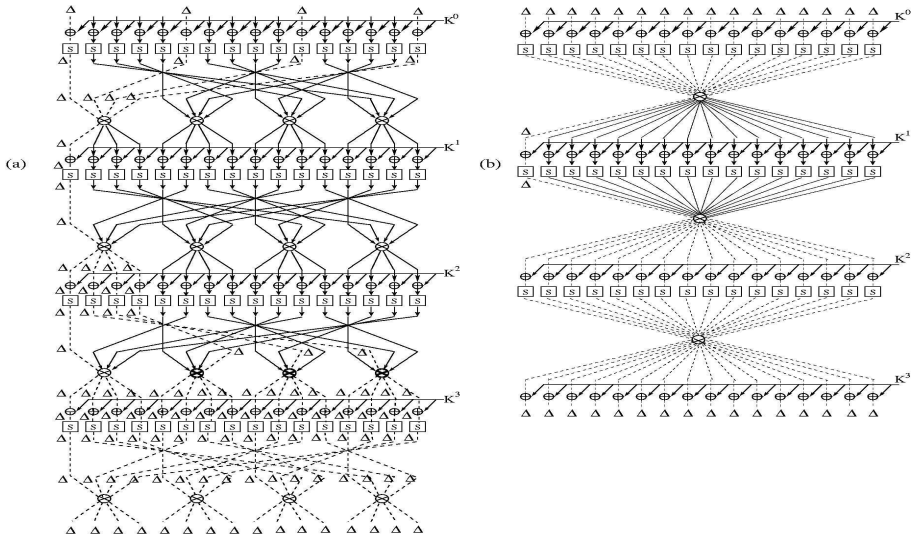

FIGURA 3.2 – (a) Diferencial (linha pontilhada) para o AES, e (b) para MDS-AES. Fonte [38].

tornando o ataque impraticável.

Considerando o ataque bumerangue [10], a construção de diferenciais truncados para o invariante bumerangue possui as mesmas premissas do ataque ID, ou seja, qualquer par de diferenciais truncados para um ataque bumerangue terá vários bytes ativos devido a matriz $M_{16\times16}$. Este fenômeno acontece em ambos os diferenciais E_0 e E_1, independentemente do número de bytes não nulos. Inclusive, a expectativa é que as rodadas para invariantes bumerangue para o MDS-AES sejam mais curtas (duas ou três rodadas) que as mesmos para o AES (cinco rodadas), e não relevantes se comparadas ao ataque *multiset*.

Baseada na análise de segurança descrita anteriormente, recomenda-se no mínimo cinco rodadas para o MDS-AES.

TABELA 3.3 – Comparação entre ataques ao AES (com poucas rodadas) e o MDS-AES.

Cifra	DC	LC	Multiset	Colisão	ID	Bumerangue
AES	2^{150} CP (4 rodadas)	2^{150} KP (4 rodadas)	2^{9} CP (4 rodadas)	$\approx 2^{128}$ CP (7 rodadas)	$2^{29.5}$ CP (5 rodadas)	2^{39} CPACC (5 rodadas)
MDS-AES	2^{198} CP (3 rodadas)	2^{198} KP (3 rodadas)	2^{9} CP (3 rodadas)	-	-	-

CP: Mensagem escolhida; KP: Mensagem conhecida.

3.5 Conclusão

Este trabalho apresenta uma nova matriz MDS 16×16 para a cifra AES. A nova cifra foi chamada MDS-AES. A nova matriz $M_{16\times16}$ substitui as camadas SR e MC em cada rodada. Além disso, a natureza involutória da nova matriz permite rápida difusão, tanto para a operação de ciframento quanto para o deciframento. A nova matriz foi encontrada após uma busca heurística com auxílio de um software por nós desenvolvido satisfazendo todas as seguintes propriedades: (i) MDS, (ii) involutória, (iii) 16×16, (iv) ter elementos com baixo peso Hamming e (v) o bit mais significativo em cada elemento da matriz está na posição menos significativa.

A construção do MDS-AES mostra boa resistência contra ataques diferenciais, lineares, *multiset*, colisão, diferencial impossível e bumerangue. Dados os ataques na Tabela 3.3, concluimos que (i) a resistência do MDS-AES é maior do que a do AES; (ii) o melhor ataque (significando alto número de rodadas e menor custo computacional) no AES com poucas rodadas parece ser o ataque *multiset*.

Nota-se que a difusão completa por si só, não previne outros ataques como $slide$ ou $slide$ avançado [12], nem ataques mod-n [31]; porém o escalonamento da chave do AES já evita que as rodadas sejam idênticas, o que é crucial para estes ataques. Nós assuminos que o MDS-AES utiliza o mesmo escalonamento de chaves do AES. Difusão completa sozinha também não previne ataques algébricos [16]; uma contra-medida sugerida é a utilização de uma caixa-s com uma representação algébrica mais elaborada em GF(2^8), por exemplo a caixa-s do Skipjack [41], cuja representação algébrica não é tão simples quanto a do AES, e que parece aleatória. Adicionalmente nessa caixa-s não parece ser derivada do mapeamento inverso em GF(2^8). Além disso, a caixa-s do Skipjack possuir características diferenciais e lineares semelhantes a da caixa-s do AES.

Deixamos como um problema em aberto se outras matrizes MDS, involutórias, 16×16, podem ser encontradas com um peso Hamming mais baixo que $M_{16\times16}$, e por conseqüência menor custo de implememtação.

A única inconveniência do MDS-AES é a baixa performance devido ao grande número de operações primitivas (xor, xtime, acesso a tabela de dados, variáveis temporárias) complicada pelo grande número de componentes da matriz, fazendo-a muito mais lenta do que o AES. Este fato indica que o novo projeto é mais de interesse teórico do que prático.

Um tópico para pesquisa futura é a determinação de matrizes MDS involutórias (não do tipo de Cauchy) para o Rijndael-160, Rijndael-192, Rijndael-224. Nós não

podemos derivar estas matrizes pois suas dimensões não são potências de 2 (para se ajustar ao tamanho do bloco destas cifras).

