

**Cartão MIFARE classic
ataques e medidas de contorno**

Wellington Baltazar de Souza

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Programa: Ciência da Computação
Orientador: Prof. Dr. Routh Terada

São Paulo, Novembro de 2011

Cartão MIFARE classic, ataques e medidas de contorno

Resumo

O cartão MIFARE Classic é um dos cartões sem contato mais utilizados no mundo. Foi criado pela NXP Semiconductors (antiga Philips), e utiliza comunicação RFID compatível com o padrão ISO 14443A. No entanto utiliza protocolo e criptografia proprietários, tendo como princípio “segurança por obscuridade”. Este tipo de cartão vem sendo utilizado pela indústria em sistemas de controle de acesso em edifícios e principalmente substituindo tíquetes de transporte público. Em 2008 dois grupos de pesquisadores, trabalhando de forma mais ou menos independente realizaram a engenharia reversa do protocolo de comunicação e da cifra Crypto-1 utilizados no cartão descobrindo uma série de fraquezas de segurança que comprometeram seriamente a reputação do cartão. Como consequência um usuário mal intencionado poderá clonar o cartão em alguns segundos. Desde então, o MIFARE Classic tem recebido muita exposição na mídia, e vem sendo pesquisado outras formas de ataque, pois existem diversos sistemas importantes que ainda utilizam esta tecnologia com a segurança do cartão comprometida. Neste trabalho serão apresentados as características do cartão, os principais tipos de ataques, bem como alternativas para contorná-los e na medida do possível manter o sistema seguro.

Palavras-chave: MIFARE Classic, Crypto-1, fraquezas, ataques, medidas de contorno.

MIFARE classic card, attacks and countermeasures

Abstrac

The MIFARE Classic is one of the most used contactless cards in the world. It has been created by NXP Semiconductors (formely Philips) and uses RFID communication, which is ISO 14443A compliant. Nonetheless, its protocol and cryptography are proprietary, based upon the “security by obscurity” principle. The Industry has been using this card in access control systems deployed in buildings, as well as in the public transportation as a ticket replacement. In 2008, two groups of researchers, conducting their work almost independently, have performed the card communication protocol and Crypto-1 cipher reverse engineering, uncovering several security weaknesses, which has jeopardized the card reputation. As a consequence, malicious users might clone this card in a couple of seconds. Since then, the MIFARE Classic has been highly exposed on the media. Besides that, other forms of attack have been researched, once there are numerous important systems yet using this undermined technology. This work is intended to present the card features, the main types of attack, workarounds to control them and, as much as possible, keeping the system secure.

Keywords: MIFARE Classic, Crypto-1, weakness, attacks, countermeasures.

Sumário

Lista de Abreviaturas	ix
Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
1.1 Motivação	2
1.2 Objetivos e contribuições	3
1.3 Organização da Dissertação	3
2 Fundamentos de Criptografia	5
2.1 Criptanálise e Ataques	5
2.2 <i>One-time pad</i>	6
2.3 Criptografia de chave secreta/simétrica	6
2.4 Cifras de fluxo	7
2.5 Criptografia de chave pública	8
2.6 Funções de <i>Hash</i>	9
2.7 Assinatura Digital	10
3 O Cartão MIFARE Classic	11
3.1 Estrutura Lógica	11
3.2 Bloco do Fabricante	12
3.3 Bloco <i>Trailer</i> do Setor	12
3.4 Bloco de Dados	12
3.5 Condições de Acesso	13
3.6 Condições de Acesso para o Bloco <i>Trailer</i> do Setor	13
3.7 Condições de Acesso para Blocos de Dados	14
3.8 Acesso à Memória	15
3.9 Comunicação	16
3.9.1 Requisição Padrão/Todos	16
3.9.2 Loop Anti-Colisão	17
3.9.3 Seleção do Cartão	17
3.9.4 Integridade de Dados	17
3.9.5 Autenticação em Três Passos	18
3.9.6 Parada	18

3.9.7	Conjunto de Comandos	19
4	A Cifra Crypto-1	21
4.1	Cifra de Fluxo	22
4.2	Geração do <i>keystream</i>	22
4.3	Gerador de números pseudo-aleatórios	23
4.4	Autenticação	23
4.5	Inicialização da Cifra e Geração do <i>Keystream</i>	24
4.6	Bits de Paridade	25
5	Fraquezas Encontradas	27
5.1	Baixa entropia do gerador pseudo-aleatório	27
5.2	Algoritmo gerador do <i>keystream</i> não usa os bits mais à esquerda	28
5.3	Geração do <i>keystream</i> somente com bits ímpares	28
5.4	Bits de paridade e reuso do <i>one-time pad</i>	29
5.5	Exposição parcial do <i>keystream</i> nos bits de paridade	29
5.6	Autenticações Aninhadas	29
5.7	Autenticação com <i>timeout</i> expõe parte do <i>keystream</i>	29
5.8	Leitor aceita <i>frames</i> de tamanho inválido	30
5.9	Estatística viciada na cifra	30
5.10	Tamanho inadequado da chave	30
5.11	Aplicações em produção usando chaves de transporte	31
5.12	Cartões compatíveis muito fracos	31
6	Ataques	33
6.1	Hardware e Software	33
6.1.1	Leitores	33
6.1.2	Crypto1	34
6.1.3	libnfc.org	34
6.1.4	Proxmark III	34
6.2	Ataque por força bruta	35
6.3	Ataque por <i>timeout</i> na autenticação do cartão	36
6.4	Ataque por interceptação da autenticação	37
6.5	Ataque Somente ao Cartão	39
6.5.1	Detalhes do Ataque	40
6.5.2	Complexidade do Ataque	40
7	Medidas de Contorno	43
7.1	Diversificação de Chaves	44
7.2	Cifragem dos Dados	44
7.3	Verificação de Integridade com HMAC	44
7.4	Verificação de Integridade com Assinatura	45
7.5	Associação de UID	45
7.6	Verificação do contador de decremento	45
7.6.1	Cartões mais Seguros	46

7.7 Considerações Finais	46
8 Conclusões	49
8.1 Considerações Finais	49
8.2 Trabalhos Futuros	50
Referências Bibliográficas	51

Listas de Abreviaturas

ACK	Confirmação de recebimento de dados (<i>signal to acknowledge the receipt of data</i>).
API	<i>Application Programming Interface.</i>
ATQA	Resposta de Requisição para uma tag do Tipo A(<i>Answer To Request of Type A</i>).
BCC	Verificador de bits (<i>Bit Count Check</i>).
CI	Círculo Integrado.
CRC	Verificação de redundância cíclica (<i>Cyclic redundancy check</i>).
CFB	<i>Cipher Feedback.</i>
DES	<i>Data Encryption Standard.</i>
DSA	Algoritmo de Assinatura Digital (<i>Digital Signature Algorithm</i>).
DSS	Padrão de Assinatura Digital (<i>Digital Signature Standard</i>).
EEPROM	Memória Somente de Leitura Programável Apagável Eletricamente (<i>Electrically-Erasable Programmable Read-Only Memory</i>).
GSM	Sistema Global para Comunicações Móveis (<i>Global System for Mobile Communications</i>).
HLTA	Comando de “parar” para uma tag do tipo A (<i>HALT Command, Type A</i>).
IEC	<i>International Electronics Commission.</i>
ISO	<i>International Organization for Standardization.</i>
LFSR	Registrador de Deslocamento com Retroalimentação Linear (<i>Linear Feedback Shift Register</i>).
NACK	Confirmação de erro nos dados recebidos (<i>Negative acknowledgement response</i>).
NFC	<i>Near Field Communication.</i>
NIST	<i>National Institute of Standards and Technology.</i>
NONCE	Número utilizado apenas uma vez (<i>Number Used Once</i>)
NSA	<i>National Security Agency.</i>
OFB	<i>Output Feedback.</i>
PC	Computador Pessoal (<i>Personal Computer</i>).
PCD	Dispositivo de Acoplamento por Aproximação (<i>Proximity Coupling Device</i>).
PICC	Cartão de Circuito Integrado por Aproximação (<i>Proximity Integrated Circuit Card</i>).
REQA	Comando de Requisição para uma tag do Tipo A (<i>REQuest command, Type A</i>).
RF	Rádio Frequência.
RFID	Identificação por Rádio Frequência (<i>Radio Frequency Identification</i>).
SAK	Confirmação da resposta do comando de Seleção (<i>Select AcKnowledge reply</i>).
SAM	<i>Security Access Module</i>
SDK	Kit de Desenvolvimento de Software (<i>Software Development Kit</i>).
SIM	Módulo de Identificação do Assinante (<i>Subscriber Identity Module</i>).

- UID Identificador Único (*Unique Identifier*).
WUPA Comando para “ligar” uma tag do Tipo A (*Wake UP command, Type A*).

Listas de Figuras

2.1	Criptografia de Chave Secreta ou Simétrica	6
2.2	Estrutura das cifras de fluxo	7
2.3	Cifra de fluxo em operação	7
2.4	Criptografia de Chave Pública	8
2.5	Procedimentos para assinar digitalmente um documento	10
3.1	Estrutura Lógica do cartão MIFARE <i>Classic</i> de 1K	11
3.2	Bloco 0, Setor 0 - Dados do Fabricante	12
3.3	Bloco Trailer do Setor	12
3.4	Bloco de Valor	12
3.5	Disposição dos Bits das Condições de Acesso	13
3.6	Sequência de procedimentos para acesso à memória	15
3.7	Fluxo da Comunicação utilizada no MIFARE Classic	16
3.8	Autenticação em três passos	18
4.1	Diagrama com LFSR e as funções filtro da Cripto-1	22
4.2	Inicialização da Cifra Crypto-1	24
5.1	Gerador de números pseudo-aleatórios usa apenas 16 bits	27
5.2	Bits mais à esquerda do LFSR não utilizados pelo filtro gerador	28
5.3	Geração do keystream utiliza apenas bits ímpares do LFSR	28
5.4	Criptografia dos bits de paridade	29
6.1	Leitores comuns padrão de mercado	33
6.2	Proxmark III	34
6.3	Visão superior do COPACOBANA	35
6.4	Autenticação com Timeout	36
6.5	Captura de logs da comunicação entre cartão e leitor	37
6.6	Exemplo de Log Capturado pelo Proxmark III	37
7.1	Exemplo de fraude detectada pelo back office	46
7.2	Nível de segurança de acordo com as medidas de contorno aplicadas	46

Lista de Tabelas

3.1	Condições de acesso	13
3.2	Condições de Acesso para o bloco Trailer do Setor	14
3.3	Condições de acesso para bloco de dados	14
3.4	Operações na Memória	15
3.5	Tipos de cartões identificados pelas respostas ATQA/SAK	17
3.6	Conjunto de comandos utilizados entre o leitor e o cartão	19
3.7	Exemplo de um log da comunicação entre Leitor e Cartão	19
5.1	Comparativo entre o MIFARE Classic e o SIM do GSM	30
5.2	Chaves de transporte gravadas pelo fabricante	31

Capítulo 1

Introdução

Nos últimos anos os procedimentos de identificação automática vem se popularizando em diversos segmentos da indústria tais como suprimentos, logística, manufatura e fluxo de materiais. Os procedimentos de identificação automática são usados para fornecer informações a respeito de pessoas, animais, bens e produtos em trânsito.

De acordo com Finkenzeller, um procedimento de identificação automática ideal deve ser capaz de armazenar uma quantidade razoável de dados e ser passível de atualização. A transferência de dados entre o identificador e o leitor deve ser preferencialmente sem contato, pois o contato mecânico dependendo da aplicação é inviável. A comunicação sem contato é mais adequada pela maior flexibilidade. (Finkenzeller, 2003). Além disso, a comunicação sem contato pode aumentar consideravelmente a vida útil tanto do leitor como do identificador, pois não ocorre contato físico.

RFID (*Radio Frequency Identification* - Identificação por Rádio Frequência) é uma técnica de identificação automática que utiliza comunicação sem fio, através da aproximação entre um transponder (conhecido genericamente como tag) e um leitor (conhecido como PCD, *Proximity Coupling Device* - Dispositivo de Acoplamento por Aproximação). Esta tecnologia pode transferir pequenas quantidades de dados através de uma determinada distância. Finkenzeller (2003) explica que as tags ativas incorporam uma bateria para fornecer toda a energia, ou parte dela, necessária para a operação de um microchip. Por outro lado, as tags passivas não possuem fonte própria de energia, sendo que toda a energia necessária para as operações de uma tag passiva deve ser obtida do campo eletromagnético do leitor.

Embora muitas pessoas tratem RFID como uma nova tecnologia, foi desenvolvida na década de 40, e vem sendo utilizada desde a Segunda Guerra Mundial. Na época foi empregada para distinguir um avião de guerra dos aliados de um avião inimigo, sendo que hoje ela é usada em diversas áreas do nosso dia a dia (Wolfram *et al.*, 2008).

Os sistemas RFID vem substituindo gradualmente sistemas legados tais como código de barras, cartões com trilha magnética, tíquetes de papel usados em eventos e no transporte público. Além disso, vem sendo amplamente utilizado em pedágios, passaportes eletrônicos, controle de acesso em áreas restritas, celulares e até mesmo em implantes sob a pele de animais.

Alguns padrões ISO descrevem como deve funcionar um dispositivo RFID, sendo que grande parte das técnicas utilizam identificação de alta frequência conforme o padrão ISO 14443. Dois métodos de modulação são descritos no padrão ISO 14443, nas variantes *A* e *B*. O foco deste trabalho é na variante *A* denominado ISO14443A. Este é o padrão utilizado nos cartões inteligentes sem contato¹). A tag neste caso é o cartão, também conhecido como PICC (*Proximity Integrated Circuit Card* - Cartão de Circuito Integrado por Aproximação).

¹Do inglês, *contactless smart cards*

Os padrões ISO para sistemas RFID não especificam nenhuma segurança como autenticação, integridade, autorização ou disponibilidade. No entanto, no caso de cartões de identificação e nos dispositivos que utilizam o padrão ISO 14443A as funcionalidades de segurança são essenciais, pois em função da interface sem fio, eles não precisam de contato direto e assim, estão vulneráveis à comunicação despercebida entre o cartão e um leitor mal intencionado. Nesse sentido, é de fundamental importância implementar uma comunicação segura na camada de transporte. A questão é se o padrão ISO não cobre segurança, significa que qualquer empresa precisa desenvolver sua própria camada de segurança. O que se vê na prática é que uma empresa projeta um sistema adaptado de outras empresas para manter baixo o custo de produção e um certo grau de compatibilidade. (Verdult, 2008).

Existe uma grande variedade de cartões sem contato no mercado, como por exemplo as famílias dos cartões MIFARE, HID iCLASS, FeliCa, Toppan, ASK, Oberthur, Orga, Gemplus, entre outras. Se diferenciam de tamanho, invólucro, memória e capacidade de processamento. Normalmente eles fornecem algum suporte para criptografia simétrica, o que os tornam bastante convenientes para o uso em aplicações com requisitos de segurança. O cartão sem contato mais conhecido e amplamente utilizado é o MIFARE, um produto da NXP Semiconductors (antiga Philips). De acordo com a NXP, em 2008 existiam cerca de 200 milhões de cartões MIFARE em uso no mundo, representando cerca de 85% dos cartões inteligentes sem contato. A família MIFARE contém quatro tipos de cartões: *Ultralight*, *Classic*, *Plus*, *DESFire* e *SmartMX*. A versão *Classic* é a mais utilizada atualmente, disponível em três capacidades de armazenamento: 320B, 1K e 4K; possui autenticação mútua e segurança através da cifra proprietária *Crypto-1* que foi mantida em segredo desde a sua concepção.

A especificação do MIFARE Classic permaneceu em segredo por mais de 10 anos, até que em 2008 dois grupos de pesquisadores (Gans *et al.*, 2008; Nohl *et al.*, 2008) trabalhando de forma mais ou menos independente e com diferentes abordagens, realizaram a engenharia reversa do cartão MIFARE Classic e descobriram fraquezas importantes. Em particular uma deficiência no gerador de números aleatórios tanto no cartão como no leitor, e uma vulnerabilidade na cifra Crypto-1 Gans *et al.* (2008). A consequência disso, é que a chaves secreta utilizada pode ser descoberta em alguns segundos se um atacante tiver acesso à comunicação RF (Rádio Frequência), possibilitando a clonagem ou alteração não autorizada dos dados. Depois de publicações apontando fraquezas encontradas no cartão MIFARE Classic, tornou-se possível a exploração de outros tipos de ataques, os quais foram publicados posteriormente em (Garcia *et al.*, 2009) e (Courtois, 2009b).

1.1 Motivação

O MIFARE Classic foi introduzido no mercado em 1995, quando a NXP ainda era Philips, e rapidamente se tornou um produto de sucesso. Vem sendo utilizado principalmente em sistemas de transporte público, controle de acesso em prédios e até mesmo em bases militares. Estima-se que até o final de 2008 foram produzidos 3,5 bilhões de cartões MIFARE Classic. Esta é uma situação preocupante, pois todos estes cartões estão vulneráveis e sujeitos a fraude Mills (2008).

Mesmo depois das publicações que apontam sérios problemas de segurança no cartão MIFARE Classic mostrando de fato que a segurança dele foi comprometida, a indústria parece que não está muito preocupada com isso, pois ele ainda continua sendo bastante utilizado, em alguns casos até mesmo em novos projetos. Provavelmente continuará sendo utilizado por muito tempo em função dos investimentos já realizados e pelo baixo custo. A mudança para um modelo de cartão mais seguro, certamente envolveria aumento de custo na aquisição de equipamentos que teoricamente utilizam tecnologia mais segura. Entretanto, o volume de fraudes com aplicações que utilizam o MIFARE Classic pode aumentar significativamente, pois o custo para a realização de alguns tipos de ataques é relativamente barato, pois requerem apenas um leitor genuíno acoplado ao PC. Além disso, existem

ataques implementados em linguagem C, com o código aberto e disponíveis na internet, que servem de modelo e podem ser facilmente adaptados conforme o interesse do atacante.

1.2 Objetivos e contribuições

Os principais objetivos e contribuições deste trabalho são:

- Realizar um levantamento de trabalhos publicados sobre os problemas de segurança encontrados com o cartão MIFARE Classic;
- Explicar as características do cartão, estrutura interna e protocolo de comunicação, para facilitar o entendimento dos tipos de ataque;
- Expor as fraquezas encontradas, indicando possíveis pontos de ataque;
- Apresentar os principais ataques;
- Propor alternativas para evitar ataques;
- Implementar provas de conceito para demonstrar a viabilidade técnica da solução proposta para manter o sistema seguro.

1.3 Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

- No Capítulo 2, encontram-se alguns conceitos de criptografia necessários ao entendimento dos próximos capítulos;
- No Capítulo 3, são descritas as características físicas, lógicas e os detalhes do protocolo de comunicação do cartão MIFARE Classic;
- No Capítulo 4, são apresentados os detalhes de implementação da cifra Crypto-1 descobertos por pesquisadores via engenharia reversa;
- No Capítulo 5, são apresentadas as fraquezas encontradas no cartão MIFARE Classic como possíveis brechas para ataques;
- No Capítulo 6, são tratados os possíveis tipos de ataque bem como suas consequências;
- No Capítulo 7, são propostas as possíveis medidas de contorno referentes aos ataques do Capítulo 6.
- Por fim, no Capítulo 8, será apresentado um balanço a respeito dos ataques e tais medidas de contenção e trabalhos futuros.

Capítulo 2

Fundamentos de Criptografia

Este capítulo apresenta uma introdução dos conceitos de criptografia necessários para o entendimento de assuntos que serão abordados nos próximos capítulos.

A palavra criptografia vem do grego *kryptós* (escondido) + *gráphein* (escrita). Terada (2008) argumenta que objetivo de um algoritmo criptográfico é “esconder” informações sigilosas de qualquer pessoa desautorizada a lê-las, isto é, de qualquer pessoa que não conheça a *chave* secreta de criptografia.

Um algoritmo criptográfico pode ser visto como uma função f que usando uma chave K , recebe de entrada uma mensagem x , sendo capaz de produzir um texto criptografado y que somente pode ser decriptografado aplicando a inversa da função f utilizando a mesma chave K e o valor y como entrada. Por exemplo, suponha que Alice precise enviar uma mensagem confidencial m para Beto. Então Alice calcula $y = f_K(x)$ e envia para Beto. Tanto Alice como Beto conhecem a chave secreta K . Então Beto calcula $x = f_K^{-1}(y)$ e obtém o texto em claro. Este tipo de criptografia é conhecido como *Criptografia de Chave secreta* e será abordado na Seção 2.3.

2.1 Criptanálise e Ataques

Criptanálise é o estudo de métodos para descobrir a informação criptografada, sem no entanto conhecer a chave secreta. Normalmente, este trabalho requer o conhecimento de um profissional especializado em quebrar códigos chamado criptanalista.

Diz-se que um algoritmo criptográfico é quebrado ou quebrável quando um criptanalista consegue calcular a chave K conhecendo algumas entradas e suas respectivas saídas (tempo polinomial ao comprimento da chave).

Diversos tipos de ataques podem ser utilizados por um criptanalista para quebrar um algoritmo criptográfico, os quais dependem do tipo de informação disponível. Os tipos de ataque mais importantes ao Cartão MIFARE Classic são:

1. *Ataque por replay*: o criptanalista grava comunicações legítimas entre o cartão e o leitor, depois usa parte da informação gravada em seu favor. Se esta comunicação gravada não for alterada durante a comunicação, este ataque é passivo, caso contrário é chamado de ativo.
2. *Ataque adaptativo com texto escolhido*: A escolha de um novo texto em claro pelo criptanalista pode depender de textos cifrados anteriormente. Assim, a escolha de um novo texto em claro depende do conhecimento adquirido na análise de textos que foram cifrados anteriormente.
3. *Personificação*: o criptanalista simula um usuário legítimo, sem que o sistema note a diferença.

2.2 One-time pad

O *one-time pad* é conhecido como um sistema de criptografia incondicionalmente seguro, pois com o texto criptografado não há nenhuma possibilidade de se chegar ao texto em claro, a não ser que se conheça a chave. Como cifra, diz-se que o *one-time pad* oferece segurança perfeita e a sua análise é vista como um dos pilares da criptografia moderna (Shannon, 1949).

A ideia do *one-time pad* consiste em usar um fluxo de bits gerados de forma completamente aleatória, sendo a chave com tamanho maior ou igual a mensagem do texto em claro. Desta forma tem-se uma chave única pra cada mensagem. A mensagem em claro e a chave são combinadas utilizando operações de ou-exclusivo (XOR) para produzir o texto criptografado. Uma vez que a chave é aleatória, seria necessário um adversário com recursos computacionais infinitos para determinar o texto em claro, conhecendo apenas o texto criptografado.

O *one-time pad* foi usado em tempos de guerra, em canais de comunicação com requisitos de segurança excepcionalmente altos. O fato da chave secreta ser utilizada somente uma vez para cada mensagem, introduz diversos problemas para gerenciamento de chaves (RSA Laboratories, 1996). Além disso, o *one-time pad* é difícil de ser implementado, não só porque é muito difícil construir um algoritmo gerador de chaves realmente aleatórias (Terada, 2008), mas também devido a dificuldade de se gerenciar essas sequências entre o remetente e o destinatário. Embora perfeitamente seguro, o *one-time pad* é implementado para aplicações específicas.

2.3 Criptografia de chave secreta/simétrica

A *criptografia de chave secreta* também conhecida como *criptografia simétrica* utiliza a mesma chave secreta compartilhada para cifrar e decifrar, ou seja, utiliza chaves simétricas. Assim, juntando um algoritmo de criptografia simétrico, a chave secreta compartilhada, o texto em claro original é transformado em um texto cifrado em um processo de cifração. Usando o mesmo algoritmo com a chave secreta compartilhada, o texto em claro é recuperado novamente, em um processo inverso de decifração. A figura 2.1 ilustra essas transformações.

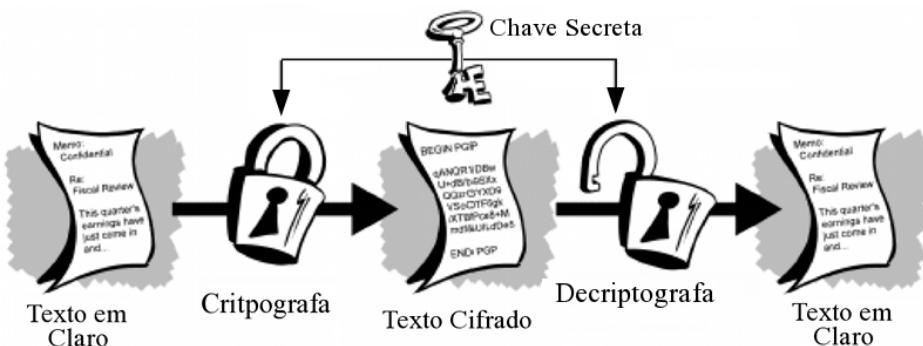


Figura 2.1: Criptografia de Chave Secreta ou Simétrica

A dificuldade de sem implementar criptografia de chave simétrica em sistemas de segurança criptográfica é a distribuição de chaves dos usuários. Esse ponto será discutido na Seção 2.5.

De acordo com Menezes *et al.* (1996), é possível classificar os algoritmos de criptografia dependendo da forma que o *texto em claro* é processado. As cifras de bloco processam o texto em claro em blocos relativamente grandes, por exemplo, n maiores que 64 bits. A mesma função é utilizada para criptografar blocos sucessivos. As cifras de bloco não possuem memória. Em contrapartida, cifras de fluxo¹ processam o texto em claro dividido em blocos menores, operando normalmente em

¹Do inglês, *stream cipher*

bits, e a função usada para criptografar pode variar à medida que o texto em claro é processado. Elas são às vezes chamadas de *cifras de estado*, pois a criptografia não depende apenas da *chave secreta* e do *texto em claro*, mas também do *estado corrente*.

2.4 Cifras de fluxo

As cifras de fluxo, tiveram sua origem com o objetivo de se ter uma aproximação do *one-time pad*. Elas não são capazes de satisfazer a segurança teórica do *one-time pad*, mas são mais fáceis de ser implementadas (RSA Laboratories, 1996).

As cifras de fluxo são mais adequadas para implementação em hardware que as cifras de bloco, pois possuem menos complexidade no circuito do hardware. Além disso, são mais apropriadas quando existe uma limitação no volume de dados a serem transferidos, ou quando os caracteres devem ser processados individualmente. Embora uma cifra de bloco possa se comportar como uma cifra de fluxo se for usada nos modos CFB (*Cipher feedback*) ou OFB (*Output Feedback*), as cifras de fluxo são projetadas serem mais rápidas (RSA Laboratories).

A chave é carregada em um gerador que produz uma sequência de bits aparentemente aleatórios. No final do processo de criptografia, a saída desse gerador, chamado *keystream*, é combinado um bit por vez com o fluxo do texto em claro usando operações de XOR (OU-exclusivo) para obter o texto criptografado. No final do processo de decriptografia, se o gerador usar a mesma chave, irá produzir o mesmo *keystream* que ao ser combinado com o texto criptografado usando XOR obtém-se o texto em claro. As Figuras 2.2 e 2.3 mostram a estrutura e o funcionamento típico de uma cifra de fluxo.

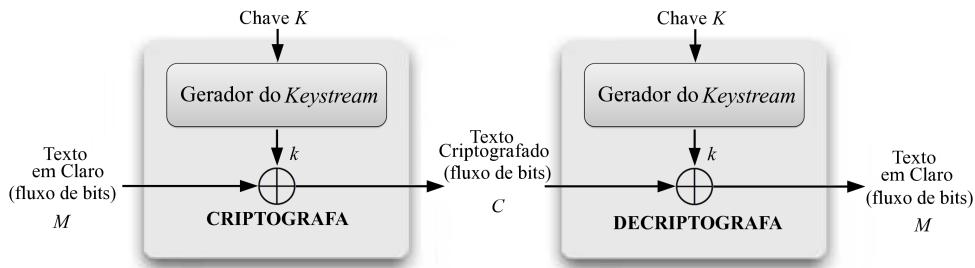


Figura 2.2: Estrutura das cifras de fluxo

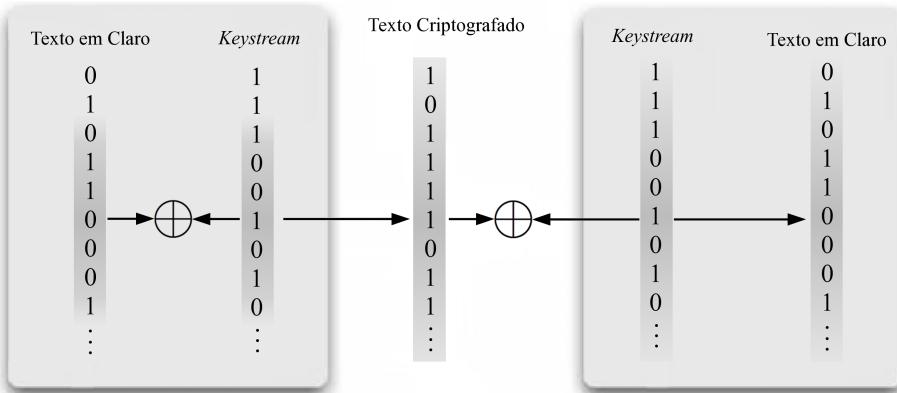


Figura 2.3: Cifra de fluxo em operação

Existem diversas maneiras de implementar um algoritmo gerador para o *keystream*. Uma das formas é combinar um LFSR (*Linear Feedback Shift Register* - Registrador de Deslocamento com Retroalimentação Linear) com um filtro gerador não linear. Um LFSR é um registrador cujo bit de entrada ou retroalimentação é uma função linear do seu estado anterior.

Nesse caso, a localização dos bits pré-determinados usados para produzir o bit de entrada são normalmente expressos na forma de um polinômio gerador. A função linear neste caso é um XOR (OU-exclusivo), portanto o bit de entrada é o XOR da soma de bits pré-determinados extraídos do valor do registrador de deslocamento.

O valor inicial do LFSR é chamado de chave. Como a função desse registrador é determinística, o fluxo de valores produzidos pelo registrador é completamente determinado pelo seu estado corrente (ou anterior). Do mesmo modo, como o registrador possui um número finito de estados possíveis, pode ocorrer eventualmente a entrada em um ciclo repetido. Porém, um LFSR com uma função linear de retroalimentação que possui grandes ciclos, pode produzir uma sequência de bits que aparentam ser aleatórios. Os LFSRs podem ser implementados em hardware de forma relativamente barata, tornando-os úteis em aplicações com requisitos de rápida geração de bits, como por exemplo as cifras de fluxo.

De acordo com Menezes *et al.* (1996), os LFSRs são largamente utilizados como geradores de *keystream*, pois são bem servidos por implementações de hardware, produzem sequências com grandes períodos e com boas propriedades estatísticas. Infelizmente as sequências de saída dos LFSRs são também legíveis em análises usando técnicas algébricas, sendo facilmente previsíveis.

2.5 Criptografia de chave pública

Apesar da simplicidade, a criptografia de chave secreta tem alguns problemas. Para se comunicar com segurança, cada dupla de usuários deve utilizar uma chave distinta. Nesse sentido, para uma rede com n usuários, seriam necessárias n^2 chaves, dificultando bastante o gerenciamento das chaves. Além disso, essas chaves devem ser trocadas e guardadas de forma segura entre as partes, o que nem sempre é possível.

O conceito de criptografia de chave pública foi introduzido por Diffie e Hellman (1976), para resolver o problema de gerenciamento de chaves. Cada usuário possui um par de chaves (S, P), sendo que a chave pública P de cada usuário é publicada, enquanto que a chave secreta S é cuidadosamente mantida em segredo. A necessidade do remetente e do destinatário compartilhar uma informação em segredo é eliminada. Toda a comunicação envolve somente as chaves públicas, sendo que nenhuma chave privada é transmitida ou compartilhada. Na Figura 2.4 depois da primeira seta, o remetente pode transmitir o texto criptografado para o destinatário. Quando o destinatário receber o texto criptografado, será capaz de decriptografá-lo e chegar ao texto em claro original.

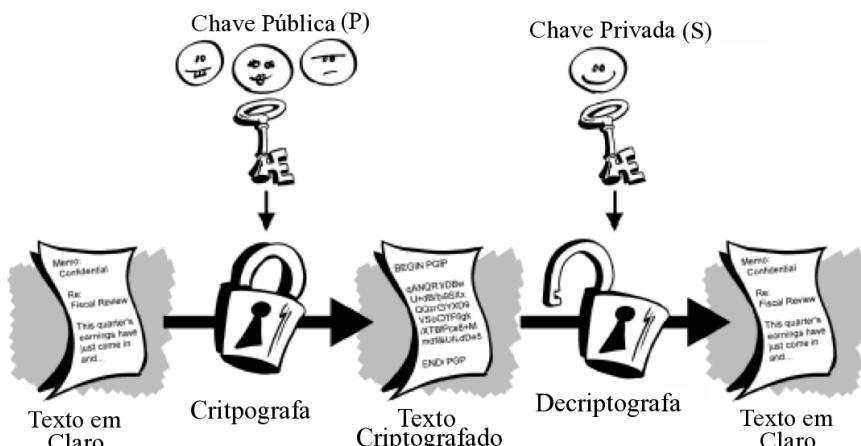


Figura 2.4: Criptografia de Chave Pública

As chaves P e S estão relacionadas matematicamente (Terada, 2008) de forma que:

1. Seja x um *texto em claro*, e $S()$ a aplicação da chave S que transforma x em $y = S(x)$, ou seja, y é o *texto criptografado*; então $P(y) = x$, sendo que $P()$ é a aplicação da chave P . Em outras palavras, a chave S é a inversa da chave P – $P(S(x)) = x$;
2. O custo computacional para se calcular o par de chaves (P, S) é relativamente fácil;
3. É muito difícil computacionalmente calcular S conhecendo P , e vice-versa;
4. De posse das chaves (P, S) , é fácil computacionalmente $P()$ e $S()$;
5. É muito difícil computacionalmente aplicar $S()$ sem conhecer S .

Essas propriedades tornam possível o seguinte cenário:

1. Cada usuário calcula seu par de chaves (P, S) em seu próprio computador;
2. A chave secreta S é cuidadosamente guardada de forma segura no computador;
3. A chave pública P é distribuída publicamente para todos.

A Infraestrutura de Chaves Públcas (ICP) é uma forma de relacionar as chaves públicas com seus respectivos usuários através de um certificado digital. A ICP é composta por conjunto de hardware, software, pessoas, políticas e procedimentos necessários para criar, gerenciar, distribuir, utilizar, armazenar e revogar certificados digitais.

2.6 Funções de Hash

Uma função de hash $H()$ (RSA Laboratories, 1996) é uma transformação que recebe uma mensagem x de tamanho variável e calcula $h = H(x)$, sendo h de tamanho fixo. Também é conhecida como, função *hashing* ou *Message Digest* (Terada, 2008). De maneira geral pode-se dizer uma função de hash gera um resumo de tamanho fixo para mensagens de qualquer tamanho.

Quando aplicadas para criptografia elas devem ter algumas propriedades adicionais:

- A entrada x pode ser longa,
- A saída possui tamanho fixo,
- $H(x)$ deve ser relativamente fácil de calcular para qualquer x ,
- $H(x)$ deve ser não inversível,
- Livre de colisões.

Uma função de hash é dita não-inversível se for muito difícil obter a sua inversa, significa que a partir de um valor de hash h , é computacionalmente inviável encontrar alguma entrada x tal que $H(x) = h$.

Diz-se que uma função de hash H é livre de colisões quando a partir de uma mensagem x , é computacionalmente inviável encontrar uma mensagem y tal que $H(x) = H(y)$. Além disso, como segundo critério, deve ser inviável encontrar quaisquer duas mensagens x e y tal que $H(x) = H(y)$.

O valor do hash é obtido através da função de hash, e representa concisamente uma mensagem ou um documento, pois com a característica de anti-colisão, é praticamente uma *impressão digital* do documento de origem. Assim, é possível aplicar as funções de hash para verificação de integridade

e melhorar o desempenho nos algoritmos de assinatura digital, usando apenas o *hash* resultante ao invés do texto inteiro. Mais detalhes sobre *Assinatura Digital* serão apresentados na Seção 2.7. Alguns exemplos de funções de hash são MD5 e SHA1, SHA256 e SHA512.