Anexo A - Programa Gerador de Matrizes de Cauchy *GMC*

```
    package gmc;
    import java.awt.Toolkit;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import java.awt.Dimension;
    /**
<p>Title: GMC</p>
<p>Description: Gerador de Matrizes de Cauchy</p>
<p>Copyright: Copyright (c) 2006</p>
<p>Company: Unisantos</p>
@author Elcio Abrahão e Jorge Nakaraha Jr.
@version 1.3
/
public class Application1 {
boolean packFrame = false;
    /**
Construct and show the application.
/
public Application1() {
Frame1 frame = new Frame1();
// Validate frames that have preset sizes
// Pack frames that have useful preferred size info, e.g. from their layout
if (packFrame) {
frame.pack();
} else {
frame.validate();
}
```

```
    // Center the window
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
Dimension frameSize = frame.getSize();
if (frameSize.height > screenSize.height) {
frameSize.height = screenSize.height;
}
if (frameSize.width > screenSize.width) {
frameSize.width = screenSize.width;
}
frame.setLocation((screenSize.width - frameSize.width) / 2,
(screenSize.height - frameSize.height) / 2);
frame.setVisible(true);
try {
jbInit();
} catch (Exception ex) {
ex.printStackTrace();
}
}
    /**
Application entry point.
@param args String[]
/
public static void main(String[] args) {
SwingUtilities.invokeLater(new Runnable() {
public void run() {
try {
UIManager.setLookAndFeel(UIManager.
getSystemLookAndFeelClassName());
} catch (Exception exception) {
exception.printStackTrace();
}
    new Application1();
}
});
}
    private void jbInit() throws Exception {
}
}
    ————————————————————-
    package gmc;
    /**
```

```
<p>Title: GMC</p>
<p>Description: Gerador de Matrizes de Cauchy</p>
<p>Copyright: Copyright (c) 2006</p>
<p>Company: Unisantos</p>
@author Elcio Abrahão e Jorge Nakaraha Jr.
@version 1.3
/
public class Elemento {
String decimal = '';
String hexadecimal = '';
String binario = '';
String valorXor = '';
int xorInt = 0;
int pesoHamming = 0;
int xTimeCount = 0;
int xorCount = 0;
    /** Creates a new instance of Elemento */
public Elemento(double a,String b,String c) {
this.decimal = ''+a;
this.hexadecimal = b;
this.binario = completaZero(c);
this.pesoHamming = calculaPeso(this.binario);
this.xorCount = calculaXor(this.pesoHamming);
this.xTimeCount = calculaXTime(this.binario);
    }
public Elemento(){}
    public void setXTimeCount(int xTimeCount){
this.xTimeCount = xTimeCount;
}
    public void setXorCount(int xorCount){
this.xorCount = xorCount;
}
public int getXorCount(){
return this.xorCount;
}
public int getXTimeCount(){
return this.xTimeCount;
}
    public void setDecimal(String decimal){
this.decimal = decimal.trim();
}
public void setHexadecimal(String hexadecimal){
this.hexadecimal = hexadecimal;
}
public void setBinario(String binario){
this.binario = binario.trim();
}
```

```
public void setValorXor(String xor){
this.valorXor = xor.trim();
this.xorInt =(int)Integer.parseInt(xor);
}
public void setValorHamming(int hamming){
this.pesoHamming=hamming;
}
public int getValorHamming(){
return this.pesoHamming;
}
public String getDecimal(){
return this.decimal;
}
public int getDecimalInt(){
//System.out.println("Valor dde decima:"+this.decimal+"<–");
Double dd =new Double(this.decimal);
return dd.intValue();
}
public String getHexadecimal(){
return this.hexadecimal;
}
public String getBinario(){
return this.binario;
}
public String getValorXor(){
return this.valorXor;
}
public int getxorInt(){
return this.xorInt;
}
public int calculaPeso(String bin){
    int comp = bin.length();
char caracter;
int contador = 0;
for(int i = 0;i<comp;i++){
    caracter = bin.charAt(i);
if(caracter == '1'){
contador++;
}
    }
return contador;
}
public int calculaXor(int pesoHamming){
    return (pesoHamming - 1);
}
public int calculaXTime(String bin){
    int comp = bin.length();
```

```
System.out.println("Binario: "+bin);
int[] ArrayInt = new int[8];
char caracter;
int contador = 0;
System.out.println(" ");
    for(int i = 7;i>=0;i–){
    caracter = bin.charAt(i);
if(caracter == '1'){
ArrayInt[7-i]=7-i;
}else{
ArrayInt[7-i]=0;
}
System.out.print(", "+ArrayInt[i]);
}
System.out.println(" ");
    int numeroXtimes = 0;
for(int i=0;i<7;i++){
if(ArrayInt[i+1] > ArrayInt[i]){
numeroXtimes = numeroXtimes + (ArrayInt[i+1] - ArrayInt[i]);
}else{
ArrayInt[i+1]=numeroXtimes;
}
}
System.out.println("Numero de xTimes:"+numeroXtimes);
return numeroXtimes;
}
public String completaZero(String c){
    String comeco = "";
if(c.length()<8){
for(int i = 0;i<(8-c.length());i++){
comeco +="0";
}
    }
return comeco+c;
}
    }
    ———————————————————————————
    package gmc;
    import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JLabel;
```

```
import com.borland.jbcl.layout.XYLayout;
import com.borland.jbcl.layout.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.util.Vector;
import javax.swing.border.EtchedBorder;
import javax.swing.border.Border;

    /**
<p>Title: GMC</p>

<p>Description: Gerador de Matrizes de Cauchy</p>

<p>Copyright: Copyright (c) 2006</p>

<p>Company: Unisantos</p>

@author Elcio Abrahão e Jorge Nakaraha Jr.
@version 1.3
/
public class Frame1 extends JFrame {

    private int tamanhoMatriz = 2;
private int maximoHamming = 1;
private int maxXTimes= 1;
Vector dif = new Vector(256);
Vector vec = new Vector(256);
Elemento[][] eleMatriz = new Elemento[256][256];

    JPanel contentPane;
JMenuBar jMenuBar1 = new JMenuBar();
JMenu jMenuFile = new JMenu();
JMenuItem jMenuFileExit = new JMenuItem();
JMenu jMenuHelp = new JMenu();
JMenuItem jMenuHelpAbout = new JMenuItem();