As funções de *hash* também são utilizadas para obtenção do HMAC (*Hash-based Message Authentication Code* - Código de Autenticação de Mensagem com base na função Hash) de uma mensagem. O HMAC é uma construção utilizada para calcular o MAC (*Message Authentication Code*) envolvendo uma função de *hash* e uma chave secreta. A utilização do MAC está relacionada com a verificação de integridade e autenticidade.

2.7 Assinatura Digital

Assinatura digital é uma ferramenta utilizada para garantir que uma determinada mensagem veio de um determinado remetente e que seu conteúdo não foi modificado. Para tanto, suponha-se, por exemplo, que Alice envia um documento para Beto; e Beto por sua vez precisa ter certeza que este documento foi enviado por Alice e que não foi modificado depois que Alice o enviou.

Para ter um documento assinado digitalmente por Alice, tem necessariamente que ocorrer um conjunto de procedimentos:

- Alice calcula o *hash* do documento e em seguida realiza um cálculo com sua chave privada para obter a assinatura digital;
- Alice envia assinatura digital anexada com a mensagem para Beto;
- Beto calcula o *hash* do documento e em seguida realiza um cálculo entre a chave pública de Alice e a assinatura que recebeu em anexo;
- Se o resultado desse cálculo coincidir com o *hash* da mensagem recebida, esta é íntegra e o remetente é conhecido.

A Figura 2.5 ilustra este cenário.

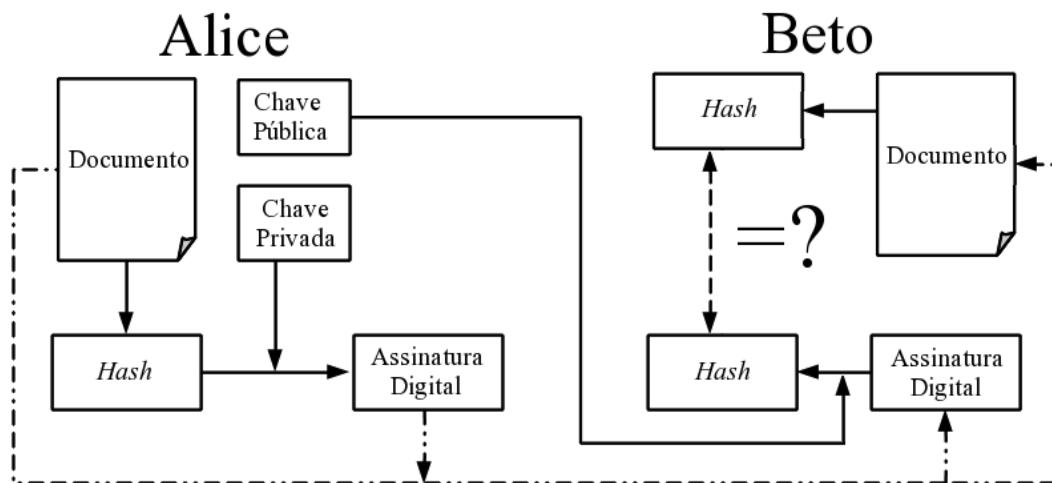


Figura 2.5: Procedimentos para assinar digitalmente um documento

Capítulo 3

O Cartão MIFARE Classic

Este capítulo aborda as características técnicas apresentadas na *Especificação Funcional do Cartão MIFARE Classic* (NXP, 2008), bem como alguns detalhes técnicos não revelados pela NXP, que foram posteriormente descobertos por pesquisadores, através de técnicas de engenharia reversa.

Por volta de 1995, a NXP Semiconductors (na época Philips) introduziu no mercado a família de cartões inteligentes MIFARE. A família é composta de quatro versões: *Ultralight*, *Classic*, *DES-Fire* e *SmartMX*, porém a versão *Classic* é a mais amplamente utilizada, disponível em diferentes capacidades de armazenamento e com chaves de 48 bits.

3.1 Estrutura Lógica

O cartão MIFARE classic é um cartão de memória EEPROM¹ com algumas funcionalidades de proteção (NXP, 2008). A unidade básica de memória é um bloco de 16 bytes. Na versão de 1K, cada 4 blocos são agrupados em um setor, totalizando 16 setores. Na versão de 4K os 32 primeiros setores são de 4 blocos e os demais são de 16 blocos. A Figura 3.1 mostra a estrutura lógica da versão de 1K.

Número do Setor	Número do Bloco	Conteúdo (16 bytes)							
00	00	BCC, UID, Fabricante (somente Leitura)							
	01. Dados/Valor	Dados ou Valor							
	02. Dados/Valor	Dados ou Valor							
	03. Trailer	Chave A	Condições de Acesso	U	Chave B				
01	04. Dados/Valor	Dados ou Valor							
	05. Dados/Valor	Dados ou Valor							
	06. Dados/Valor	Dados ou Valor							
	07. Trailer	Chave A	Condições de Acesso	U	Chave B				
:									
15	60. Dados/Valor	Valor	Valor	Valor	00	FF	00	FF	
	61. Dados/Valor	Valor	Valor	Valor	00	FF	00	FF	
	62. Dados/Valor	Dados ou Valor							
	63. Trailer	Chave A	Condições de Acesso	U	Chave B				

Figura 3.1: Estrutura Lógica do cartão MIFARE Classic de 1K

¹Electrically-Erasable Programmable Read-Only Memory

3.2 Bloco do Fabricante

O primeiro bloco do primeiro setor não permite modificação, ou seja, é somente leitura. Contém nos 4 primeiros bytes o *UID*, um identificador único do cartão; no byte subsequente contém o *BCC*, um verificador de integridade que faz XOR nos bytes do *UID*. Os demais bytes do bloco contém as informações do fabricante. A figura 3.2 ilustra a disposição dos dados no bloco do fabricante.

Campos	UID	BCC	Dados do Fabricante
Posição dos Bytes	0	4 5	15

Figura 3.2: Bloco 0, Setor 0 - Dados do Fabricante

3.3 Bloco *Trailer* do Setor

Este bloco é o último de cada setor, conforme visto anteriormente na Figura 3.1. Antes de realizar qualquer operação na memória do cartão, o leitor deve antes se autenticar no setor onde se encontra(m) o(s) bloco(s) de interesse. As condições de acesso podem ser definidas para cada bloco individualmente.

Neste bloco estão gravadas duas chaves de 6 bytes cada uma. A chave A é obrigatória, não é passível de leitura. Dependendo dos bits que definem as Condições de Acesso (CA) do setor, a chave B é opcional, podendo ser passível de leitura ou não. As condições de acesso são gravadas em 3 bytes. Existe um byte livre denominado U que pode ser utilizado para armazenamento de dados. Quando a chave B não for utilizada, a sua área poderá ser destinada para armazenamento de dados. A figura 3.3 ilustra a disposição dos dados no bloco *trailer* do setor.

Chave A	CA	U	Chave B (opcional)
0	6	9 10	15

Figura 3.3: Bloco *Trailer* do Setor

3.4 Bloco de Dados

No MIFARE *Classic* de 1K, o armazenamento de dados é feito nos três primeiros blocos de cada setor, exceto no primeiro setor que tem o primeiro bloco ocupado pelos dados do fabricante. Qualquer modificação de dados em qualquer bloco, deve ser precedida de uma autenticação no setor.

Além da leitura e escrita, o cartão MIFARE tem um formato específico para lidar com valores. Este tipo de configuração é denominado *Value Block* (Bloco de Valor), permitindo detecção e correção de erros, além de possibilitar o uso de backups. Por questões de segurança e integridade de dados, os 12 primeiros bytes consistem na representação de um valor inteiro de 4 bytes, sendo que o valor é armazenado 3 vezes. Duas vezes no formato normal (nos bytes 0-3 e 8-11) e uma vez com os bits invertidos (nos bytes 4-7). Nos 4 bytes restantes é mantido o endereçamento do bloco de 1 byte, repetido 4 vezes, duas no formato normal e duas com os bits invertidos. A figura 3.4 ilustra a disposição dos dados de um bloco de valor.

Valor	Valor	Valor	End	End	End	End
0	4	8	12	13	14	15

Figura 3.4: Bloco de Valor

3.5 Condições de Acesso

As condições de acesso para cada bloco de dados e para o bloco *trailer* do setor são definidas por 3 bits que estão armazenados de forma normal e com os bits invertidos no bloco *trailer* do setor. Os bits controlam as permissões de acesso à memória usando a chave A e a chave B. A cada acesso na memória do cartão é feita uma verificação das condições de acesso. Se ocorrer uma violação do formato, o bloco inteiro é bloqueado de maneira irreversível. A figura 3.5 ilustra a disposição dos bits nas condições de acesso.

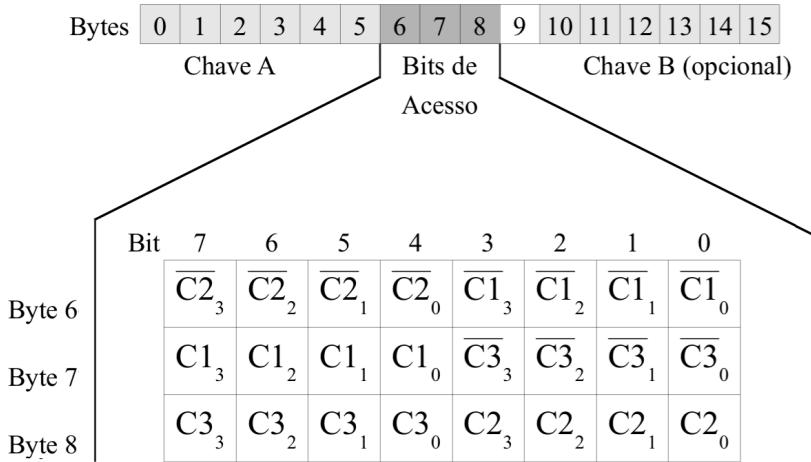


Figura 3.5: Disposição dos Bits das Condições de Acesso

Por questão de clareza, as próximas tabelas sobre condições de acesso, os bits de acesso serão mencionados somente no modo normal (sem inversão dos bits). A tabela 3.1 mostra os bits das condições de acesso associados a cada bloco.

Bits de Acesso	Comandos Válidos	Bloco	Descrição
$C1_3C2_3C3_3$	leitura, escrita	→ 3	setor <i>trailer</i>
$C1_2C2_2C3_2$	leitura, escrita, incremento, decremeno, transferência, restauração	→ 2	bloco de dados
$C1_1C2_1C3_1$	leitura, escrita, incremento, decremeno, transferência, restauração	→ 1	bloco de dados
$C1_0C2_0C3_0$	leitura, escrita, incremento, decremeno, transferência, restauração	→ 0	bloco de dados

Tabela 3.1: Condições de acesso

3.6 Condições de Acesso para o Bloco *Trailer* do Setor

Dependendo dos bits de acesso para o bloco *trailer* do setor (último bloco do setor) o acesso de leitura e escrita para as chaves e para os bits de acesso são especificados como: “nunca”, “chave A”, “chave B” e “chave A|B” (chave A ou B).

Nos cartões novos, as condições de acesso do bloco *trailer* do setor e a chave A são configuradas para transporte. Uma vez que a chave B pode ser lida na configuração de transporte, novos cartões devem ser autenticados com a chave A. A tabela 3.2 mostra as possíveis condições de acesso para o bloco *trailer* do setor.

Bits de Acesso			Condições de acesso para						Pode ser lida
			Chave A		Bits de Acesso		Chave B		
C1	C2	C3	leitura	escrita	leitura	escrita	leitura	escrita	
0	0	0	nunca	chave A	chave A	nunca	chave A	chave A	chave B
0	1	0	nunca	nunca	chave A	nunca	chave A	nunca	chave B
1	0	0	nunca	chave B	chave A B	nunca	nunca	chave B	
1	1	0	nunca	nunca	chave A B	nunca	nunca	nunca	
0	0	1	nunca	chave A	chave A	chave A	chave A	chave A	chave B
0	1	1	nunca	chave B	chave A B	chave B	nunca	chave B	
1	0	1	nunca	nunca	chave A B	chave B	nunca	nunca	
1	1	1	nunca	nunca	chave A B	nunca	nunca	nunca	

Tabela 3.2: Condições de Acesso para o bloco Trailer do Setor

As linhas de cor cinza na Tabela 3.2 indicam que a região da chave B é passível de leitura, podendo ser utilizada para armazenamento de dados.

3.7 Condições de Acesso para Blocos de Dados

Dependendo dos bits de acesso para os blocos de dados (0-2) o acesso de leitura e escrita é especificado como ‘nunca’, ‘chave A’, ‘chave B’ e ‘chave A|B’ (chave A ou B). As configurações de bits definem a aplicação e os comandos aplicáveis correspondentes.

- Bloco de leitura e escrita: são permitidas operações de leitura e escrita
- Bloco de valor: permite operações de incremento, decremento, transferência e restauração. Em um caso particular “001”, somente leitura e decremento é possível. No outro caso “110”, a escrita é possível usando a chave B.
- Bloco do fabricante: a condição somente para leitura não é afetada pela configuração dos bits de acesso.
- Gerenciamento de chaves: Na configuração de transporte, a chave A deve ser usada para autenticação.

Bits de Acesso			leitura	escrita	Incremento	Decreimento Transerência Restauração	Aplicação	
C1	C2	C3						
0	0	0	chave A B	chave A B	chave A B	chave A B	config. de transporte	
0	1	0	chave A B	nunca	nunca	nunca	bloco de leitura/escrita	
1	0	0	chave A B	B	nunca	nunca	bloco de leitura/escrita	
1	1	0	chave A B	B	chave A B	chave A B	bloco de valor	
0	0	1	chave A B	nunca	chave A B	chave A B	bloco de valor	
0	1	1	B	B	nunca	nunca	bloco de leitura/escrita	
1	0	1	B	nunca	nunca	nunca	bloco de leitura/escrita	
1	1	1	nunca	nunca	nunca	nunca	bloco de leitura/escrita	

Tabela 3.3: Condições de acesso para bloco de dados

Se a chave B puder ser lida no bloco *trailer* do setor, ela não poderá ser utilizada para autenticação. As linhas em destaque na cor cinza na Tabela 3.3 indicam este estado. Se um leitor tentar se autenticar em qualquer setor com uma chave B usando as condições de acesso das linhas destacadas em cinza, o cartão recusará qualquer acesso subsequente à memória após a autenticação.

3.8 Acesso à Memória

Antes que qualquer operação possa ser realizada, o cartão deve estar autenticado. As operações possíveis na memória para um determinado bloco, depende da chave e das condições de acesso armazenadas no bloco *trailer* do setor. A especificação do cartão MIFARE *Classic* define seis operações que podem ser realizadas em um bloco de dados. A tabela 3.4 mostra esse conjunto de operações. Adicionalmente, a figura 3.6 ilustra a sequência de procedimentos necessários para utilização dessas operações.

Operação	Descrição	Tipo de Bloco Válido
Leitura	lê um bloco de memória	leitura/escrita, valor e <i>trailer</i> do setor
Escrita	escreve um bloco de memória	leitura/escrita, valor e <i>trailer</i> do setor
Incremento	incrementa o conteúdo de um bloco e armazena o resultado no registro interno de dados	valor
Decremento	decrementa o conteúdo de um bloco e armazena o resultado no registro interno de dados	valor
Transferência	grava o conteúdo do registro interno de dados para o bloco	valor
Restauração	lê conteúdo de um bloco copiando o conteúdo lido para dentro do registro interno de dados	valor

Tabela 3.4: Operações na Memória

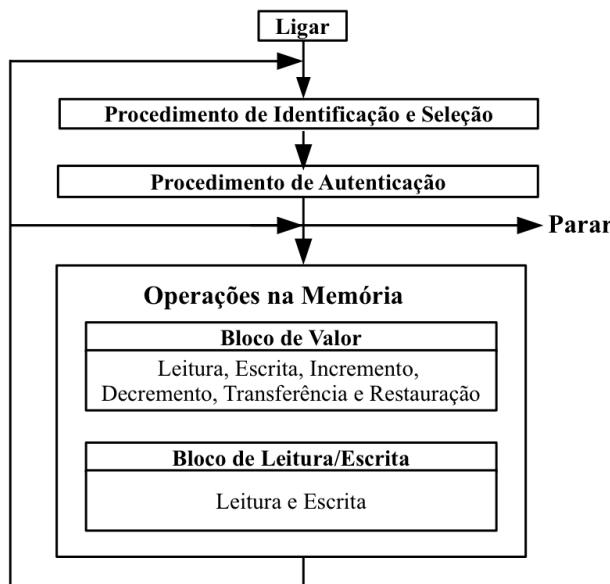


Figura 3.6: Sequência de procedimentos para acesso à memória

3.9 Comunicação

A comunicação utilizada no MIFARE *Classic* segue a terceira parte do padrão ISO 14443A. Os comandos são inicializados pelo leitor e controlados por uma Unidade de Controle Digital, de acordo com as condições de acesso válidas para o setor correspondente.

A Figura 3.7 mostra a comunicação utilizada em uma transação típica com o cartão, começando no topo, movendo-se de estágio para estágio e de cima para baixo, conforme mostrado pelas setas com linhas sólidas. Entretanto, em certos estágios a comunicação pode reiniciar conforme indicado pelas setas com linhas pontilhadas mais a esquerda.

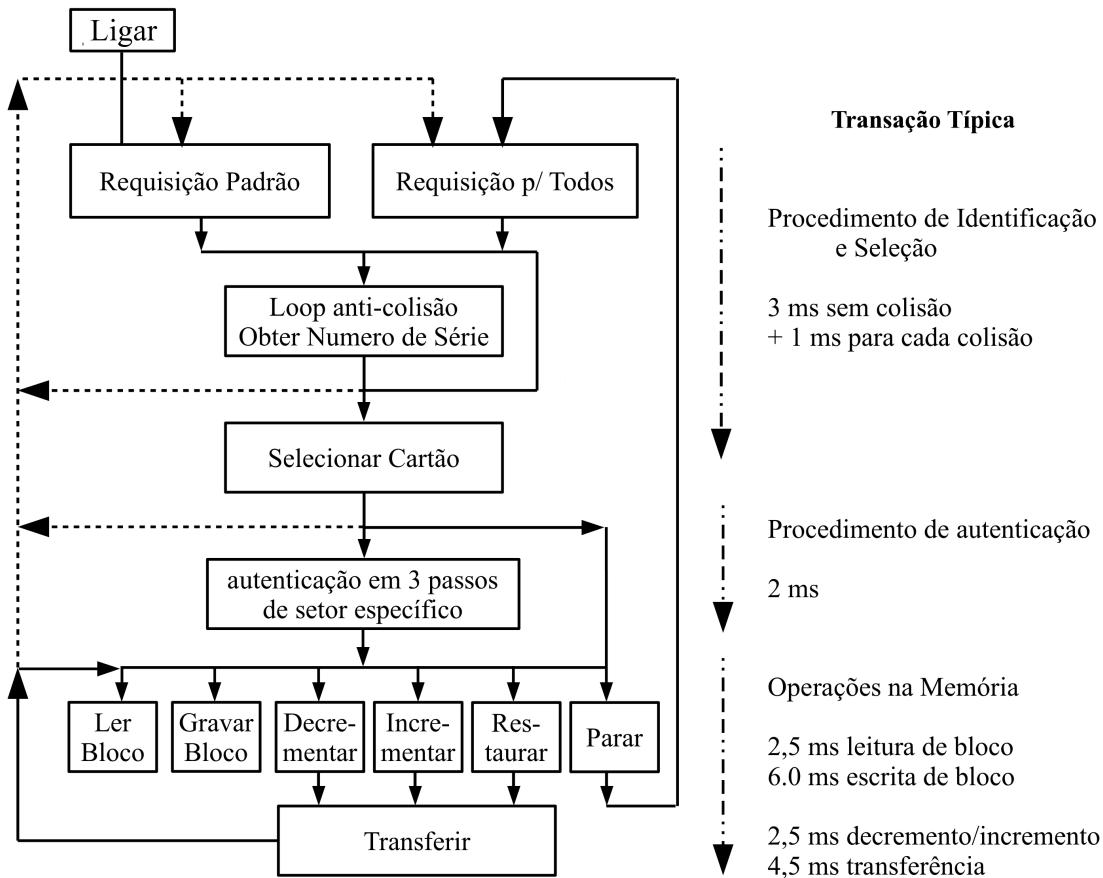


Figura 3.7: Fluxo da Comunicação utilizada no MIFARE Classic

3.9.1 Requisição Padrão/Todos

Quando um cartão é aproximado do campo eletromagnético gerado pelo leitor, ele recebe a energia suficiente para se ligar (*Power on Reset*). Então o leitor começa a enviar comandos de Requisição Padrão² (REQA) ou Requisição para Todos³ (WUPA), daí o cartão responde ao leitor com código compatível com tags do tipo A⁴ (ATQA) de acordo com o padrão (ISO 14443).

²Do inglês, *REQuest command, Type A*

³Do inglês, *Wake UP command, Type A*

⁴Do inglês, *Answer To Request of Type A*

3.9.2 Loop Anti-Colisão

Neste estágio o número de série do cartão é lido. Se houver mais de um cartão no raio de alcance do leitor, pode ocorrer colisões. Uma colisão ocorre quando o leitor envia um comando de anti-colisão e pelo menos dois cartões respondem simultaneamente. O leitor percebe que ocorreu uma colisão e envia outro comando de anti-colisão até que não ocorra mais colisões. Os cartões serão distinguidos pelos seus UIDs permitindo que um deles seja selecionado para uma eventual transação. Os cartões não selecionados voltam para o modo *standby* e ficam aguardando uma nova requisição de comando.

3.9.3 Seleção do Cartão

Com o comando de seleção do cartão (SEL), o leitor seleciona um cartão individual para autenticação e operações na memória. O cartão retorna o código de resposta para o comando de seleção⁵ (SAK). Através das respostas SAK e ATQA é possível determinar o tipo de cartão. A Tabela 3.5 mostra alguns exemplos conhecidos.

Fabricante	Produto	ATQA	1º byte do SAK
NXP	MIFARE Classic Mini	04 00	09
	MIFARE Classic 1K	04 00	08
	MIFARE Classic 4K	02 00	18
	MIFARE Ultralight	44 00	00
	MIFARE DESFire	44 03	20
	MIFARE DESFire EV1	44 03	20
Infineon	MIFARE Classic 1K	04 00	88

Tabela 3.5: Tipos de cartões identificados pelas respostas ATQA/SAK

3.9.4 Integridade de Dados

Os seguintes mecanismos são implementados na comunicação entre o leitor e o cartão para certificar a confiabilidade na transmissão de dados:

- Verificação de redundância cíclica⁶ (CRC) para cada bloco;
- Bits de paridade para cada byte;
- *Bit count checking* (BCC);
- Codificação de bits para distinguir entre “0”, “1” e nenhuma informação;
- Canal de monitoramento (sequência do protocolo e análise do fluxo de bits).

⁵Do inglês, *Select Acknowledge reply*

⁶Do inglês, *Cyclic redundancy check*

3.9.5 Autenticação em Três Passos

Depois da seleção do cartão, o leitor especifica o local da memória que será acessada e usa a chave correspondente para o procedimento de autenticação descritos a seguir:

1. O leitor especifica o setor a ser acessado, escolhendo a chave A ou B.
2. O cartão lê a chave secreta e acessa as condições de acesso do bloco *trailer* do setor. Então o cartão envia um número aleatório (*nonce*⁷) como um desafio para o leitor (passo 1).
3. O leitor calcula a resposta usando a chave secreta e a entrada de dados adicional. A resposta, junto com o desafio do leitor, é transmitida para o cartão (passo 2).
4. O cartão verifica a resposta do leitor com o desafio que enviou anteriormente para ele, então calcula a resposta do desafio para transmiti-la ao leitor (passo 3)
5. O leitor verifica a resposta do cartão, comparando-a com o desafio que foi enviado.

Após a transmissão do primeiro desafio gerado aleatoriamente, toda a comunicação entre o leitor e o cartão é criptografada. Além disso, todas as operações na memória do cartão são criptografadas. A Figura 3.8 ilustra esse processo de autenticação passo a passo.

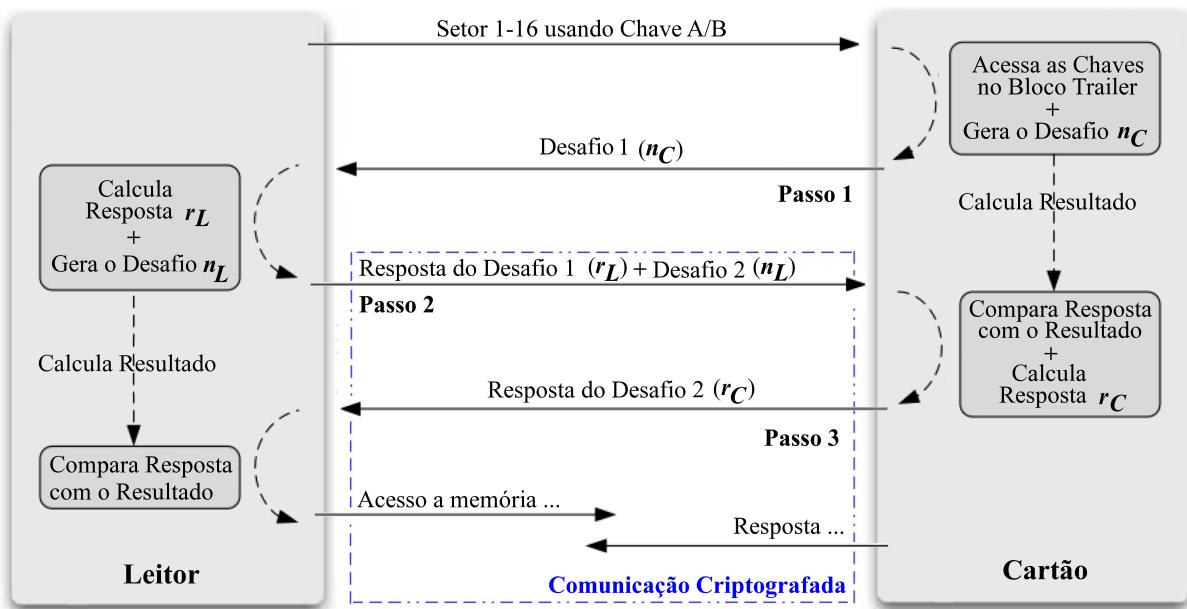


Figura 3.8: Autenticação em três passos

3.9.6 Parada

A qualquer momento depois que o cartão for selecionado, o leitor poderá paralisar toda a comunicação enviando um comando de parada HLTA⁸. Assim, o cartão irá entrar em estado de *standby*, voltando somente pelo comando WUPA⁹.

⁷Usado uma única vez - Do inglês, *number used once*

⁸HALT Command, Type A

⁹Wake UP command, Type A

3.9.7 Conjunto de Comandos

O MIFARE Classic possui um protocolo de alto nível que diverge da parte 4 do padrão ISO 14443. Através de experimentos Gans *et al.* (2008) eles descobriram que os comandos entre o cartão e o leitor utilizavam os mesmos códigos, e que todos os comandos enviados pelo leitor continham um byte de comando, um byte de parâmetro e dois bytes de CRC. Então, utilizando técnicas de engenharia reversa recuperaram todo o conjunto de comandos apresentados na Tabela 3.6. Além disso, também descobriram que o cartão respondia ACK/NACK nos comandos de escrita em blocos de dados, bem como nos comandos de manipulação de valor tais como incremento, decremento, transferência e restauração.

<i>LEITOR</i>	<i>CARTÃO</i>	<i>LEITOR</i>	<i>CARTÃO</i>
Autenticação			
60 YY* Usando a Chave A	nonce de 4 bytes	resposta de 8 bytes	resposta de 4 bytes
61 YY* Usando a chave B	nonce de 4 bytes	resposta de 8 bytes	resposta de 4-bytes
Dados			
30 YY* Leitura	16 bytes e dados*		
A0 YY* Escrita	ACK/NACK	16 bytes de dados*	
Valor			
C0 YY* Decremento	ACK/NACK	valor de 4 bytes*	Transferência
C1 YY* Incremento	ACK/NACK	valor de 4 bytes*	Transferência
C2 YY* Restauração	ACK/NACK		
Outros			
50 00* Parada (halt)		YY = endereço do bloco * = seguidos de 2 bytes de CRC	

Respostas do Cartão (ACK/NACK)

- A (1010) ACK
- 4 (0100) NACK, não permitido
- 5 (0101) NACK, erro de transmissão

Tabela 3.6: Conjunto de comandos utilizados entre o leitor e o cartão

A Tabela 3.7 mostra um exemplo de comunicação entre o leitor e o cartão. Veremos mais adiante no Capítulo 6, através de um log como esse será possível realizar um tipo de ataque bastante eficiente com objetivo de recuperar a chave secreta utilizada na autenticação.

Seq.	Quem	Bytes	Comando
1	Leitor	26	REQA (WPA se tiver no estado <i>standby</i>)
2	Cartão	04 00	ATQA
3	Leitor	93 20	AC
4	Cartão	2A 69 8D 43 8D	UID + BCC
5	Leitor	93 70 2A 69 8D 43 8D 52 55	SEL
6	Cartão	08 B6 DD	SAK
7	Leitor	60 04 D1 3D	Autenticação do bloco 4 usando chave A
8	Cartão	3B AE 03 2D	Nonce do Cartão
9	Leitor	C4 94 A1 D2 6E 96 86 42	Nonce do Leitor + resposta
10	Cartão	84 66 05 9E	Resposta do Cartão
11	Leitor	50 00 57 CD	Parada

Tabela 3.7: Exemplo de um log da comunicação entre Leitor e Cartão

Nesse capítulo, foram abordados a especificação funcional do Cartão MIFARE Classic, bem

como detalhes do protocolo obtidos por engenharia reversa. No próximo capítulo a cifra Crypto-1 e o Gerador Pseudo-Aleatório utilizados pelo MIFARE Classic serão apresentados em detalhes.

Capítulo 4

A Cifra Crypto-1

Este capítulo descreve em detalhes a cifra Crypto-1, um algoritmo proprietário desenvolvido pela NXP Semiconductors para execução em hardware objetivando desempenho. Utiliza chaves de 48 bits e foi projetada especificamente para o chip do cartão MIFARE Classic. Aqui o objetivo é apresentar os resultados dos trabalhos de engenharia reversa que permitiram a recuperação da cifra Crypto-1.

Parte da segurança do cartão era confiada no segredo da cifra Crypto-1. Esta prática é conhecida como “segurança por obscuridade”. De acordo com o Princípio de Kerckhoffs (1883), um criptosistema não deve confiar no sigilo do algoritmo para a sua própria segurança. É bastante aceito que um algoritmo fechado um dia poderá vazrar através de seus colaboradores ou será recuperado por engenharia reversa. De fato, por volta de 2004 houve rumores na NXP de vazamento de informações confidenciais ou engenharia reversa para fabricantes asiáticos não autorizados (Mayes e Cid, 2010).

De acordo com Mayes e Cid (2010), apesar de ter sido mantido em segredo pela NXP, o MIFARE Classic foi aberto publicamente após o trabalho de dois grupos de pesquisadores, de forma mais ou menos independente e com diferentes abordagens. Enquanto Nohl e Plötz (2007); Nohl *et al.* (2008), estudantes de doutorado em Ciência da Computação na *Virginia University*, realizaram a engenharia reversa de parte do algoritmo, reconstruindo o CI¹ com a ajuda de um microscópio eletrônico, utilizando técnicas de engenharia reversa para implementações no silício; O grupo de pesquisadores da *Radboud University Nijmegen* Gans *et al.* (2008); Garcia *et al.* (2008) na Holanda realizaram a recuperação de todo o algoritmo através da espionagem da comunicação entre o leitor e o cartão, permitindo uma implementação em software denominada *Crapto-1*. Tan (2009) fez uma síntese destes trabalhos envolvendo a cifra Crypto-1; essa síntese também será usada para explicar os detalhes da cifra Crypto-1.

¹Círcuito Integrado

4.1 Cifra de Fluxo

A Crypto-1 é essencialmente uma *cifra de fluxo*², tendo como peça principal o gerador de bits pseudo-aleatórios que produz os bits do *keystream* gerados por um LFSR de 48 bits, com o polinômio gerador $x_{48} + x_{43} + x_{39} + x_{38} + x_{36} + x_{34} + x_{33} + x_{31} + x_{29} + x_{24} + x_{23} + x_{21} + x_{19} + x_{13} + x_9 + x_7 + x_6 + x_5 + 1$ e uma função filtro não linear (Nohl *et al.*, 2008). A Figura 4.1 mostra o LFSR com os bits rotulados de 0 a 47, e o filtro gerador composto pelas funções f_a , f_b e f_c .

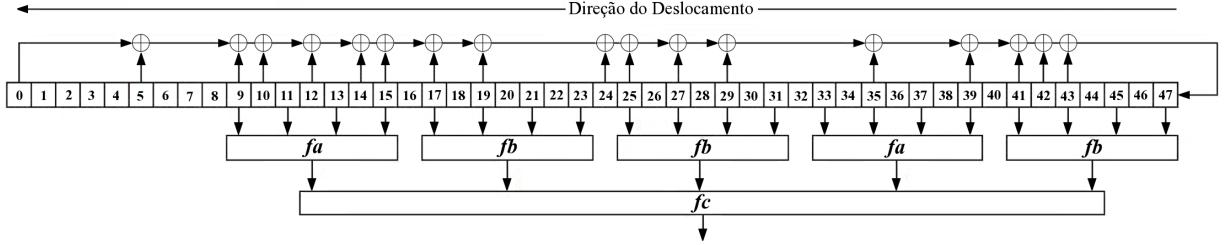


Figura 4.1: Diagrama com LFSR e as funções filtro da Cripto-1

4.2 Geração do *keystream*

A cada ciclo de *clock*³, o filtro gerador não linear obtém 20 bits do LFSR (conforme Figura 4.1) e gera um bit do *keystream*. O LFSR desloca um bit para a esquerda, descartando o bit mais significativo à esquerda (o bit zero) e insere um novo bit gerado à direita, usando a função geradora, também conhecida como a função de *feedback*. Usando a notação x_n para o valor do bit (0 ou 1) na posição n e \oplus a função lógica OU-exclusivo (XOR), então a função *feedback* é definida por:

$$L(x_0x_1\dots x_{47}) := x_0 \oplus x_5 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{19} \oplus x_{24} \oplus x_{25} \oplus x_{27} \oplus x_{29} \oplus x_{35} \oplus x_{39} \oplus x_{41} \oplus x_{42} \oplus x_{43} \quad (4.1)$$

Os bits escolhidos como entradas na função de *feedback* são mostradas na Figura 4.1 pelas setas de cima. O filtro gerador que gera um bit do *keystream* a cada ciclo de *clock*, pode ser definido como uma combinação de três subfunções lógicas f_a , f_b e f_c abaixo. Usando a mesma notação acima e introduzindo os operadores lógicos \vee como OU (OR) e \wedge como E (AND), pode-se definir o gerador e as três subfunções f_a , f_b e f_c .

$$f(x_0x_1\dots x_{47}) := f_c(f_a(x_9, x_{11}, x_{13}, x_{15}), f_b(x_{17}, x_{19}, x_{21}, x_{23}), f_b(x_{25}, x_{27}, x_{29}, x_{31}), f_a(x_{33}, x_{35}, x_{37}, x_{39}), f_b(x_{41}, x_{43}, x_{45}, x_{47})) \quad (4.2)$$

$$f_a(a, b, c, d) := ((a \vee b) \oplus (a \wedge d)) \oplus (c \wedge ((a \oplus b) \vee d)) \quad (4.3)$$

$$f_b(a, b, c, d) := ((a \wedge b) \vee c) \oplus ((a \oplus b) \wedge (c \vee d)) \quad (4.4)$$

$$f_c(a, b, c, d, e) := (a \vee ((b \vee e) \wedge (d \oplus e))) \oplus ((a \oplus (b \wedge d)) \wedge ((c \oplus d) \vee (b \wedge e))) \quad (4.5)$$

Um ponto importante a ser observado é que as entradas para a função de filtro são espaçadas em números ímpares, e que os bits de estado se iniciam a partir do bit 9. Estas fraquezas serão exploradas mais adiante nas Sessões 5.3 e 5.2.