XYLayout xYLayout1 = new XYLayout();
JPanel jPanel1 = new JPanel();
TitledBorder titledBorder1 = new TitledBorder("");
XYLayout xYLayout3 = new XYLayout();
JCheckBox jCheckBox1 = new JCheckBox();
JCheckBox jCheckBox2 = new JCheckBox();
JLabel jLabel1 = new JLabel();
JComboBox jComboBox1 = new JComboBox();
JLabel jLabel2 = new JLabel();
JComboBox jComboBox2 = new JComboBox();
JButton jButton1 = new JButton();
JButton jButton2 = new JButton();
JComboBox jComboBox3 = new JComboBox();
JLabel jLabel3 = new JLabel();
JButton jButton3 = new JButton();
JMenu jMenu1 = new JMenu();
JMenuItem jMenuItem1 = new JMenuItem();
JMenuItem jMenuItem2 = new JMenuItem();
```

```
JMenuItem jMenuItem3 = new JMenuItem();
JMenuBar jMenuBar2 = new JMenuBar();
JLabel jLabel4 = new JLabel();
JProgressBar jProgressBar1 = new JProgressBar();
JTextArea jTextArea1 = new JTextArea();
JScrollPane scrollableTextArea = new JScrollPane(jTextArea1);
//scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLL-
BAR_ALWAYS);
//scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_-
ALWAYS);
    Border border1 = BorderFactory.createEtchedBorder(EtchedBorder.RAISED,
Color.white, new Color(178, 178, 178));
public Frame1() {
try {
setDefaultCloseOperation(EXIT_ON_CLOSE);
jbInit();
} catch (Exception exception) {
exception.printStackTrace();
}
}
    /**
Component initialization.
@throws java.lang.Exception
/
private void jbInit() throws Exception {
contentPane = (JPanel) getContentPane();
contentPane.setLayout(xYLayout1);
setSize(new Dimension(500, 400));
setTitle("Gerador de Matrizes de Cauchy - GMC");
jMenuFile.setText("Arquivo");
jMenuFileExit.setText("Sair");
jMenuFileExit.addActionListener(new Frame1_jMenuFileExit_ActionAdapter(this));
jMenuHelp.setText("Ajuda");
jMenuHelpAbout.setText("Sobre");
jMenuHelpAbout.addActionListener(new
Frame1_jMenuHelpAbout_ActionAdapter(this));
jPanel1.setBorder(BorderFactory.createEtchedBorder());
jPanel1.setLayout(xYLayout3);
jCheckBox1.setToolTipText("");
jCheckBox1.setText("Otimizar Custos 1");
jCheckBox2.setToolTipText("");
jCheckBox2.setText("Otimizar Custos 2");
jLabel1.setText("Max Peso Hammig");
jLabel2.setText("Max X Times");
jButton1.setText("Nova");
jButton1.addActionListener(new Frame1_jButton1_actionAdapter(this));
```

```
jButton2.setText("Gerar");
jButton2.addActionListener(new Frame1_jButton2_actionAdapter(this));
jLabel3.setText("Tamanho da Matriz");
jButton3.setText("Salvar");
jButton3.addActionListener(new Frame1_jButton3_actionAdapter(this));
jMenu1.setText("Executar");
jMenuItem1.setText("Nova");
jMenuItem2.setText("Gerar");
jMenuItem3.setText("Salvar");
jComboBox3.addActionListener(new Frame1_jComboBox3_actionAdapter(this));
jComboBox1.addActionListener(new Frame1_jComboBox1_actionAdapter(this));
jComboBox2.addActionListener(new Frame1_jComboBox2_actionAdapter(this));
jLabel4.setText("Progresso");
jTextArea1.setBorder(border1);
jMenuBar1.add(jMenuFile);
jMenuFile.add(jMenuFileExit);
jMenuBar1.add(jMenuHelp);
jMenuBar1.add(jMenu1);
jMenuHelp.add(jMenuHelpAbout);
setJMenuBar(jMenuBar1);
contentPane.add(jPanel1, new XYConstraints(7, 9, 485, 134));
contentPane.add(jLabel4, new XYConstraints(9, 374, -1, -1));
contentPane.add(jProgressBar1, new XYConstraints(81, 372, 409, 17));
jPanel1.add(jCheckBox1, new XYConstraints(6, 104, -1, -1));
jPanel1.add(jCheckBox2, new XYConstraints(164, 104, -1, -1));
jPanel1.add(jComboBox3, new XYConstraints(150, 2, 125, -1));
jPanel1.add(jLabel2, new XYConstraints(9, 70, -1, -1));
jPanel1.add(jLabel1, new XYConstraints(9, 36, -1, -1));
jPanel1.add(jLabel3, new XYConstraints(8, 7, -1, -1));
jPanel1.add(jButton3, new XYConstraints(372, 83, -1, -1));
jPanel1.add(jButton2, new XYConstraints(372, 46, -1, -1));
jPanel1.add(jButton1, new XYConstraints(372, 8, -1, -1));
jPanel1.add(jComboBox1, new XYConstraints(150, 32, 125, -1));
jPanel1.add(jComboBox2, new XYConstraints(150, 64, 126, -1));
jMenu1.add(jMenuItem1);
jMenu1.add(jMenuItem2);
jMenu1.add(jMenuItem3);
contentPane.add(scrollableTextArea, new XYConstraints(9, 157, 483, 203));
jTextArea1.setAutoscrolls(true);
jComboBox3.addItem("2 x 2");
jComboBox3.addItem("4 x 4");
jComboBox3.addItem("8 x 8");
jComboBox3.addItem("16 x 16");
jComboBox3.addItem("32 x 32");
jComboBox3.addItem("64 x 64");
jComboBox1.addItem("1");
jComboBox1.addItem("2");
```

```
jComboBox1.addItem("3");
jComboBox1.addItem("4");
jComboBox1.addItem("5");
jComboBox1.addItem("6");
jComboBox1.addItem("7");
jComboBox1.addItem("8");
jComboBox2.addItem("1");
jComboBox2.addItem("2");
jComboBox2.addItem("3");
jComboBox2.addItem("4");
jComboBox2.addItem("5");
jComboBox2.addItem("6");
jComboBox2.addItem("7");
jComboBox2.addItem("8");
jComboBox2.addItem("9");
jComboBox2.addItem("10");
    }
    /**
File | Exit action performed.
@param actionEvent ActionEvent
/
void jMenuFileExit_actionPerformed(ActionEvent actionEvent) {
System.exit(0);
}
    /**
Help | About action performed.
@param actionEvent ActionEvent
/
void jMenuHelpAbout_actionPerformed(ActionEvent actionEvent) {
Frame1_AboutBox dlg = new Frame1_AboutBox(this);
Dimension dlgSize = dlg.getPreferredSize();
Dimension frmSize = getSize();
Point loc = getLocation();
dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x,
(frmSize.height - dlgSize.height) / 2 + loc.y);
dlg.setModal(true);
dlg.pack();
dlg.show();
}
    public void jButton1_actionPerformed(ActionEvent actionEvent) {
jCheckBox1.setSelected(false);
jCheckBox2.setSelected(false);
jProgressBar1.setValue(0);
jTextArea1.replaceRange(" ",0,100);
}
```

```
    public void jButton2_actionPerformed(ActionEvent actionEvent) {
vec.clear();
dif.clear();
String entra = "";
double dec,dec1 = 0.0;
String hexa,hexa1 = "";
String bin,bin1 = "";
int contador = 1;
String ultimo = "";
jProgressBar1.setValue(0);

    entra = convertToTen("1",10);
dec = Double.valueOf(entra).doubleValue();
hexa = (convertFromTen(dec,16));
bin = (convertFromTen(dec,2));

    Elemento ele = new Elemento(dec,hexa,bin);
ele.setValorXor(ele.getBinario());
vec.add(ele);
System.out.println("Tamanho da Matriz:"+tamanhoMatriz);
geraCabecario();
adicionaLimiteHamming();
int i = 2;
boolean continua = true;
while(continua){
if(i>256){
//geraErro();
continua=false;
}
//System.out.println("i"+i+"i");
// para elemento atual
entra = convertToTen(""+i,10);
dec = Double.valueOf(entra).doubleValue();
hexa = (convertFromTen(dec,16));
bin = (convertFromTen(dec,2));

    // para elemento atual + 1
entra = convertToTen(""+(i+1),10);
dec1 = Double.valueOf(entra).doubleValue();
hexa1 = (convertFromTen(dec1,16));
bin1 = (convertFromTen(dec1,2));

    Elemento elemento1 = (Elemento)vec.elementAt(contador-1);
//System.out.println("XXX"+elemento1.getBinario()+"XXXX");
Elemento elemento2 = new Elemento(dec,hexa,bin);
elemento2.setValorXor(calculaXor(elemento1.getValorXor(),elemento2.getBinario()));
Elemento elemento3 = new Elemento(dec1,hexa1,bin1);
elemento3.setValorXor(calculaXor(elemento2.getValorXor(),elemento3.getBinario()));

    //System.out.println("xorInt:"+elemento2.getxorInt());
if(elemento3.getxorInt()!=0){
```

```
    //System.out.println("Hamming Objeto:"+elemento3.getValorHamming());
//System.out.println("Hamming Limite:"+maximoHamming);
    if((elemento2.getValorHamming()<=maximoHamming)&&(elemento2.getXTimeCount()<=maxXTim
vec.add(elemento2);
contador++;
}else {
dif.add(elemento2);
}
}else{
dif.add(elemento2);
}
    if(contador==(tamanhoMatriz-1)){
continua = false;
}
i++;
}
if(i<257){
Elemento ele3 = (Elemento) vec.elementAt(contador - 1);
ultimo = calculaXor(ele3.getValorXor(), "1");
entra = convertToTen(ultimo, 2);
dec = Double.valueOf(entra).doubleValue();
hexa = (convertFromTen(dec, 16));
bin = (convertFromTen(dec, 2));
    Elemento ele4 = new Elemento(dec, hexa, bin);
ele4.setValorXor(calculaXor(ele3.getValorXor(), ele4.getBinario()));
vec.add(ele4);
progresso(tamanhoMatriz, tamanhoMatriz * 0.1);
    if (jCheckBox1.isSelected()) {
verificaMatriz();
}
if (jCheckBox2.isSelected()) {
jTextArea1.append("\nElementos para troca: "+dif.size()+"\n\n");
Elemento ell = new Elemento();
jTextArea1.append("lista:\n");
for(int h=0; h<dif.size();h++){
ell = (Elemento)dif.get(h);
jTextArea1.append(ell.getHexadecimal()+" - "+ell.getBinario()+" - "+ell.getValorHamming()+"
- "+ell.getXTimeCount()+"\n");
}
}
confereXor();
    geraMatriz();
    geraCustos();
    mostraMatrizXor(tamanhoMatriz, tamanhoMatriz);
mostraMatrizXTime(tamanhoMatriz, tamanhoMatriz);
```

```
}else{
geraErro();
    }
geraRodape();
    }
    private void geraErro(){
    jTextArea1.append("\n\nNão é possivel encontrar matrizes Cauchy com \n");
jTextArea1.append("\npeso Hamming: "+maximoHamming+ " \n");
jTextArea1.append("\nxtimes: "+maxXTimes+" \n\n");
jTextArea1.append("\nProcessamento terminado.\n\n");
    }
    private void confereXor(){
    int tamVec= vec.size();
Vector vec2 = new Vector();
vec2.add((Elemento)vec.elementAt(0));
String ultimoBinario = "";
    for(int i =0;i<tamVec;i++){
Elemento ele0 = (Elemento)vec.elementAt(i);
ultimoBinario=ele0.getValorXor();
}
    /**
for(int i =0;i<tamVec-1;i++){
Elemento ele0 = (Elemento)vec.elementAt(i);
Elemento ele1 = (Elemento)vec.elementAt(i+1);
ele1.setValorXor(calculaXor(ele0.getBinario(),ele1.getBinario()));
vec2.add(ele1);
jTextArea1.append("i:"+(i+1)+" ");
ultimoBinario=ele1.getValorXor();
jTextArea1.append("Xor:"+ultimoBinario+"\n");
}
    jTextArea1.append("Conferência do XOR do œltimo elemento: "+ultimoBina-
rio+"\n\n");
vec.removeAllElements();
for(int j=0;j<tamVec;j++){
vec.add(vec2.elementAt(j));
}
/
}
    private void verificaMatriz(){
    int tamanhoVec = vec.size();
int tamanhoDif = dif.size();
jTextArea1.append("\nVerificando e corrigindo œltimo elemento: "+tamanhoDif+"
```

```
elementos troc‡veis\n\n");
String ultimo = "";
String entra = "";
double dec = 0.0;
String hexa ="";
String bin ="";
int i = 0;

    Elemento ele = (Elemento)vec.elementAt(tamanhoVec-3);
Elemento ele0 = (Elemento)vec.elementAt(tamanhoVec-2);
Elemento ele1 = (Elemento)vec.elementAt(tamanhoVec-1);

    Elemento ele2 = new Elemento(0.0,"0","0");

    if(ele1.getDecimalInt() <= ele0.getDecimalInt()){

    vec.removeElementAt(tamanhoVec-1);
vec.removeElementAt(tamanhoVec-2);

    ele2 = (Elemento)dif.elementAt(i);
ele2.setValorXor(calculaXor(ele.getValorXor(),ele2.getBinario()));
vec.add(ele2);

    ultimo = calculaXor(ele2.getValorXor(),"1");
entra = convertToTen(ultimo,2);
dec = Double.valueOf(entra).doubleValue();
hexa = (convertFromTen(dec,16));
bin = (convertFromTen(dec,2));

    Elemento ele4 = new Elemento(dec,hexa,bin);
ele4.setValorXor(calculaXor(ele2.getValorXor(),ele4.getBinario()));
vec.add(ele4);

    }
/**
Elemento ele5 = (Elemento)vec.elementAt(tamanhoVec-2);
Elemento ele6 = (Elemento)vec.elementAt(tamanhoVec-1);
if(ele6.getDecimalInt() <= ele5.getDecimalInt()){
verificaMatriz();
}
/
if(vec.size()<(tamanhoMatriz-2)){
vec.add(ele0);
vec.add(ele1);
} else{
mostraPrimeiraLinha();
}
}

    private void mostraPrimeiraLinha(){

    jTextArea1.append("\nPrimeira Linha Corrigida: {");
for(int i=0;i<tamanhoMatriz-1;i++){
```

```
Elemento ele = (Elemento)vec.elementAt(i);
jTextArea1.append(ele.getHexadecimal()+",");
}
Elemento ele = (Elemento)vec.elementAt(tamanhoMatriz-1);