²As Cifras de Fluxo foram apresentados na Seção 2.4

³Unidade de tempo na qual são executadas as instruções

4.3 Gerador de números pseudo-aleatórios

Retomando os conceitos apresentados na sub Seção 3.9.5, o protocolo de autenticação do MIFARE Classic utiliza um mecanismo de autenticação mútua do tipo desafio-resposta. O desafio é um nonce de 32 bits que é gerado por um gerador de números pseudo-aleatórios. Nohl e Plötz (2007) explicam que este gerador é um LFSR de 16 bits com um circuito separado daquele LFSR de 48 bits utilizado na cifra. De imediato tem-se uma fraqueza em relação ao tamanho do LFSR - mesmo usando um nonce de 32 bits, o LFSR que gera o nonce usando somente a metade dos bits. Veja mais detalhes adiante na Seção 5.1 a respeito desta fraqueza.

A cada ciclo de clock, o LFSR desloca um bit para a esquerda, descartando o bit mais significativo à esquerda (bit zero) e inserindo um bit mais à direita usando a função de *feedback*. Novamente usando x_n para indicar o valor do bit (0 ou 1) na posição n e \oplus para a função lógica OU-exclusivo (XOR). A função de feedback é definida por:

$$L(x_0x_1\dots x_{15}) := x_0 \oplus x_2 \oplus x_3 \oplus x_5 \quad (4.6)$$

Além disso, se for definida uma sequência de um LFSR de 32 bits $x_0x_1\dots x_{31}$ consistindo deste LFSR de 16 bits, a próxima sequência de 32 bits pode ser definida usando uma função *suc* onde:

$$suc(x_0x_1\dots x_{31}) := x_1x_2\dots x_{31}L(x_{16}x_{17}\dots x_{31}) \quad (4.7)$$

$$suc^n(x_0x_1\dots x_{31}) := suc^{(n-1)}(suc(x_0x_1\dots x_{31})) \quad (4.8)$$

Na autenticação em três passos com desafio-resposta, a resposta para um desafio é obtida através de uma função pré-determinada, tendo o desafio como a sua entrada. A regra da função sucessora é pré-determinada precisamente. A próxima Seção explora em detalhes este protocolo de autenticação mútua com desafio-resposta.

4.4 Autenticação

Na documentação oficial da NXP para o chip do MIFARE Classic, a autenticação entre o leitor e o cartão é descrita como uma autenticação mútua em três passos. O leitor primeiro solicita para ser autenticado em um bloco em particular. Então ocorrem três passos.:

1. O cartão envia um desafio (nonce) n_C para o leitor.
2. Desta mensagem em diante a comunicação é criptografada (a Seção seguinte explica em detalhes a inicialização da cifra). O leitor responde de forma criptografada um nonce n_L e uma resposta r_L para o desafio do passo 1.
3. Se o leitor respondeu corretamente, o cartão irá responder de forma criptografada uma resposta r_C . Caso contrário o cartão não irá responder.

Garcia *et al.* (2008) descobriram os detalhes do protocolo de autenticação e a inicialização da cifra. As respostas do leitor (r_L) e do cartão (r_C) são ambas derivadas do nonce gerado pelo cartão com as seguintes relações:

$$r_L = suc^{64}(n_C) \quad (4.9)$$

$$r_C = suc^{96}(n_C) \quad (4.10)$$

Assim, é possível entender como a função suc é usada para calcular a resposta do desafio inicial. Verificando a corretude da r_L depois decriptografando seu texto criptografado, o cartão é capaz de verificar se o leitor possui a mesma chave. Se r_L for incorreto, o cartão deixa de comunicar com o leitor. Segundo essa mesma ideia, o leitor irá verificar a corretude de r_C .

4.5 Inicialização da Cifra e Geração do Keystream

A cifra deve ser inicializada tanto no leitor como no cartão através de uma Seção de autenticação, para posteriormente ser utilizada para criptografar e decriptografar. Os parâmetros necessários são uma chave K de 48 bits de um determinado setor do cartão, u como o UID do cartão e os nonces do cartão e do leitor, n_C e n_L . Primeiramente a chave de 48 bits é inserida no LFSR, ficando o bit mais significativo na posição x_0 , o próximo bit é colocado na posição x_1 e assim sucessivamente. Então o $n_C \oplus u$ é deslocado em conjunto com o *feedback* bit a bit, e finalmente o n_L é deslocado junto com o *feedback* bit a bit. Tanto o cartão quanto o leitor devem chegar no mesmo estado da cifra ao final da inicialização para que a autenticação seja estabelecida. A Figura 4.2 ilustra a inicialização.

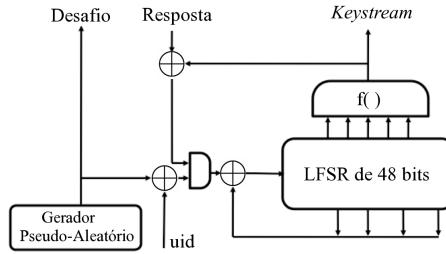


Figura 4.2: Inicialização da Cifra Crypto-1

A definição deste processo pode ser formalizado como:

Definição 4.5.1. Inicialização da Cifra - Seja uma chave K de setor, um nonce n_C , um identificador único u , um nonce do leitor n_L , onde

$$K := k_0k_1k_2\dots k_{47} \quad (4.11)$$

$$n_C = n_C, 0n_C, 1\dots n_C, 31 \quad (4.12)$$

$$u := u_0u_1\dots u_31 \quad (4.13)$$

$$n_L := n_L, 0n_L, 1\dots n_L, 31 \quad (4.14)$$

O estado interno da cifra no tempo i é representado por S_i e contém o valor dos 48 bits da cifra no tempo i definido como:

$$S_i := s_is_{(i+1)}\dots s_{(i+47)} \quad (4.15)$$

Sendo s_i definido pelas seguintes relações:

$$s_i := k_i, \forall i \in [0, 47] \quad (4.16)$$

$$s_{(i+48)} := L(s_i, s_{(i+1)}, \dots, s_{(i+47)}) \oplus n_{C,i} \oplus u_i, \forall i \in [0, 31] \quad (4.17)$$

$$s_{(i+80)} := L(s_{(i+32)}, s_{(i+33)}, \dots, s_{(i+79)}) \oplus n_{L,i}, \forall i \in [0, 31] \quad (4.18)$$

$$s_{(i+112)} := L(s_{(i+64)}, s_{(i+65)}, \dots, s_{(i+111)}), \forall i \in \mathbb{N} \quad (4.19)$$

Definição 4.5.2. Geração do *Keystream* - De forma similar, pode-se definir o bit b_i do *keystream* no tempo i por:

$$b_i := f(s_i s_{(i+1)} \dots s_{(i+47)}), \forall i \in \mathbb{N} \quad (4.20)$$

Definição 4.5.3. Decriptografia do n_L , r_L e r_C - Usando a notação entre chaves {} para denotar dados criptografados tem-se:

$$\{n_{L,i}\} := n_{L,i} \oplus b_{(i+32)}, \forall i \in [0, 31] \quad (4.21)$$

$$\{r_{L,i}\} := r_{L,i} \oplus b_{(i+64)}, \forall i \in [0, 31] \quad (4.22)$$

$$\{r_{C,i}\} := r_{C,i} \oplus b_{(i+96)}, \forall i \in [0, 31] \quad (4.23)$$

Definição 4.5.4. Geração do *Keystream* - De forma similar, pode-se definir o bit b_i do *keystream* no tempo i por:

$$b_i := f(s_i s_{(i+1)} \dots s_{(i+47)}), \forall i \in \mathbb{N} \quad (4.24)$$

Definição 4.5.5. *Keystreams Especiais* - Além disso, tem-se um grupo de bits de *keystreams* usando a notação abaixo:

$$ks1 := b_{32} b_{33} \dots b_{63} \quad (4.25)$$

$$ks2 := b_{64} b_{65} \dots b_{95} \quad (4.26)$$

$$ks3 := b_{96} b_{97} \dots b_{127} \quad (4.27)$$

4.6 Bits de Paridade

O padrão (ISO 14443) especifica que cada byte enviado é seguido de um bit de paridade ímpar. O MIFARE Classic calcula paridade sobre o texto em claro ao invés do texto criptografado. Para cada palavra de 32 bits, tem-se 4 bits de paridade. Além disso, o bit de paridade é criptografado com o mesmo bit do *keystream* usado para criptografar o próximo bit do texto em claro.

Definição 4.6.1. *Paridade* - Definindo a paridade de bits p_j das mensagens n_L e r_L , e sua forma criptografada $\{p_j\}$ tem-se (o $\oplus 1$ no final implicando em paridade ímpar):

$$p_j := n_{L,8j} \oplus n_{L,(8j+1)} \oplus n_{L,(8j+7)} \oplus 1, \forall j \in [0, 3] \quad (4.28)$$

$$p_{(j+4)} := r_{L,8j} \oplus r_{L,(8j+1)} \oplus r_{L,(8j+7)} \oplus 1, \forall j \in [0, 3] \quad (4.29)$$

$$\{p_j\} := p_j \oplus b_{(8j+8)}, \forall j \in [0, 7] \quad (4.30)$$

Capítulo 5

Fraquezas Encontradas

Neste capítulo é apresentada uma síntese das principais fraquezas encontradas no algoritmo e protocolo, bem como as consequências. O conteúdo deste capítulo serve de referência para o Capítulo 6, onde serão abordados os ataques explorando essas fraquezas. Além disso, também pode ser referenciado no Capítulo 7 onde serão propostas as medidas de contorno.

5.1 Baixa entropia do gerador pseudo-aleatório

A baixa entropia das sequências geradas pelo PRNG (*Pseudo Random Number Generator - Gerador de Números Pseudo Aleatórios*) é uma das fraquezas mais críticas do MIFARE Classic. Os nonces de 32 bits utilizam um LFSR de 16 bits com o polinômio gerador $x^{16} + x^{14} + x^{13} + x^{11} + 1$, resultando em baixa entropia. Isso pode ser observado na Figura 5.1. De acordo com Garcia *et al.* (2009), em função do período do PRNG conter apenas 16 bits (65.535) e cada deslocamento gastar cerca de $9,44\mu\text{s}$, seu ciclo se completa em apenas 618ms. Nohl *et al.* (2008) explicam que leva cerca de 0,6 segundos para o estado interno do gerador gerar todos os 65.536 nonces possíveis.



Figura 5.1: Gerador de números pseudo-aleatórios usa apenas 16 bits

Como os nonces utilizados no processo de autenticação são gerados por esse gerador, o adversário pode facilmente esperar um número fixo de ciclos de *clock* a serem gastos, desde que o cartão se ligou até o momento que antecede a primeira requisição para o nonce, enviando dessa forma, sempre o mesmo nonce. Além disso, se o atacante puder controlar o cartão, ele saberá quando o ciclo do gerador se completou, podendo enviar o mesmo nonce toda vez (Garcia *et al.*, 2009).

5.2 Algoritmo gerador do *keystream* não usa os bits mais à esquerda

O algoritmo utilizado para geração do *keystream* expõe uma fraqueza criptográfica na cifra. Na figura 5.2 é possível perceber que os 9 primeiros bits da esquerda para a direita não são utilizados. Isso permite ao adversário realizar um ataque para obter o reverso da função geradora, objetivando “restaurar” o estado bit a bit do LFSR. Assim, é possível chegar ao estado inicial do LFSR através desse tipo de ataque realizado sucessivas vezes. [Garcia et al. \(2009\)](#) explicam que esta fraqueza permite que um atacante possa recuperar a chave secreta conhecendo apenas o UID u , o nonce do cartão n_C e o nonce do leitor criptografado $\{n_L\}$.

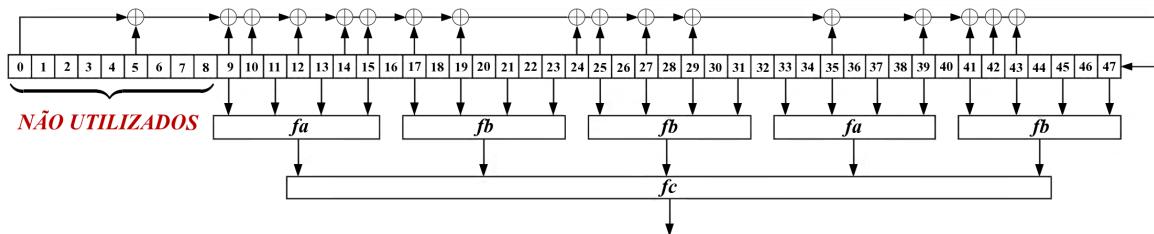


Figura 5.2: Bits mais à esquerda do LFSR não utilizados pelo filtro gerador

Partindo de um estado conhecido do LFSR, como por exemplo logo após a inserção do cartão no campo magnético do leitor, sabe-se que neste ponto o LFSR acabou de receber a inserção de $n_C \oplus u$ e subsequentemente n_L (conforme descrito na Seção 4.5). O bit mais à direita será descartado e o bit mais à esquerda será atribuído com um valor arbitrário r . Então pode-se usar o filtro gerador para calcular o bit do *keystream* que por sua vez será usado para criptografar o bit $n_L, 31$. Já que o bit r mais à esquerda que acabou de ser inserido não participa do filtro gerador, então não importa o valor contido em r . Com o bit do *keystream* obtido de maneira simples, realizando um XOR com $\{n_L, 31\}$ para obter $n_L, 31$, é possível atribuir corretamente r usando a função de *feedback* do LFSR (conforme equação 4.18). Repetindo este procedimento por 31 vezes será possível recuperar o estado do LFSR antes que o n_L seja inserido, e mais 32 vezes para recuperar a chave secreta antes que $(u \oplus n_T)$ seja inserido ([Garcia et al., 2008](#)).

5.3 Geração do *keystream* somente com bits ímpares

A geração do *keystream* somente com bits ímpares é outra fraqueza criptográfica na cifra. A figura 5.3, bem como a equação 4.1 mostram que as entradas da função filtro são apenas bits ímpares, extraídos do estado corrente do LFSR. Utilizando técnica de divisão e conquista é possível focar somente nos bits ímpares para a recuperação do *keystream*, facilitando trabalhos de criptanálise na cifra.

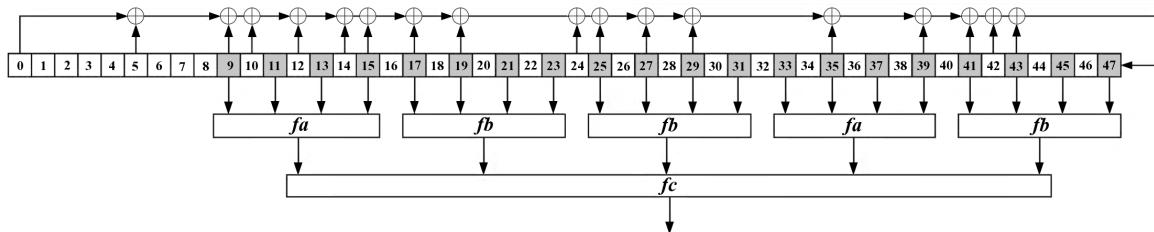


Figura 5.3: Geração do *keystream* utiliza apenas bits ímpares do LFSR

5.4 Bits de paridade e reuso do *one-time pad*

Os bits de paridade expõe uma fraqueza no protocolo de comunicação. O padrão ISO 14443 define que todo byte enviado deve ser seguido de um bit de paridade. Entretanto o MIFARE Classic calcula o bit de paridade do texto em claro ao invés do texto criptografado. Garcia *et al.* (2008) demonstram que o estado interno do LFSR não é deslocado quando o bit de paridade é criptografado. Através da figura 5.4 é possível observar que tanto o bit de paridade quanto o primeiro bit do texto em claro são criptografados com o mesmo bit do *keystream*. Conforme descrito na Seção 2.2, uma das propriedades do *one-time pad* é que ele não deve nunca ser reutilizado.

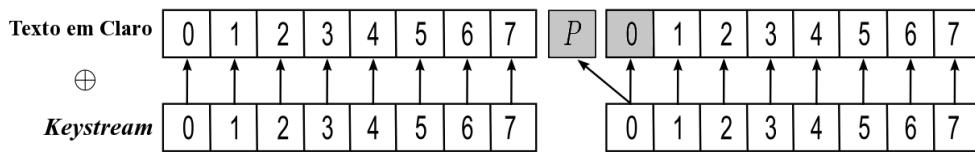


Figura 5.4: Criptografia dos bits de paridade

5.5 Exposição parcial do *keystream* nos bits de paridade

A exposição parcial do *keystream* nos bits de paridade é outra fraqueza no protocolo de comunicação. Durante o processo de autenticação, quando o leitor envia $\{n_L\}$ e $\{r_L\}$, o cartão verifica os bits de paridade antes de verificar a resposta do leitor r_L . Se pelo menos um dos oito bits estiver errado, o cartão não responde. Entretanto, se todos os bits estiverem corretos, mas a resposta r_L estiver errada, o cartão responde em 4 bits o código de erro 0x5 indicando erro de transmissão (Gans *et al.*, 2008). Além disso, este código de erro é enviado criptografado. Isso acontece mesmo quando o leitor não realizou a autenticação e não é capaz de decriptografar a mensagem. Combinando o código de erro 0x5 com sua versão criptografada, pode-se recuperar 4 bits do *keystream* (Garcia *et al.*, 2009).

5.6 Autenticações Aninhadas

Se um atacante conhece a chave de um único setor, existe uma vulnerabilidade que permite ao adversário recuperar mais chaves. Após a autenticação com essa chave, toda a comunicação entre o leitor e o cartão é cifrada. Um comando de autenticação subsequente para um novo setor também será enviado cifrado. Garcia *et al.* (2009) explicam que depois do comando de autenticação, o estado interno da cifra recebe a chave do novo setor, e o protocolo de autenticação inicia-se novamente. Nesse momento, no entanto, o desafio do cartão também será enviado cifrado. Na prática, os dados de tempo entre a primeira e segunda tentativa de autenticação expõe muita informação adicional que permite ao atacante prever com precisão qual nonce será usado, pois a distância entre os nonces utilizados pelo cartão em tentativas de autenticação consecutivas dependem fortemente do tempo entre essas tentativas, conforme exposto pela Definição 5.6.1. Além disso, os dados expostos através dos bits de paridade também podem ser utilizados para melhorar o desempenho do ataque.