jTextArea1.append(ele.getHexadecimal()+"}\n\n");
}

    private void geraCustos(){

    jTextArea1.append("\nCalculo de Custos\n\n");

    jTextArea1.append("\nBase Comparação (AES):\n\n");

    jTextArea1.append("\nUnid. Xtimes Xor\n");
jTextArea1.append("\nLinha 2 1\n");
jTextArea1.append("\nMatriz 8 4\n");
jTextArea1.append("\n4 MTZ 32 16\n");
jTextArea1.append("\n10 IT 320 160\n");
jTextArea1.append("\n12 IT 384 192\n");
jTextArea1.append("\n14 IT 448 224\n\n");

    int xtimes = 0;
int xors = 0;
for(int i = 0 ;i<tamanhoMatriz;i++){
xtimes = xtimes + eleMatriz[i][0].getXTimeCount();
xors = xors + eleMatriz[i][0].getXorCount();
}
xtimes = xtimes * tamanhoMatriz;
xors = xors * tamanhoMatriz;

    jTextArea1.append("\nEsta!  +"+xtimes+" xTimes e "+xors+"xors - Total de
"+(xtimes+xors)+" operaçes\n");

    }

    private void geraMatriz(){

    for(int i =0;i<tamanhoMatriz;i++){
eleMatriz[i][0] =(Elemento)vec.elementAt(i) ;
}
progresso(tamanhoMatriz,tamanhoMatriz*0.2);

    for(int j=1;j<tamanhoMatriz;j=j+2){
eleMatriz[j-1][1] = eleMatriz[j][0];
eleMatriz[j][1] = eleMatriz[j-1][0];
}
for(int k=3;k<tamanhoMatriz;k=k+4){

    eleMatriz[k-3][2] = eleMatriz[k-1][0];
eleMatriz[k-2][2] = eleMatriz[k-0][0];
eleMatriz[k-3][3] = eleMatriz[k-1][1];
eleMatriz[k-2][3] = eleMatriz[k-0][1];

    eleMatriz[k-1][2] = eleMatriz[k-3][0];
eleMatriz[k-0][2] = eleMatriz[k-2][0];
```

```
eleMatriz[k-1][3] = eleMatriz[k-3][1];
eleMatriz[k-0][3] = eleMatriz[k-2][1];
    }
progresso(tamanhoMatriz,tamanhoMatriz*0.4);
    for(int s = 4;s<=tamanhoMatriz;s=s*2){
for(int m = 0;m<tamanhoMatriz;m=m+s){
for(int n = 0;n<tamanhoMatriz;n=n+s){
for(int d=0;d<s;d++){
for(int h=0;h<s;h++){
eleMatriz[m+d+s][n+s+h]=eleMatriz[m+d][n+h];
}
}
}
}
for(int m = s;m<tamanhoMatriz;m=m+(s*2)){
for(int n = 0;n<tamanhoMatriz;n=n+s){
for(int d=0;d<s;d++){
for(int h=0;h<s;h++){
eleMatriz[m+d-s][n+s+h]=eleMatriz[m+d][n+h];
}
}
}
}
}
progresso(tamanhoMatriz,tamanhoMatriz*0.8);
    mostraMatriz(tamanhoMatriz,tamanhoMatriz);
}
    private void mostraMatriz(int a, int b){
jTextArea1.append("Matriz Gerada:\n");
for(int y = 0;y<b;y++){
for(int x=0;x<a;x++){
jTextArea1.append(eleMatriz[x][y].getHexadecimal()+" ");
}
jTextArea1.append("\n");
}
jTextArea1.append("\n");
progresso(tamanhoMatriz,tamanhoMatriz*1);
}
    private void mostraMatrizXor(int a, int b){
jTextArea1.append("Matriz Gerada XOR:\n");
for(int y = 0;y<b;y++){
for(int x=0;x<a;x++){
jTextArea1.append(eleMatriz[x][y].getXorCount()+" ");
}
jTextArea1.append("\n");
```

```
}
jTextArea1.append("\n");
}
private void mostraMatrizXTime(int a, int b){
jTextArea1.append("Matriz Gerada xTime:\n");
for(int y = 0;y<b;y++){
for(int x=0;x<a;x++){
jTextArea1.append(eleMatriz[x][y].getXTimeCount()+" ");
}
jTextArea1.append("\n");
}
jTextArea1.append("\n");
    }
    private String calculaXor(String primeiro, String segundo){
String resultado = "";
String zero = "";
int dif = segundo.length() - primeiro.length();
if(dif > 0){
for(int o=1;o < dif+1;o++){
zero = "0" + zero;
}
primeiro = zero + primeiro;
}
zero = "";
if(dif < 0){
dif = dif * -1;
for(int j=1;j < dif+1;j++){
zero = "0" + zero;
}
segundo = zero + segundo;
}
//System.out.println("—->"+primeiro+"<—-");
//System.out.println("++++>"+segundo+"<+++++");
    int comp = primeiro.length();
char bitP,bitS,bitR = '-';
for(int c = 0;c<comp;c++){
//System.out.println("**"+primeiro.charAt(c)+"**");
//System.out.println(">>>"+segundo.charAt(c)+"<<<");
bitP = primeiro.charAt(c);
bitS = segundo.charAt(c);
if(bitP != bitS){
bitR = '1';
}else{
bitR = '0';
}
resultado = resultado + bitR;
    }
```

```
//System.out.println("=====>"+resultado+"<====");
return resultado;
}

    public String convertToTen(String s1, int base){
//System.out.println("Received: "+s1+" base: "+base);
String s;
if (s1.indexOf(".") > 0)
s = s1.substring(0,s1.indexOf("."));
else
s = new String(s1);
// Convert higher bases to 10
int []a = new int[s.length()];
for (int i=0;i<s.length();i++)
a[s.length()-i-1] = charValueOf(s.charAt(i));

    double x = 0;
for (int i=0;i<s.length();i++) {
//System.out.print(" "+a[i]);
x+=a[i]*Math.pow(base,i);
}
//System.out.println("In base 10 is: "+x);
return ""+x;
}

    public int charValueOf(char c){
if (c >= 'a' && c <= 'z')
return (int)Integer.parseInt(""+(int)(c-97+10));

    if (c >= 'A' && c <= 'Z')
return (int)Integer.parseInt(""+(int)(c-'A'+10));

    if (c >= '0' && c <= '9')
return (int)Integer.parseInt(""+(char)c);

    return 0;
}

    public char intValueOf(int x){
if (x >= 0 && x <= 9)
return (char)('0'+x);
if (x >= 10)
return (char)('a'+(x-10));

    return 0;
}

    public String convertFromTen(double num, int toBase){
String s = "";
int i = findLargest(num,toBase);
while (num > 0.0000000001 || i >= 0) {
if (s.length()>20)
num = 0.0;
if (i == -1)
```

```
s+=".";
//System.out.println("G: "+findNumLargest(num,toBase,i));
s+=""+intValueOf(findNumLargest(num,toBase,i));
num -= Math.pow(toBase,i)*findNumLargest(num,toBase,i);
i–;
    }
if(s.length()==1 && toBase==16){
s="0"+s;
}
return s;
}
    public int findLargest(double num, int toBase){
int n=0;
while (Math.pow(toBase,n) <= num)
n++;
return n-1;
}
    public int findNumLargest(double num, int toBase,int t){
int n=1;
while (n*Math.pow(toBase,t) <= num)
n++;
return n-1;
}
    private void geraCabecario(){
jTextArea1.append("Gerador de Matriz de Cauchy - GMC - Versão 1.0\n");
jTextArea1.append("Abrahão, E. e Nakahara, J. - Unisantos - Brasil\n");
jTextArea1.append("\n");
}
    private void geraRodape(){
jTextArea1.append("————————————FIM DE CçLCULO————————————\n");
    }
    private void adicionaLimiteHamming(){
jTextArea1.append("Limite Peso Hamming: "+maximoHamming +"\n");
jTextArea1.append("Limite xTimes : "+maxXTimes +"\n");
    }
    private void progresso(double valor, double atual){
    int rs= (int)(atual / valor * 100);
jProgressBar1.setValue(rs);
}
    public void jButton3_actionPerformed(ActionEvent actionEvent) {
//implementar logica da gravacao de dados
}
    public void jComboBox3_actionPerformed(ActionEvent actionEvent) {
int contador = (jComboBox3.getSelectedIndex());
```

```
//System.out.println(contador);
switch (contador){
case 0:
tamanhoMatriz = 2;
break;
case 1:
tamanhoMatriz = 4;
break;
case 2:
tamanhoMatriz = 8;
break;
case 3:
tamanhoMatriz = 16;
break;
case 4:
tamanhoMatriz = 32;
break;
}
}

    public void jComboBox1_actionPerformed(ActionEvent actionEvent) {
int contador = (jComboBox1.getSelectedIndex());
maximoHamming = contador+1;
}

    public void jComboBox2_actionPerformed(ActionEvent actionEvent) {
int contador = (jComboBox2.getSelectedIndex());
maxXTimes = contador+1; }
}

    class Frame1_jComboBox2_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jComboBox2_actionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}

    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jComboBox2_actionPerformed(actionEvent);
}
}

    class Frame1_jComboBox1_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jComboBox1_actionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}

    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jComboBox1_actionPerformed(actionEvent);
}
}

    class Frame1_jComboBox3_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jComboBox3_actionAdapter(Frame1 adaptee) {
```

```
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jComboBox3_actionPerformed(actionEvent);
}
}
    class Frame1_jButton3_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jButton3_actionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jButton3_actionPerformed(actionEvent);
}
}
    class Frame1_jButton2_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jButton2_actionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jButton2_actionPerformed(actionEvent);
}
}
    class Frame1_jButton1_actionAdapter implements ActionListener {
private Frame1 adaptee;
Frame1_jButton1_actionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jButton1_actionPerformed(actionEvent);
}
}
    class Frame1_jMenuFileExit_ActionAdapter implements ActionListener {
Frame1 adaptee;
    Frame1_jMenuFileExit_ActionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jMenuFileExit_actionPerformed(actionEvent);
}
}
    class Frame1_jMenuHelpAbout_ActionAdapter implements ActionListener {
```

```
Frame1 adaptee;
    Frame1_jMenuHelpAbout_ActionAdapter(Frame1 adaptee) {
this.adaptee = adaptee;
}
    public void actionPerformed(ActionEvent actionEvent) {
adaptee.jMenuHelpAbout_actionPerformed(actionEvent);
}
}
    package gmc;
    import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
    import javax.swing.*;
    /**
<p>Title: GMC</p>
<p>Description: Gerador de Matrizes de Cauchy</p>
<p>Copyright: Copyright (c) 2006</p>
<p>Company: Unisantos</p>
@author Elcio Abrahão e Jorge Nakaraha Jr.
@version 1.3
/
public class Frame1_AboutBox extends JDialog implements ActionListener {
JPanel panel1 = new JPanel();
JPanel panel2 = new JPanel();
JPanel insetsPanel1 = new JPanel();
JPanel insetsPanel2 = new JPanel();
JPanel insetsPanel3 = new JPanel();
JButton button1 = new JButton();
JLabel imageLabel = new JLabel();
JLabel label1 = new JLabel();
JLabel label2 = new JLabel();
JLabel label3 = new JLabel();
JLabel label4 = new JLabel();
ImageIcon image1 = new ImageIcon();
BorderLayout borderLayout1 = new BorderLayout();
BorderLayout borderLayout2 = new BorderLayout();
FlowLayout flowLayout1 = new FlowLayout();
GridLayout gridLayout1 = new GridLayout();
String product = "GMC";
String version = "1.3";
String copyright = "Copyright (c) 2006";
String comments = "Gerador de Matrizes de Cauchy";
    public Frame1_AboutBox(Frame parent) {
super(parent);
```

```
try {
setDefaultCloseOperation(DISPOSE_ON_CLOSE);
jbInit();
} catch (Exception exception) {
exception.printStackTrace();
}
}
    public Frame1_AboutBox() {
this(null);
}
    /**
Component initialization.
@throws java.lang.Exception
/
private void jbInit() throws Exception {
image1 = new ImageIcon(gmc.Frame1.class.getResource("about.png"));
imageLabel.setIcon(image1);
setTitle("About");
panel1.setLayout(borderLayout1);
panel2.setLayout(borderLayout2);
insetsPanel1.setLayout(flowLayout1);
insetsPanel2.setLayout(flowLayout1);
insetsPanel2.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
gridLayout1.setRows(4);
gridLayout1.setColumns(1);
label1.setText(product);
label2.setText(version);
label3.setText(copyright);
label4.setText(comments);
insetsPanel3.setLayout(gridLayout1);
insetsPanel3.setBorder(BorderFactory.createEmptyBorder(10, 60, 10, 10));
button1.setText("OK");
button1.addActionListener(this);
insetsPanel2.add(imageLabel, null);
panel2.add(insetsPanel2, BorderLayout.WEST);
getContentPane().add(panel1, null);
insetsPanel3.add(label1, null);
insetsPanel3.add(label2, null);
insetsPanel3.add(label3, null);
insetsPanel3.add(label4, null);
panel2.add(insetsPanel3, BorderLayout.CENTER);
insetsPanel1.add(button1, null);
panel1.add(insetsPanel1, BorderLayout.SOUTH);
panel1.add(panel2, BorderLayout.NORTH);
setResizable(true);
}
```

```
    /**
Close the dialog on a button event.

@param actionEvent ActionEvent
/
public void actionPerformed(ActionEvent actionEvent) {
if (actionEvent.getSource() == button1) {
dispose();
}
}
}
```