Definição 5.6.1. Sejam n_C e n'_C dois nonces do cartão, podemos definir a distância entre n_C e n'_C como

$$d(n_C, n'_C) := \min \text{ suc}^i(n_C) = n'_C, \forall i \in \mathbb{N} \quad (5.1)$$

5.7 Autenticação com *timeout* expõe parte do *keystream*

No último estágio do processo de autenticação, quando o leitor envia $\{n_L\}$ e $\{r_L\}$, o cartão deve enviar a resposta $\{r_C\}$ para o leitor. Quando o cartão não envia essa resposta, o leitor envia um

sinal de *timeout* contendo o comando de parada $halt \oplus ks3$. Essa é uma fraqueza no protocolo de autenticação que permite ao atacante recuperar facilmente $ks3$, em seguida, levando em conta as informações utilizadas na inicialização, é possível calcular os estados anteriores do LFSR, permitindo dessa forma chegar ao $ks2$ e $ks1$. Mais detalhes explorando essa fraqueza serão apresentados no *Ataque por timeout* na seção 6.3.

5.8 Leitor aceita *frames* de tamanho inválido

De acordo com Rossum (2009), enquanto o chip do cartão se desativa depois de receber um *frame* de tamanho inesperado, o chip do leitor simplesmente realiza o *parse* sem verificar o tamanho. Essa é uma fraqueza no protocolo de comunicação. Quando um atacante envia um *frame* maior que o normal, os dados contidos nesse *frame* entrarão na composição do LFSR, permitindo desse modo ao atacante controlar o estado interno da cifra (48 bits).

5.9 Estatística viciada na cifra

Através de experimentos, Courtois (2009b) demonstrou que fixando o n_C , variando somente os três últimos bits de $\{n_L\}$, em 75% dos casos não ocorre nenhuma mudança no *keystream* $ks1$. O ideal seria que houvesse mudança no *keystream* $ks1$ em 50% dos casos. Observe que esse tipo de fraqueza permite melhorar bastante o desempenho de ataques para o cartão, pois elimina boa parte de tentativas que normalmente seriam feitas em um ataque por força bruta. Mais detalhes explorando essa fraqueza serão apresentados no *Ataque Somente ao Cartão* na seção 6.5.

5.10 Tamanho inadequado da chave

O tamanho da chave no MIFARE Classic é de 48 bits, considerada pequena para os dias de hoje. Mayes e Cid (2010) explicam que o projeto do MIFARE Classic é muito antigo, foi criado em 1994 e assim que foi lançado no mercado se tornou um sucesso, pois prometia equilíbrio entre segurança, velocidade e custo. Outro chip criado mais ou menos na mesma época foi o SIM, utilizado nos aparelhos de telefones celulares GSM. A Tabela 5.1 mostra um comparativo das duas tecnologias quando foram lançadas.

Características	MIFARE Classic	GSM
Tamanho da chave de autenticação	48 bits (A e B)	128 bits
Tamanho do nonce	Não Revelado	128 bits
Algoritmo de Autenticação	Não Revelado/Proprietário	Não Revelado/Proprietário
Tamanho da Chave de Criptografia	Não Revelado	máximo de 64 bits
Algoritmo de Criptografia	Não Revelado/Proprietário	A5/1

Tabela 5.1: Comparativo entre o MIFARE Classic e o SIM do GSM

Observe que o tamanho da chave no MIFARE Classic é bem menor do que no cartão SIM do GSM. Na época o custo da telefonia móvel não era muito acessível, a indústria ainda sofria muitos ataques no sistema analógico tomando posição de destaque. Mesmo assim, tentaram um projeto robusto e resistente a ataques. Levando em conta o tamanho da chave, um ataque por força bruta, levaria em média 2^{80} tentativas a mais na descoberta da chave. Uma questão importante a ser considerada é que o MIFARE Classic foi criado para caber em um cartão usando RFID com restrições de capacidade de processamento, e com custo bastante reduzido, justificando dessa forma as diferenças de projetos.

Na década de 90, em um esforço patrocinado pela *Electronic Frontier Foundation* o DES foi quebrado em pouco mais de 22 horas. Se em 2000 houvesse uma revisão no MIFARE Classic,

provavelmente indicaria que o produto era vulnerável a ataques por força bruta, caso ele se tornasse público e um *cracker* dedicado fosse implementado. A cifra de bloco DES tinha 18 anos quando o MIFARE Classic foi criado, e mesmo com seus 56 bits foi quebrado. Então os 48 bits do MIFARE não pode ser considerado seguro, pois é cerca de 2^8 vezes mais fraco.

5.11 Aplicações em produção usando chaves de transporte

Utilizar a chave de transporte, não é exatamente uma fraqueza no cartão MIFARE Classic, mas uma má utilização do produto. Os cartões vêm de fábrica com uma chave de transporte conhecida, para facilitar o trabalho dos integradores de sistemas. A Tabela 5.2 mostra algumas chaves que podem vir por padrão no cartão MIFARE Classic.

0xffffffffffff	0xa0a1a2a3a4a5	0xb0b1b2b3b4b5	0x4d3a99c351dd
0x1a982c7e459a	0x000000000000	0xd3f7d3f7d3f7	0xaabbccddeeff

Tabela 5.2: Chaves de transporte gravadas pelo fabricante

A maioria das pessoas envolvidas em projetos de software estão muito preocupadas com o prazo de entrega e provavelmente acabam passando por cima de questões importantes de segurança. Grunwald (2007) explica que cerca de 75% de importantes aplicações utilizam as chaves de transporte. Através de uma busca no Google pela string “A0A1A2A3A4A5”, uma das chaves de transporte padrão do cartão MIFARE, é possível verificar a popularidade das chaves de transporte. Lupták (2009) também relata que na República Tcheca/Esvlováquia existem muitas aplicações utilizando as chaves de transporte em seus cartões.

5.12 Cartões compatíveis muito fracos

Esta também não é exatamente uma fraqueza no cartão MIFARE Classic, mas nos cartões de fabricantes não autorizados, oriundos principalmente da China. Esses fabricantes não são licenciados oficialmente pela NXP para produzir o MIFARE Classic genuíno, porém eles estão produzindo produtos compatíveis. Segue alguns exemplos de cartões “compatíveis” apresentados no fórum Proxmark Developers Community (2009).

- Bellin BL75R06SM 8K-bit EEPROM
- Fudan Microelectronics FM11RF08 8KBits Contactless Card IC
- ISSI IS23SC4439
- ISSI IS23SC4456
- Quanray Electronics HF RFID QR2217 CHIP
- UNC20C01R 1Kbyte EEPROM Contactless Card IC

Courtois (2009a) relata um tipo de cartão completamente indistinguível de um MIFARE genuíno, utilizado no metrô de Kiev, na Ucrânia. Em seus experimentos verificou que o cartão sempre respondia como se fosse um cartão MIFARE Classic. Entretanto, ele sempre respondia a uma tentativa de falsificação, podendo ser clonado em alguns segundos. Alguns clones do cartão MIFARE removem as verificações de paridade, sem no entanto desabilitar as verificações de CRC. De acordo com Courtois, particularmente neste tipo de cartão as verificações de paridade estavam habilitadas.

Capítulo 6

Ataques

Este capítulo descreve os principais tipos de ataque em função das fraquezas descritas no capítulo anterior. Além disso, faz uma análise da consequência de cada tipo de ataque para aplicações que utilizam a tecnologia do MIFARE Classic. Todos os ataques aqui apresentados objetivam a recuperação das chaves secretas usadas no cartão.

6.1 Hardware e Software

Conforme descrito no Capítulo 1, um sistema RFID consiste de uma tag (nesse caso um cartão) e um leitor. O leitor possui um módulo de rádio frequência, uma unidade de controle e um dispositivo de acoplamento por aproximação para o cartão. O cartão por sua vez, contém um dispositivo de acoplamento por aproximação e um micro chip.

Além do hardware padrão, dependendo do tipo de ataque, faremos o uso de ferramentas adicionais incluindo hardware especializado, bibliotecas e softwares de apoio para análise de sinal RFID, da cifra Crypto-1 e do protocolo utilizado pelo cartão MIFARE.

6.1.1 Leitores

Nos ataques serão utilizados os leitores ACR122 da Touchatag e o OMNIKEY CardMan 5321, conforme ilustra a Figura 6.1. Esses leitores são compatíveis com o cartão MIFARE Classic, baratos e relativamente fáceis de se encontrar no mercado. Além disso, possuem bom suporte de *drivers* para Windows, Linux e Mac OS.



Figura 6.1: Leitores comuns padrão de mercado

6.1.2 Crpto1

A biblioteca *Crpto1*¹ é uma implementação aberta da cifra Crypto-1 para execução em software. Foi escrita em linguagem C, publicada por um *hacker* usando o pseudônimo Bla (2009). Além de conter a cifra Crypto-1, possui funções auxiliares que foram utilizadas no ataque proposto por Gans *et al.* (2008). Em sua versão mais recente, recebeu colaborações da comunidade de desenvolvedores para viabilizar os ataques propostos por (Courtois, 2009b; Garcia *et al.*, 2009). No entanto, não existe suporte na biblioteca para interfaceamento entre o leitor e o cartão. A seguir temos os protótipos das principais funções da Crpto1.

6.1.3 libnfc.org

A biblioteca *libnfc.org* - *Public platform independent NFC library*². Foi criada por Verdult (2009) em linguagem C, possibilitando o uso de dispositivos RFID em aplicações que utilizam Comunicação por Proximidade de Campo³ (NFC), sem no entanto estar vinculada a soluções proprietárias que dominam este segmento de mercado. Além do site do projeto, existe um fórum de discussão destinado à comunidade com bom índice de respostas.

6.1.4 Proxmark III

Para alguns ataques será necessário espionar a comunicação entre o leitor e o cartão. Eventualmente também será necessário simular um cartão. Jonathan Westhues (Westhues, 2007) desenvolveu o hardware e o software inicial do *Proxmark III*⁴, que é capaz de capturar a comunicação entre o cartão e o leitor, simular um cartão MIFARE Classic, permitindo inclusive modificar o UID. Além disso, também pode funcionar como um leitor. Este equipamento pode ser adquirido diretamente pelo site por aproximadamente \$400.

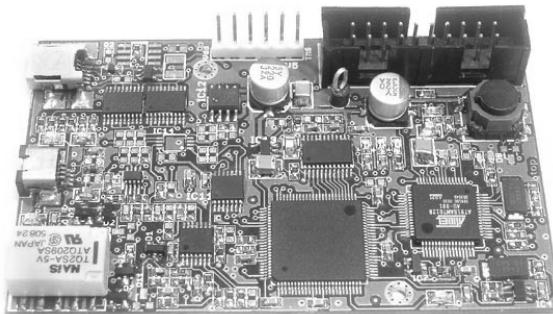


Figura 6.2: *Proxmark III*

O software do *Proxmark* é dividido em duas partes, *firmware* e *client*. O *firmware* foi escrito em linguagem C, utilizando um conjunto de bibliotecas mais restrito para executar de forma embarcada no hardware do *Proxmark*. O *client* também foi escrito em linguagem C, para execução no PC com versões disponíveis para Linux e Windows. Através do *client* é possível enviar comandos para modificar o comportamento do *Proxmark*, como por exemplo agir como leitor, como cartão ou mesmo como um espião da comunicação que ocorre entre o leitor e o cartão.

¹Projeto de software aberto disponível em <http://code.google.com/p/crypt01/>

²Projeto de software aberto disponível em <http://code.google.com/p/libnfc/>

³Do inglês *Near Field Communication*

⁴Projeto do hardware e o software abertos disponível em <http://cq.cx/proxmark3.pl>

6.2 Ataque por força bruta

O ataque por força bruta é o tipo de ataque mais simples, é uma variação do ataque adaptativo por texto cifrado escolhido. Foi proposto por (Garcia *et al.*, 2009). Para realizá-lo é necessário utilizar as bibliotecas Cryptool e libnfc.org como ferramenta de apoio, um cartão e um leitor simples conectados a um PC. Além disso também serão necessários recursos computacionais adicionais descritos mais adiante.

Usando a notação entre chaves {} para denotar dados criptografados, n_L representando o nonce do leitor, r_L o resposta do leitor, o atacante controlando o leitor, tenta se autenticar em um setor de sua escolha, explorando a exposição parcial do *keystream* nos bits de paridade discutidos na Seção 5.5, respondendo o desafio do cartão com oito bytes (e 8 bits de paridade) para $\{n_L\}$ e $\{r_L\}$. com a probabilidade de 1/256, os bits de paridade estarão corretos e o cartão responderá com o código de erro em 4 bits criptografados.

Repetindo esse procedimento diversas vezes (na prática 6 vezes são suficientes) é possível determinar a chave. Uma vez que a chave possui 48 bits, o atacante pode calcular a chave por força bruta: pode simplesmente verificar qual das 2^{48} chaves produz todas as seis vezes os bits de paridade corretamente. Na prática, juntando estas seis sessões de autenticações com os bits de paridade corretos, levam somente $6 \cdot 256 = 1536$ tentativas de autenticação que podem levar menos de um segundo. O tempo gasto para realizar um ataque por força bruta *off-line* dependente muito dos recursos computacionais do atacante.

O COPACOBANA (*Cost-Optimized Parallel COde Breaker*) é um computador baseado em FPGAs (*Field-Programmable Gate Array*), otimizado para execução de algoritmos criptanalíticos (Pelsl, 2006). Através da Figura 6.3 temos a ideia de suas dimensões. De acordo com o site do fabricante, ele é adequado para computação paralela com requisitos de baixa latência na comunicação. Em média, uma busca exaustiva de chave do DES de forma paralela, não leva mais que uma semana, enquanto que a mesma busca com um Pentium4/3GHz levaria cerca de 545 anos. Outras cifras também podem ser atacadas usando o COPACOBANA. Além disso, ele também pode ser usado para outros problemas de computação paralela. Seu custo é de aproximadamente \$10.000.



Figura 6.3: Visão superior do COPACOBANA

Com base na performance do COPACOBANA, considerando de maneira pessimista que a Crypto-1 em um FPGA precise realizar as mesmas verificações ao decriptografar o DES, e que o espaço de busca é um fator 256 vezes menor, ele levaria cerca de $6,4 \text{ dias}/256 = 36 \text{ minutos}$. Uma consideração importante a ser feita é que o algoritmo DES é muito mais complexo, e consequentemente mais caro computacionalmente; nesse sentido, esse cálculo está superestimado.

6.3 Ataque por *timeout* na autenticação do cartão

O ataque por *timeout* é uma mistura de ataque por *replay* e adaptativo com texto escolhido. Foi proposto em (Lupták, 2009) em detalhes com base do ataque apresentado em (Garcia *et al.*, 2008). Para realizá-lo é necessário utilizar um leitor genuíno do sistema interagindo com um cartão, a biblioteca Cryptool e o Proxmark III simulando um cartão MIFARE Classic. Esse ataque explora as fraquezas descritas na Seção 5.7 e 5.1. Para esse ataque é necessário um simulador de cartão (Proxmark III) conectado a uma leitora genuína do sistema que conhece as chaves.

Usando a notação entre chaves {} para denotar dados criptografados, n_C representando o nonce do cartão, n_L o nonce do leitor e \oplus a função lógica OU-exclusivo, o atacante faz uma autenticação parcial, controlando o cartão para enviar sempre o mesmo nonce n_C e quando o leitor responder o cartão com o desafio, o cartão simplesmente não responde e o leitor responderá $HALT \oplus ks3$. A Figura 6.4 ilustra esse processo.

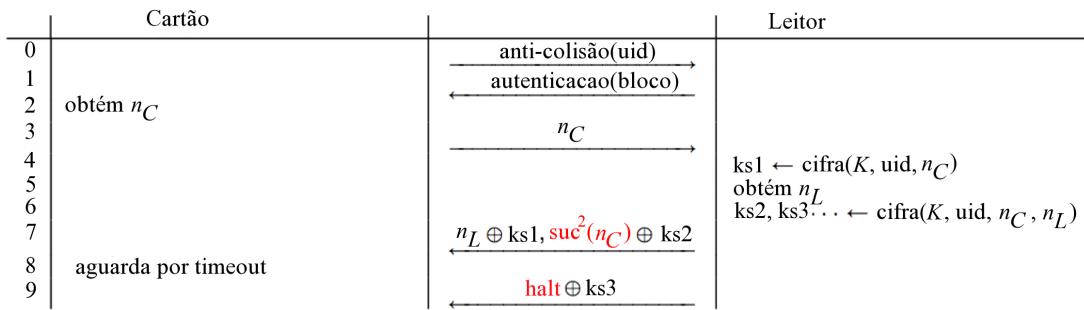


Figura 6.4: Autenticação com *Timeout*

Segue abaixo passo a passo a sequência de procedimentos para realizar o ataque:

1. Criar de maneira *off-line* uma tabela pré-calculada, contendo o estado do LFSR para 2^{36} com os estados entre 0 e 0xFFFFFFFF explorando as fraquezas descritas nas sessões 5.3, 5.2 e 5.5 para adequar $ks2$ com $ks3$ (leva de 4-8 horas).
2. Calcular de forma *on-line* uma tabela com os nonces n_C para 2^{12} entradas de 0 até 0xFFFF e adequar com $ks2, ks3$, de forma que exista um n_C produzindo um LFSR para um determinado $ks2, ks3$ (leva de 2-14 minutos).
3. Restaurando o $n_L, n_C \oplus uid$, o resultado é a chave.

6.4 Ataque por interceptação da autenticação

O ataque por interceptação da comunicação é um tipo de ataque conhecido como *replay*, foi proposto inicialmente por (Gans *et al.*, 2008). Para realizá-lo basta interceptar a comunicação realizada entre o cartão e o leitor durante o processo de autenticação de um determinado setor do cartão. Apenas uma autenticação é suficiente para recuperação da chave. Essa comunicação deverá ser gravada em um arquivo de log, que será analisado posteriormente. A consequência deste ataque é que com a chave recuperada, o atacante poderá ler e modificar os dados do setor que foi autenticado.