Referências Bibliográficas

[1] Abrahão, E., *Improving RFID Tags Protocol to Assure Secure Ownership Transfer*, Relatório Técnico, Unisantos, Santos, SP, 2007. xiv, xv, 39, 40, 44, 47, 49, 51

[2] AES, *The Advanced Encryption Standard Development Process*, 1997, http://csrc.nist.gov/encryption/aes/ 27, 58

[3] Alien press release, *http://www.alientechnology.com/*, Mar. 30, 2004. 34

[4] Avoine, G *Open Issues in Radio Frequency Identification*. Conference MyCrypt, Kuala Lumpur, Malaysia, September, 2005. xiv, 24, 33, 37

[5] Avoine, G *Rfid security*. SMART UNIVERSITY. Sophia-Antipolis, France, 2006. 36, 39

[6] AutoID Labs homepage, http://www.autoidlabs.org. 35

[7] P.S.L.M. Barreto, V. Rijmen, *The ANUBIS Block Cipher*, 1st NESSIE Workshop, Heverlee, Belgium, Nov. 2000, http://cryptonessie.org 61, 62, 66

[8] P.S.L.M. Barreto, V. Rijmen, *The KHAZAD Legacy-Level Block Cipher*, 1st NESSIE Workshop, Heverlee, Belgium, Nov. 2000, http://cryptonessie.org 61, 62

[9] E. Biham, N. Keller, *Cryptanalysis of Reduced Variants of Rijndael*, 3rd AES Conference, New York, USA, 2000, http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html 73, 74

[10] A. Biryukov, *Analysis of Involutional Ciphers: Khazad and Anubis*, 10th Fast Software Encryption Workshop, T. Johansson,Ed., Springer-Verlag, LNCS 2887, 2003, 45–53. 75

[11] A. Biryukov, *The Boomerang Attack on 5 and 6-round Reduced AES*, Proceedings of AES4 Conference, H. Dobbertin, V. Rijmen, A. Sowa, Eds., Springer-Verlag, LNCS 3373, 2004, 11–15. 73

[12] A. Biryukov, D. Wagner, *Slide Attacks*, 6th Fast Software Encryption Workshop, L.R. Knudsen,Ed., Springer-Verlag, LNCS 1636, 1999, 245–259. 77

[13] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, D. Zuckerman, *An XOR-Based Erasure-Resilient Coding Scheme*, ICSI TR-95-048, Aug. 1995. 66

[14] J.H. Cheon, M. Kim, K. Kim, J.-Y. Lee, S.W. Kang, *Improved Impossible Differential Cryptanalysis of Rijndael and Crypton*, Proceedings of ICISC 2001, K. Kim, Ed., Springer-Verlag, LNCS 2288, 2001, 39–49. 72

[15] D. Coppersmith, *The Data Encryption Algorithm and its Strength Against Attacks*, IBM Journal on Research and Development, 38:(3), 1994, 243–250. 73

[16] N.T. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Quadratic Equations*, Advances in Cryptology, Asiacrypt'02, Y. Zheng,Ed., Springer-Verlag, LNCS 2501, 2002, 267–287. 77

[17] J. Daemen, L.R. Knudsen, V. Rijmen, *The Block Cipher SQUARE*, 4th Fast Software Encryption Workshop, E. Biham,Ed., Springer-Verlag, LNCS 1267, 1997, 149–165. 64, 74

[18] J. Daemen, V. Rijmen, *AES Proposal: Rijndael*, 1st AES Conference, California, USA, 1998, http://www.nist.gov/aes 61, 72, 73

[19] J. Daemen, V. Rijmen, *The Design of Rijndael – AES – The Advanced Encryption Standard*, Springer-Verlag, 2002. 58, 61, 62, 66, 68, 72

[20] T. Dimitriou *A lightweight RFID protocol to protect against traceability and cloning attacks*. In Conference on Security and Privacy for Emerging Areas in Communication Networks - SecureComm, Athens, Greece, September 2005. IEEE. 23, 43, 46, 54

[21] EAN International and the Uniform Code Council. *http://www.ean-int.org* 22, 40

[22] M. Feldhofer, S. Dominikus, J.Wolkerstorfer *Strong authentication for RFID systems using the AES algorithm*. In Marc Joye and Jean-Jacques Quisquater, editors, Workshop on Cryptographic Hardware and Embedded Systems (CHES), volume 3156 of Lecture Notes in Computer Science, pages 357–370, Boston, Massachusetts, USA, Aug. 2004, Springer-Verlag. 39

[23] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, D. Whiting, *Improved Cryptanalysis of Rijndael*, 7th Fast Software Encryption Workshop, B. Schneier, Ed., Springer-Verlag, LNCS 1978, 2000, 213–230. 72

[24] S.L. Garfinkel *Adopting Fair Information Practices in Low-Cost RFID Systems*, Ubiquitous Computing, Sep. 2002.