Nesse ataque serão exploradas duas fraquezas na cifra Crypto-1, a fraqueza dos bits ímpares usados na geração do *keystream*, conforme descrito Seção 5.3 para recuperar o estado do LFSR, e a fraqueza do filtro gerador que não usa os nove primeiros bits do gerador, também descrito na Seção 5.2.

Para esse ataque é necessário utilizar um leitor genuíno do sistema interagindo com um cartão, e o equipamento Proxmark III para interceptar a comunicação. Além disso, será utilizada a biblioteca Cryptool como componente de software complementar. Para iniciar captura de dados deve-se executar o programa *cliente* e enviar o comando `hi14asnoop`. Na Figura 6.5 ilustra o processo de captura da comunicação entre o cartão e o leitor, enquanto que a figura 6.6 mostra um exemplo de log capturado, com duas colunas a mais (*SEQ*, *Comments*) para facilitar a compreensão.

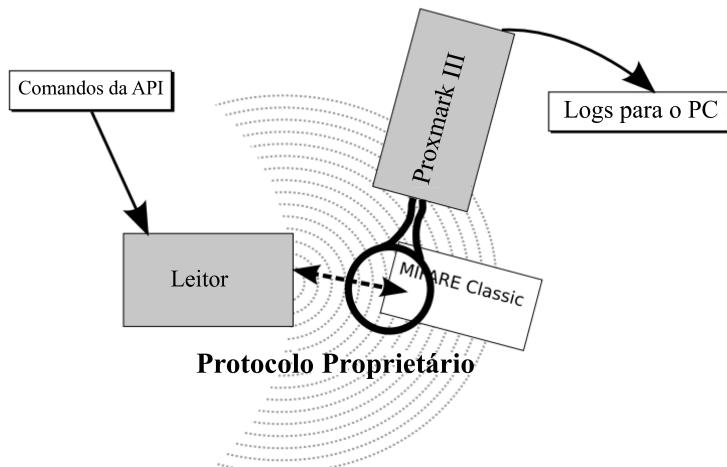


Figura 6.5: Captura de logs da comunicação entre cartão e leitor

SEQ	ETU	size	sender	bytes	Comments
1 :	561882:	1	: PCD	26	REQA
2 :	64:	2	: TAG	04 00	Answer REQA
3 :	10217:	2	: PCD	93 20	SELECT
4 :	64:	5	: TAG	9c 59 9b 32 6c	UID: 9c 59 9b 32
5 :	12313:	9	: PCD	93 70 9c 59 9b 32 6c 6b 30	SELECT with UID
6 :	64:	3	: TAG	08 b6 dd	Tag Type (Mifare 1K)
7 :	923318:	4	: PCD	60 00 f5 7b	AUTH (Block 00)
8 :	112:	4	: TAG	82 a4 16 6c	Challenge Tag (nt, Nonce Tag)
9 :	6985:	8	: PCD	a1 e4! 58 ce! 6e ea! 41 e0!	nr XOR ks1 (Nonce Reader, ciphered, 4 bytes),
:	:				ar XOR ks2 (Answer Reader to Nonce Tag, ciphered)
10 :	64:	4	: TAG	5c! ad f4 39!	at XOR ks3 (Answer Tag, ciphered)

Figura 6.6: Exemplo de Log Capturado pelo Proxmark III

De posse do arquivo de log de uma autenticação capturado pelo Proxmark III, basta escrever um simples programa usando as funções da Cryptol com o UID, nonce do cartão, nonce do leitor, resposta do cartão e a resposta do leitor. A listagem 6.1 mostra um programa exemplo como prova de conceito, utilizando os dados do log apresentado na Figura 6.6. Observe que no arquivo de log, a partir de um determinado momento aparece o símbolo '!' nos bytes enviados. Essa é uma representação indicando que os dados estão cifrados. Para a escrita do programa, esse símbolo será ignorado.

```

1 #include "cryptol.h"
2 #include <stdio.h>
3
4 int main (int argc, char* argv[]) {
5
6     struct Crypto1State *revstate;
7     uint64_t lfsr;
8     unsigned char* plfsr = (unsigned char*)&lfsr;
9
10    uint32_t uid = 0x9c599b32;
11    uint32_t nt = 0x82a4166c; /* Nonce Tag */
12    uint32_t nr = 0xa1e458ce; /* Nonce Reader */
13    uint32_t at = 0x5cadf439; /* Answer Tag */
14    uint32_t ar = 0x6eea41e0; /* Answer Reader */
15
16    uint32_t ks2 = ar ^ prng_successor(nt, 64);
17    uint32_t ks3 = at ^ prng_successor(nt, 96);
18
19    printf("nt': %08x\n", prng_successor(nt, 64));
20    printf("nt'': %08x\n", prng_successor(nt, 96));
21
22    printf("ks2: %08x\n", ks2);
23    printf("ks3: %08x\n", ks3);
24
25    revstate = lfsr_recovery64(ks2, ks3);
26    lfsr_rollback_word(revstate, 0, 0);
27    lfsr_rollback_word(revstate, 0, 0);
28    lfsr_rollback_word(revstate, nr, 1);
29    lfsr_rollback_word(revstate, uid ^ nt, 0);
30
31    cryptol_get_lfsr(revstate, &lfsr);
32
33    printf("Found Key: [%02x %02x %02x %02x %02x %02x]\n\n",
34          plfsr[0], plfsr[1], plfsr[2], plfsr[3], plfsr[4], plfsr[5]);
35
36    return 0;
37 }
```

Listagem 6.1: Programa para recuperar a chave de uma autenticação

A execução desse programa em um computador com processador Intel Core 2 Duo/2.4 GHz leva em torno de 1 segundo, gerando a seguinte saída:

```

nt': 8d65734b
nt'': 9a427b20
ks2: e38f32ab
ks3: c6ef8f19
Found Key: [ff ff ff ff ff ff]
```

6.5 Ataque Somente ao Cartão

O *Ataque Somente ao Cartão* foi proposto em (Garcia *et al.*, 2009) e posteriormente em (Courtois, 2009b), sendo esse último o mais eficiente. É uma variação do ataque adaptativo por texto cifrado escolhido. Nesse trabalho será apresentado o ataque proposto por Courtois. Os ataques apresentados anteriormente exceto o ataque por força bruta, necessitam da interação com um leitor genuíno, onde a aplicação conhece a(s) chave(s) secreta(s). Esses tipos de ataque são mais difíceis de ser realizados, pois os leitores normalmente estão em locais públicos, aumentando assim as chances de um fraudador ser pego no momento do ataque e tenha que responder criminalmente por isso.

Nesse ataque é necessário apenas um leitor conectado a um PC interagindo com o cartão através das bibliotecas libnfc.org e Cryptool, sem nenhuma interação com o sistema que faz uso desse cartão, sendo capaz de recuperar todas as chaves gravadas no cartão. Esse ataque é muito mais atraente para um fraudador pois há pouca exposição, podendo ser feito em local controlado. Além disso, possui baixo custo, pois necessita apenas de um leitor, como por exemplo o ACR122 da Touchatag que pode ser facilmente adquirido pela internet por menos de \$70.

Usando a notação entre chaves {} para denotar dados criptografados, n_C representando o nonce do cartão, n_L o nonce do leitor, r_L a resposta do leitor, ksN os *keystreams*, onde N é o número do *keystream* e \oplus a função lógica OU-exclusivo, no processo de autenticação depois que o cartão envia o desafio através do nonce n_C , o leitor responde com $\{n_L\} = n_L \oplus ks1, \{r_L\} = suc^2(n_C) \oplus ks2$. Usando a mesma notação acima, representando a resposta do cartão pelo criptograma C de 8 bytes temos que $C = \{n_L\}\{r_L\}$, podendo C ser representado em bytes por $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ e $P_C = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$ os 8 bits de paridade que são enviados juntos com os 8 bytes de C , em seus experimentos Courtois observou os seguintes fatos:

1. Ao executar o processo de autenticação para um cartão o qual não se conhece a chave, algumas vezes depois que o leitor envia a resposta, o cartão responde com 4 bits ao invés de 4 bytes. Esse comportamento já havia sido relatado em (Gans *et al.*, 2008) e apresentado nesse trabalho na lista de fraquezas na Seção 5.5.
2. O cartão responde com NACK de 4 bits se, e somente se, os bits de paridade para o texto em claro tiverem corretos.
3. Não importa qual é o criptograma C forjado, existe exatamente uma escolha para os bits de paridade P_C , então o cartão irá responder com um NACK de 4 bits cifrados.
4. Ao fixar os 3 primeiros bytes de C e variar alguns bits do 4º byte C_3 , a probabilidade dos 3 bits do *keystream* gerado durante a decriptografia dos 3 últimos bits do 4º byte C_3 não depender desses bits de C_3 é muito alta, cerca de 0,75.
5. Ao fixar o nonce n_C e um prefixo de 29 bits de C , com probabilidade de cerca de 0,75, os bits do *keystream* gerados nesse processo são constantes simultaneamente para 8 diferentes cifragens de C que compartilham o mesmo prefixo de 29 bits e variam os 3 últimos bits de C_3 .
6. Do ponto de vista diferencial, ao assumir que o *keystream* gerado durante a decriptografia do 4º byte C_3 , é constante e não depende desse byte, então a diferença do estado da cifra, a qualquer momento durante o cálculo de $ks2$ e ks é uma função multivariada linear fixa que depende das diferenças em C_3 e nada mais.

6.5.1 Detalhes do Ataque

Com base em seus experimentos e observações, Courtois propôs esse ataque, o qual descrevemos em detalhes abaixo:

1. **Estágio 1.** Enviar 128 consultas, fixando o nonce do cartão n_C e utilizando um criptograma aleatório de 8 bytes $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$, e um número aleatório P_C para obter um caso onde o cartão responde com 4 bits.
2. **Estágio 2.** Manter o tempo fixo, para que o nonce do cartão n_C seja sempre o mesmo, e os mesmos 29 bits iniciais do criptograma C para o qual o cartão respondeu; além disso, mantendo a segunda metade do criptograma (c_4, c_5, c_6, c_7) fixa, de modo que somente 3 bits sejam variáveis em C (somente 8 casos), e também mantendo fixo 3 primeiros bits de paridades no P_C .
3. Tentar em ordem, cada um dos 8 possíveis valores de $\{n_L\}$ e diferentes escolhas dos últimos 5 bits de P_C . Para cada texto cifrado, levando em conta o 3º fato observado por Courtois, exatamente 1 em 2^5 casos o cartão irá responder, ele responde com probabilidade 1/32 se ternarmos de forma aleatória, e responde após 16 passos em média (então passamos para o próximo valor para $\{n_L\}$). Depois de $8 \cdot 16$ consultas em média, obtém-se 8 respostas com todos os 8 possíveis valores consecutivos de $\{n_L\}$.
4. As 8 respostas de 4 bits que recebemos, nos dá $8 \cdot 4 = 32$ bits de informação a respeito da chave de 48 bits.
5. Agora com probabilidade de cerca de 0,75, pode-se prever simultaneamente as diferenças de estados para todas as 8 cifragens (conforme fatos 5 e 6).
6. Agora usando o fato de que o algoritmo da Crypto-1 reusa a maioria dos bits de estado após 2 passos. Nesse sentido, somente 21 bits de estado determina os dois *keystreams* de bits $(ks3)_0$ e $(ks3)_2$. Examinando todos os 2^{21} casos e para cada texto cifrado onde o cartão tenha respondido podemos dividir o tamanho de nosso espaço por 4. Com 8 respostas determina-se cerca de 2^5 possíveis valores para os 21 bits.
7. Da mesma forma, determina-se 2^5 possibilidades para os outros 21 bits de estado que determina os bits de estado de $(ks3)_3$.
8. Então, tem-se a uma lista de 2^{10} estados em 24 bits, dos quais é necessário estender para 2^{16} estados possíveis em 48 bits.
9. Assim, revertendo o estado, tem-se cerca de 2^{16} possíveis chaves iniciais; daí, verificando todos os $8 \cdot 8$ bits de paridade envolvidos no ataque permite saber qual chave está correta com uma certeza aproximada. Ou, se houver uma contradição em qualquer estágio, significa que o *keystream* não depende de C_3 , ao contrário da suposição do fato 5.
10. Se isso falhar, repete-se o ataque inteiro. Em média espera-se que o ataque seja repetido $1/0,75 \approx 1,33$ vezes.

6.5.2 Complexidade do Ataque

Executando de forma repetida, pode-se salvar a complexidade do Estágio 1: Se for modificado somente c_3 da próxima vez, chega-se ao Estágio 1, o cartão irá responder com probabilidade 1/64, e serão necessárias 32 consultas em média. Assim, o número médio de consultas ao cartão nesse ataque é de $128 + (1,33 - 1) \cdot 32 + 1,33 \cdot 8 \cdot 16 \approx 300$. Observação: esse ataque ficou bastante eficiente em relação a um ataque equivalente proposto em (Garcia et al., 2009), o qual necessita de 28500 consultas ao cartão.

Esses dados levam cerca de 5 minutos para serem reunidos em uma velocidade de 1 transação por segundo. Entretanto, Courtois argumenta que se for implementado um hardware semelhante ao utilizado em (Garcia *et al.*, 2009), deverá levar cerca de 10 segundos.

Além disso, nos cartões muito fracos conforme exposto na Seção 5.12, serão necessárias apenas 12 consultas (mais que 8, pois com os bits paridade desabilitados não há forma de saber as chaves rejeitadas) repetindo cerca de 1/0,75 vezes, levando em média 16 consultas. Nesse sentido, esses cartões podem ser clonados em cerca de 1 segundo.

Depois de recuperar a chave, pode-se autenticar no setor onde a chave foi recuperada a fim de explorar a fraqueza de *Autenticações Aninhadas* apresentada na Seção 5.6 através do ataque proposto em (Garcia *et al.*, 2009) podendo recuperar todas as chaves do cartão com poucas consultas.

42

ATAQUES

6.5

Capítulo 7

Medidas de Contorno

Este capítulo apresenta propostas para medidas de contorno no sentido de evitar ou minimizar fraudes, principalmente envolvendo transações financeiras. Os ataques ao cartão MIFARE Classic objetivam a recuperação das chaves secretas do cartão, tornando possível:

- Clonar o cartão (exceto o bloco zero)
- Restaurar no cartão um estado válido anterior
- Modificar os dados contidos no cartão

Mayes e Cid (2010) argumentam que é uma boa prática em projetos de software realizar o levantamento de requisitos num primeiro momento, e implementação e tecnologia num segundo momento. Na fase de levantamento de requisitos devem ser avaliados riscos de segurança e como eles influenciam nas medidas de segurança dentro do sistema. No entanto quando se fala de RFID e cartões inteligentes essa divisão pode não ser tão clara, pois as funcionalidades passíveis de implementação podem ser fortemente limitadas por recursos. Talvez exista apenas uma abordagem técnica viável, equilibrando funcionalidade, velocidade, segurança e custo do ponto de vista de negócio. Mesmo que haja mais que uma alternativa, deve-se avaliar os riscos e fornecer possíveis medidas de contorno via *back-office*¹ ou por processos humanos.

O cartão MIFARE Classic foi considerado adequado por muito tempo, mas depois das publicações indicando sérios problemas de segurança é difícil argumentar que o sistema é tão seguro quanto os projetistas esperavam ser. No entanto é preciso levar em consideração que ele foi criado para caber em um cartão usando RFID com restrições de capacidade de processamento.

Através de experimentos, Tan (2009) percebeu que o principal problema dos emuladores de cartão está relacionado com os atrasos na implementação em software da Crypto-1; assim, conseguem simular bem somente a fase inicial da comunicação com o leitor conhecida como anti-colisão. Além disso, as funcionalidades existentes nos emuladores não cobrem tudo que existe em um cartão genuíno. Nesse sentido, ainda não temos um emulador que possa passar de forma despercebida como se fosse um cartão.

As propostas apresentadas nesse trabalho não corrigem os problemas existentes no cartão, pois não se trata de um cartão programável, a promessa do MIFARE Classic é apenas de armazenamento protegido. Uma vez que a cifra Crypto-1 e o protocolo de autenticação utilizados no MIFARE Classic estão de fato comprometidos, seria interessante utilizá-los, mas assumir que o cartão será utilizado como um sistema de arquivos. Acrescentando uma camada adicional de criptografia na aplicação, seria pelo menos capaz de evitar a clonagem e modificação não autorizada dos dados contidos no cartão. Nesse sentido, as medidas aqui apresentadas deverão ser executadas no equipamento que está conectado ao leitor, normalmente é um PC.

¹Sistemas de reaguarda para dar apoio administrativo aos departamentos de uma empresa

A vantagem é que esta medida pode ser realizada de forma completamente *off-line* e com custo relativamente baixo. No entanto, quando se fala em criptografia, o gerenciamento de chaves merece uma atenção especial. Se a opção for por uma implementação inteiramente baseada em software, será necessário implementar os algoritmos criptográficos e criar mecanismos para proteção das chaves. Entretanto, existem no mercado alternativas para gerenciamento de chaves de forma segura com preços acessíveis, como por exemplo *tokens* criptográficos ou SAMs² com suporte aos principais algoritmos criptográficos. Além de melhorar a segurança, esse tipo de hardware permite melhor desempenho em aplicações embarcadas com restrições de processamento nas medidas de contorno propostas. Obviamente a escolha pela implementação em software ou por um hardware especializado é um *trade-off* entre custo e segurança.

Conforme exposto na Seção 3.1, sabemos que o bloco zero do setor zero do cartão é somente leitura e contém o UID e informações do fabricante. Essa será a principal funcionalidade que usaremos do cartão nas medidas de contorno apresentadas. No entanto, se no futuro for descoberto alguma fraqueza que permita modificar essa característica, as medidas aqui apresentadas precisam ser revistas.

7.1 Diversificação de Chaves

Ao invés de utilizar uma única chave compartilhada para todos os cartões, deve-se derivar as chaves de criptografia de uma chave mestra (chave compartilhada) usando informações adicionais inerentes ao cartão, como por exemplo UID, setor, bloco e um algoritmo como por exemplo *HMAC*³. Aplicando esta medida, aumentamos a dificuldade de ataques ao sistema, pois se alguma chave utilizada por algum cartão for comprometida, somente o cartão em questão estará comprometido. A desvantagem desta medida, é que requer de uma chave mestre de criptografia que deve compartilhada entre os computadores que farão o acesso aos cartões.

7.2 Cifragem dos Dados

Utilizando a medida anterior e uma cifra simétrica como por exemplo AES, seria possível armazenar os dados cifrados, evitando assim, a leitura de dados confidenciais. Como cada cartão possui sua própria chave de criptografia, se um usuário tentar clonar um cartão, a chave será outra, pois leva em conta o UID, o que resultará em um cartão com dados sem sentido. Caso esse mesmo usuário modifique alguma informação no cartão, ela não fará sentido, pois deveria ser cifrada pela chave derivada do UID, ou seja, o conteúdo do cartão estará corrompido.

7.3 Verificação de Integridade com HMAC

A verificação de integridade requer a reserva de um bloco específico para armazenamento dos dados. Toda vez que os dados do cartão forem modificados, deve-se gerar em seguida um *HMAC* do conteúdo do cartão, incluindo os dados do bloco zero (onde se encontra o UID), e gravando posteriormente no bloco reservado para verificação de integridade. Assim, será possível realizar uma verificação prévia de integridade a cada vez que o cartão for utilizado, possibilitando a identificação de uma eventual clonagem ou alteração indevida de dados no cartão.

²Security Access Module

³Hash Message Authentication Code

7.4 Verificação de Integridade com Assinatura

É bastante similar à medida anterior, trocando o HMAC por uma assinatura digital como por exemplo DSS ou RSA. A vantagem desta medida é que usando criptografia de chave pública, não precisamos usar uma chave secreta compartilhada entre os computadores que farão o acesso ao cartão. Uma das desvantagens dessa medida é que os algoritmos de assinatura são significativamente mais caros computacionalmente do que um HMAC, tornando-se um problema para sistemas de baixa capacidade de processamento. Além disso, a assinatura ocupará mais espaço para armazenamento no cartão, pois é proporcional ao tamanho da chave. Uma chave de 512 bits por exemplo ocupará 4 blocos no cartão.

7.5 Associação de UID

A associação de UID tem o objetivo de evitar a clonagem de cartões. Se a chave secreta do cartão for recuperada, é possível copiar todo o cartão exceto o Bloco 0 do Setor 0, pois os dados são somente para leitura e contém o UID. Assim, devemos fazer uma associação do UID com o dono do cartão, gravando esta informação em um banco de dados para posteriormente realizar as validações dos UIDs válidos. A vantagem desta medida é que mesmo que um cartão seja clonado ele não será um cartão válido dentro do sistema. Uma clara desvantagem com esta medida é que torna-se necessário a geração de uma lista branca contendo os cartões válidos, sendo necessário conectividade de rede para a distribuição desta lista para todos os computadores que farão acesso aos cartões. Entretanto, existem alguns casos que não podem esperar caso haja alguma indisponibilidade na rede. Nesse sentido, o software deverá ser capaz de armazenar os dados localmente, de forma que possa enviar tais informações posteriormente quando a conectividade de rede se re-estabelecer.

7.6 Verificação do contador de decremento

A verificação do contador de decremento objetiva evitar a restauração de um estado válido para o cartão ou o uso de “clones perfeitos” (emuladores), caso funcionem corretamente. Os cenários em que esse tipo de fraude poderão ocorrer são descritos a seguir:

- Após realizar uma carga de créditos de forma legal em seu cartão, um usuário mal intencionado poderá fazer uma cópia do conteúdo do cartão, gastar todos os créditos, posteriormente restaurar essa cópia no cartão. O conteúdo do cartão é completamente válido para o sistema, mas os créditos foram copiados.
- Sabe-se que os emuladores possuem dificuldades para emular 100% das funcionalidades do MIFARE Classic. Há rumores da existência de um cartão “compatível” que permite reprogramar o UID. Se de fato existir algum tipo de cartão que aceite modificar o UID, ou mesmo se algum emulador conseguir emular 100% das funcionalidades, ele poderá tentar se passar por um cartão genuíno.

Através de um contador de decremento gravado no cartão (inicialmente com o valor 2^{31}), e decrementado a cada transação, é possível identificar no sistema de *back-office* que um cartão já usou um determinado número do seu contador de transações, sinalizando uma tentativa de fraude, permitindo o bloqueio do cartão de forma que não seja mais usado, preferencialmente no menor tempo possível. A desvantagem desta medida é que requer conectividade de rede para realizar estas verificações de forma *on-line* ou *semi on-line*.

UID	Data/Hora	Operação	Valor	Contador
0x0000FFFF	05/01 09:30	Crédito	R\$ 500,00	50001
0x0000FFFF	05/01 09:41	Débito	R\$ 3,00	50000
0x0000FFFF	05/01 17:49		R\$ 3,00	49999
				.
				.
0x0000FFFF	07/01 12:37	Débito	R\$ 3,00	50000

Conflito!

Contador já Utilizado

Figura 7.1: Exemplo de fraude detectada pelo back office

7.6.1 Cartões mais Seguros

Por fim, dependendo do tipo de aplicação, a recomendação é de fato a troca para modelos mais seguros, sobretudo em aplicações envolvendo transações eletrônicas. Existem outras versões de MIFARE que utilizam padrões abertos de criptografia, não foram comprometidas e são retro-compatíveis com a versão Classic. O MIFARE Plus possui suporte ao AES, o DESFire possui suporte para DES/3DES/3K3DES/AES, o SmartMX é um JavaCard que além dos algoritmos das versões Plus e DESFire, também possui suporte aos algoritmos de chave pública RSA e ECC. Obviamente esses cartões são mais caros que a versão Classic, novamente aqui temos um *trade-off* entre custo e segurança.

Além disso, não basta simplesmente trocar os cartões para um modelo mais seguro (mesmo que seja compatível), será necessário realizar modificações no software de forma a explorar os recursos de segurança oferecidos por estes cartões. Adicionalmente, poderá ser necessário trocar leitores de cartão incompatíveis com o novo modelo de cartão adotado, podendo envolver custos muito maiores do que ocorreram na implantação da versão inicial com o MIFARE Classic.

7.7 Considerações Finais

É importante lembrar que se continuar utilizando o MIFARE Classic, as medidas apresentadas devem preferencialmente ser aplicadas integralmente para garantir um nível de segurança satisfatório. A figura 7.2 mostra um diagrama contendo as medidas de contorno, de forma que quanto mais medidas empilhadas, resulta em melhor segurança. No entanto, o aumento da segurança virá acompanhada de complexidade e custo.

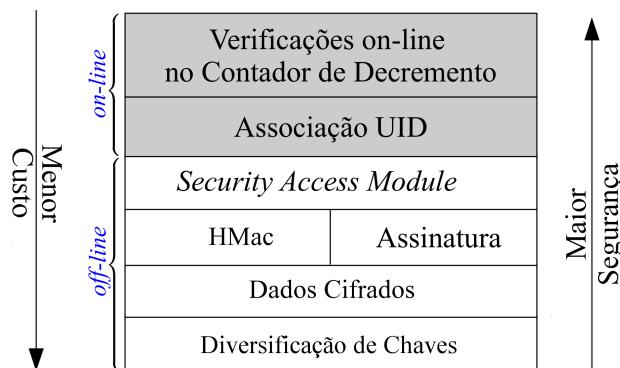


Figura 7.2: Nível de segurança de acordo com as medidas de contorno aplicadas

Se não for possível aplicar todas as medidas, aplique o máximo que puder. Caso contrário poderá surtir pouco ou nenhum efeito. As medidas *off-line* devem preferencialmente ser aplicadas integralmente, e se não for viável a aplicação das medidas *on-line*, deve-se pelo menos fazer verificações em lote ao final do dia para bloquear cartões envolvendo fraudes.

Lembrando que se houver mais problemas de segurança ainda desconhecidos no MIFARE Classic, mas que no futuro sejam evidenciados, talvez alguma destas medidas deixem de ter o efeito desejado.

Capítulo 8

Conclusões

Conforme apresentamos nesse trabalho, através de qualquer um dos ataques descritos é possível recuperar todas as chaves, e como consequência é possível clonar um cartão ou modificar os dados de um cartão sem a devida autorização. Nesse trabalho, apresentamo medidas com variados níveis de segurança e custo, usando diferentes abordagens para contornar esses problemas.

As vulnerabilidades publicadas do MIFARE Classic devem ser consideradas nos sistemas de produção, sendo importante que sejam tomadas as medidas apresentadas nesse trabalho, no sentido de obtenção de mais segurança. Assim, quando houver a necessidade, pode-se planejar uma migração de forma planejada para um modelo de cartão mais seguro de forma gradual, evitando deixar tanta responsabilidade para o sistema de *back-office*.

De acordo com Courtois (2009b) em algumas indústrias o segredo do algoritmo é indispensável, sendo importante para garantir uma melhora real na segurança. Por exemplo, é muito difícil e caro proteger cartões inteligentes contra ataques por canal secundário. Outro exemplo são os sistemas de TV por assinatura, os quais são altamente dependente do segredo de seus algoritmos embarcados, pois caso fossem abertos seria inviável o seu uso em hardware barato. Obviamente, o segredo só é uma boa ideia somente e esses algoritmos são muito bons.

A segurança por obscuridade, sem qualquer análise de segurança criptográfica prévia dos algoritmos criptográficos a serem inseridos na aplicação, sem que tenham sido criados por criptográficos a analisados por criptanalistas experientes pode ser uma temeridade, e muito falho para sistemas. Aparentemente o problema no caso do MIFARE Classic, é que não houve uma ampla revisão em seu projeto. A principal lição que tiramos disso é que caso haja necessidade de utilizar algoritmos secretos, seus projetos devem ser amplamente revistos por especialistas. No entanto, se não for possível uma revisão rigorosa, a adoção de padrões abertos é fortemente recomendada.

8.1 Considerações Finais

As medidas de contorno propostas nesse trabalho levam em conta que o bloco zero do cartão não é passível de gravação. Além disso, de acordo com as pesquisas realizadas, os emuladores de cartão ainda se encontram em um estágio inicial, sem a capacidade de emular completamente um cartão, ou seja, após a fase de anti-colisão. Se houver alguma mudança nesse cenário, as medidas propostas nesse trabalho precisam ser revistas.

8.2 Trabalhos Futuros

Além das medidas apresentadas, existem outras possibilidades para identificar ataques. Em particular, o tempo gasto durante a comunicação entre o leitor e o cartão, pois a especificação ISO-14443 possui requisitos de tempo bastante restritos para a comunicação entre o leitor e o cartão. De acordo com Silbermann (2009), muitos leitores aceitam uma pausa maior que $200 \mu\text{s}$, assim o atraso de tempo não é detectado. Nesse sentido, se o tempo for rigorosamente controlado na aplicação, provavelmente seria possível evitar ataques por emuladores e ataques por *relay*.

Adicionalmente, seria possível tornar as medidas ainda mais efetivas, se houvesse uma forma para evitar a restauração de um estado válido do cartão de maneira *off-line*. Isso tornaria o sistema imune a fraudes sem a necessidade de verificações *on-line* no sistema de *back-office*.

Referências Bibliográficas

Bla(2009) Bla. Cryptol - Open implementations of attacks against the crypto1 cipher, as used in some RFID cards. <http://code.google.com/p/cryptol/>, 2009. Disponível em 01/05/2011. Citado na pág. 34

Courtois(2009a) Nicolas T Courtois. Card-Only Attacks on MiFare Classic. http://www.cosic.esat.kuleuven.be/rfidsec09/Papers/mifare_courtois_rfidsec09.pdf, 2009a. Disponível em 01/05/2011. Citado na pág. 31

Courtois(2009b) Nicolas T Courtois. The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes, Anywhere, Anytime. *Disclosure*, 2002(4). doi: 10.1016/S1353-4858(02)00415-4. URL <http://eprint.iacr.org/2009/137>. Citado na pág. 2, 30, 34, 39, 49

Diffie e Hellman(1976) Whitfield Diffie e Martin E. Hellman. New directions in cryptography, 1976. Citado na pág. 8

Finkenzeller(2003) Klaus Finkenzeller. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons. URL <http://www.amazon.fr/exec/obidos/ASIN/0470844027/citeulike04-21>. Citado na pág. 1

Gans et al.(2008) Gerhard Gans, Jaap-Henk Hoepman, e Flavio Garcia. A Practical Attack on the MIFARE Classic. Em Gilles Grimaud e François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications*, volume 5189 of *Lecture Notes in Computer Science*, chapter 20, páginas 267–282. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-85892-8. doi: 10.1007/978-3-540-85893-5__20. URL http://dx.doi.org/10.1007/978-3-540-85893-5_20. Citado na pág. 2, 19, 21, 29, 34, 37, 39

Garcia et al.(2008) Flavio Garcia, Gerhard Gans, Ruben Muijres, Peter van Rossum, Roel Verdult, Ronny Schreur, e Bart Jacobs. Dismantling {MIFARE} Classic. Em Sushil Jajodia e Javier Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, chapter 7, páginas 97–114. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-88312-8. doi: 10.1007/978-3-540-88313-5__7. URL http://dx.doi.org/10.1007/978-3-540-88313-5_7. Citado na pág. 21, 23, 28, 29, 36

Garcia et al.(2009) Flavio D Garcia, Peter Van Rossum, Roel Verdult, e Ronny Wichers Schreur. Wirelessly Pickpocketing a Mifare Classic Card. Em *IEEE Symposium on Security and Privacy – S&P ’09*, páginas 3–15, Oakland, California, USA. IEEE, IEEE Computer Society. ISBN 9780769536330. doi: 10.1109/SP.2009.6. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5207633>. Citado na pág. 2, 27, 28, 29, 34, 35, 39, 40, 41

Grunwald(2007) Lukas Grunwald. New Attacks against RFID-Systems. <http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Grunwald.pdf>, 2007. Disponível em 01/05/2011. Citado na pág. 31

ISO 14443() ISO 14443. Identification cards - Contactless integrated circuits cards - Proximity cards, 2000. Citado na pág. 16, 25, 29

- Kerckhoffs(1883)** Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–83. URL <http://www.petitcolas.net/fabien/kerckhoffs/>. Citado na pág. 21
- Lupták(2009)** Pavol Lupták. MIFARE Classic analysis in Czech / Slovakia. <http://www.nethemba.com/mifare-classic-slides.pdf>, 2009. Disponível em 01/05/2011. Citado na pág. 31, 36
- Mayes e Cid(2010)** Keith E Mayes e Carlos Cid. The MIFARE Classic story. *Information Security Technical Report*, 15(1):8–12. ISSN 1363-4127. doi: DOI:10.1016/j.istr.2010.10.009. URL <http://www.sciencedirect.com/science/article/B6VJC-51H5SC8-1/2/020460931b16f5a257754899181fab6b>. Citado na pág. 21, 30, 43
- Menezes et al.(1996)** Alfred J. Menezes, Scott A. Vanstone, e Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edição. ISBN 0849385237. Citado na pág. 6, 8
- Mills(2008)** Elinor Mills. D-Day for RFID-based transit card systems. http://news.cnet.com/8301-1009_3-10059605-83.html, October 2008. Disponível em 01/05/2011. Citado na pág. 2
- Nohl e Plötz(2007)** Karsten Nohl e Henryk Plötz. MIFARE - Little Security, Despite Obscurity. <http://events.ccc.de/congress/2007/Fahrplan/events/2378.en.html>, December 2007. Disponível em 01/05/2011. Citado na pág. 21, 23
- Nohl et al.(2008)** Karsten Nohl, David Evans, S Starbug, e Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. *Science*, (July):185–193. URL <http://www.usenix.org/events/sec08/tech/nohl.html>. Citado na pág. 2, 21, 22, 27
- NXP(2008)** Semiconductors NXP. *MIFARE Standard 1KByte Card IC Functional Specification*, 5.3 edição, January 2008. Citado na pág. 11
- Pelsl(2006)** Jan Pelsl. Cryptanalysis with a cost-optimized FPGA cluster. http://www.scieengines.com/copacobana/paper/IPAM2006_slides.pdf, December 2006. Disponível em 01/10/2011. Citado na pág. 35
- Proxmark Developers Community(2009)** Proxmark Developers Community. Mifare Classic Clones. <http://www.proxmark.org/forum/viewtopic.php?id=169>, 2009. Disponível em 01/05/2011. Citado na pág. 31
- Rossum(2009)** Peter Van Rossum. Mifare Classic Troubles, 2009. URL <http://www.ict-forward.eu/media/workshop2/presentations/rossum-mifare.pdf>. Citado na pág. 30
- RSA Laboratories(1996)** RSA Laboratories. Frequently Asked Questions About Today's Cryptography, 1996. Citado na pág. 6, 7, 9
- RSA Laboratories()** RSA Laboratories. What is a stream cipher? <http://www.rsa.com/rsalabs/node.asp?id=2174>, May . Disponível em 01/05/2011. Citado na pág. 7
- Shannon(1949)** Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, 28:656–715. Citado na pág. 6
- Silbermann(2009)** Michael Silbermann. *Security Analysis of Contactless Payment Systems in Practice*. Tese de Doutorado, Ruhr-Universität Bochum. Citado na pág. 50
- Tan(2009)** Wee Hon Tan. Practical Attacks on the MIFARE Classic. Dissertação de Mestrado, Imperial College London. Citado na pág. 21, 43
- Terada(2008)** Routo Terada. *Segurança de Dados - Criptografia em Redes de Computador*. Edgard Blücher, São Paulo, Brazil, 2st edição. ISBN 9788521204398. Citado na pág. 5, 6, 9

Verdult(2008) Roel Verdult. Security analysis of RFID tags. Dissertação de Mestrado, Radboud University Nijmegen. Citado na pág. 2

Verdult(2009) Roel Verdult. libnfc.org - Public platform independent Near Field Communication (NFC) library. <http://www.libnfc.org/>, 2009. Disponível em 01/05/2011. Citado na pág. 34

Westhues(2007) Jonathan Westhues. A Test Instrument for HF/LF RFID. <http://cq.cx/proxmark3.pl>, 2007. Disponível em 01/05/2011. Citado na pág. 34

Wolfram et al.(2008) Gerd Wolfram, Birgit Gampl, e Peter Gabriel. *The RFID Roadmap: The Next Steps for Europe*. Springer. Citado na pág. 1