[25] S.L. Garfinkel *An RFID Bill of Rights* Technology Review, p. 32, October, 2002. 52

[26] H. Gilbert, M. Minier, *A Collision Attack on Seven Rounds of Rijndael*, 3rd AES Conference, New York, USA, 2000, http://csrc.nist.gov/encryption/aes/ 73, 74

[27] D. Henrici, P. Muller *Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers*, Workshop on Pervasive Computing and Communications Security, 2004. 42

[28] A. Juels *Minimalist cryptography for low-cost RFID tags.* In C. Blundo and S. Cimato, Ed.s, The Fourth International Conference on Security in Communication Networks - SCN 2004, LNCS 3352, Springer-Verlag, 149-164, Amalfi, Italia, Sep. 2004. 39, 41

[29] P. Junod, S. Vaudenay, *Perfect Diffusion Primitives for Block Ciphes – Building Efficient MDS Matrices*, H. Handschuh, A. Hasan, Eds., Springer-Verlag, LNCS 3357, 84–99. 65, 66

[30] D. Kahn, *The Codebreakers: the Story of Secret Writing*, MacMillan Publishing Co. Inc., 1967. 24, 61

[31] J. Kelsey, B. Schneier, D. Wagner, *Mod n Cryptanalysis, with Applications against RC5P and M6*, 6th Fast Software Encryption Workshop, L.R. Knudsen,Ed., Springer-Verlag, LNCS 1636, 1999, 139–155. 77

[32] Landt, J *Shrouds of Time - The history of RFID.* Pittsburgh, PA, USA, October http://www.aimglobal.org 30, 31

[33] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology, Eurocrypt'93, T. Helleseth, Ed., Springer-Verlag, LNCS 765, 1994, 386–397. 74

[34] R.J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, DSN progress report 42-44, Jet Propulsion Laboratory, Pasadena, 1978. 63

[35] F.J. McWilliams, N.J.A. Sloane, *The Theory of Error Correcting Codes*, Amsterdam, The Netherlands, North Holland, 1977. 63

[36] A.J. Menezes, P.C. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997. 49, 52, 55, 62

[37] D.A., Molnar *Security and Privacy in Two RFID Deployments, With New Methods For Private Authentication and RFID Pseudonyms*. Dept. of Electrical Engineering and Computer Science, University of California at Berkeley, MSc, Plan I, 2006. 44, 52

[38] Nakahara, J. Abrahão, E., *A new involutory MDS matrix for the AES*, International Jornal of Network Security (IJNS), 2007, to appear. xv, 66, 74, 75

[39] NBS, *Data Encryption Standard (DES)* , Federal Information Processing Standards, FIPS PUB 46, US Department of Commerce, Jan, 1977. 72

[40] NIST, *Advanced Encryption Standard (AES)*, FIPS PUB 197 Federal Information Processing Standard Publication 197, U.S. Department of Commerce, Nov, 2001. 58, 64

[41] NIST, *Skipjack and KEA Specification, version 2.0*, May 1998, http://csrc.nist.gov/encryption/skipjack–kea.htm 77

[42] M. Ohkubo, K. Suzuki and S. Kinoshita *Cryptographic Approach to Privacy-friendly Tags*, In RFID Privacy Workshop, MIT, 2003. 41

[43] R.C.W. Phan, *Classes of Impossible Differentials of Advanced Encryption Standard*, IEE Electronics Letters, 38:(11), May 2002, 508–510. 73

[44] R.C.W. Phan, M.U. Siddiqi, *Generalized Impossible Differentials of Advanced Encryption Standard*, IEE Electronics Letters, 37:(14), Jul. 2001, 896–898. 73, 74

[45] Raghu D; Harrop*RFID Forecast, Players and Opportunities 2006 - 2016* Downig Park, Swaffham Bulbeck, Cambridge, UK, October 2006. Acessado em 21/11/2006. http://www.idtechex.com/products/en/view.asp?publicationid=137 21

[46] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, E. De Win, *The Cipher SHARK*, 3rd Fast Software Encryption Workshop, D. Gollmann, Ed., Springer-Verlag, LNCS 1039, 1996, 99–112. 64

[47] Shannon, C. *Communication theory of secrecy systems*. 656–715 28 Bell System Technical Journal, October 1949. 25

[48] The Block Cipher Rijndael, http://www.iaik.tugraz.at/research/krypto/AES/ 58

[49] Ungaretti, R*O uso de software livre em criptografia: razões históricas*. Outubro 2002. http://www.comciencia.br/reportagens/guerra/guerra22.htm 26

[50] S. Vaudenay, *On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER*, 2nd Fast Software Encryption Workshop, B. Preneel,Ed., Springer-Verlag, LNCS 1008, 1995, 286–297. 63

[51] S. Weis and S. Sarma and R.L. Rivest and D. Engels *Security and privacy aspects of low-cost radio frequency identification systems*. In D. Hutter, G.Muller, W. Stephan, M. Ullmann, Ed.s, International Conference on Security in Pervasive Computing, SPC 2003, LNCS 2802, Springer-Verlag, 454-469, Boppard, Germany, Mar. 2003. 35

[52] Wired News. Wal-Mart, DoD forcing RFID. Available at http://www.wired.com/news/business/0,1367,61059,00.html, Nov. 2003. 22

[53] A.M. Youssef, S. Mister, S.E. Tavares, *On the Design of Linear Transformation for Substitution Permutation Encryption Networks*, Workshop on Selected Areas in Cryptography, SAC97, Ottawa, Canada, 1997, 40–48. 66