Cifra multicanal para maior segurança em redes TCP/IP

Sylvio Ximenez de Azevedo Neto

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Mestrado em Ciência da Computação

Orientador: Prof. Dr. Routo Terada

São Paulo, novembro de 2013

Cifra multicanal para maior segurança em redes TCP/IP

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 30/11/2013. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Routo Terada (orientador) - IME-USP
- Prof. Dr. Marcel Parolin Jackowski - IME-USP
- Prof. Dr. Hellinton Hatsuo Takada - INSPER

Agradecimentos

Agradeço a Deus o sincronismo dos diversos fatores aleatórios que me possibilitaram ingressar, cursar e, principalmente, concluir este programa de mestrado em tempo hábil e sem barreiras intransponíveis.

Em especial, agradeço a admirável paciência nipônica do meu orientador, Prof. Dr. Routo Terada, referência em criptografia, excelente mestre dentro e fora da sala de aula, sempre sereno, exato e aberto à troca de ideias.

Gostaria de mencionar em agradecimento os integrantes do grupo do Laboratório de Segurança de Dados (LSD), do período de 2010 a 2013 , cujo companheirismo, energia e alegria, marcaram-me profundamente. São eles: Prof^{a}. Dr^{a}. Denise Goya, Rafael Will, Cléber Okida, Dionathan Nakamura, Bernardo Magri, Fábio Monteiro, Carlos Lombizani, Reynaldo Cáceres, Wellinton Souza e Ewerton Rodrigues.

E, por fim, agradeço e dedico esse trabalho à minha família, à minha amorosa esposa Karla Liane Teixeira Ximenez de Azevedo, aos meus pais, João Bernardo Ximenez de Azevedo e Luiz Carlos Teixeira, às minhas mães, Cazue Nagata e Lourdes Beti Teixeira, e aos meus avós, Sylvio Ximenez de Azevedo e Maria Therezinha Marques dos Santos de Azevedo, sempre muito presentes em minha vida, são o refúgio da minha consciência, a minha eterna fonte de energia, inspiração, paz e harmonia.

Resumo

AZEVEDO NETO, S. X. **Cifra multicanal para maior segurança em redes TCP/IP**. 2013. 55 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

Neste documento apresentamos uma cifra multicanal inspirada nas técnicas de telecomunicações de espalhamento espectral (*Spread Spectrum*), ou como também conhecido, difusão espectral, para prevenir escuta em uma rede de computadores TCP/IP. Mostramos que dividindo um texto-ilegível *ciphertext* em blocos de tamanho fixo e transmitindo-os aleatoriamente através de diversos canais estabelecidos entre os agentes (transmissor - receptor) é possível aumentar a complexidade da criptanálise por um suposto adversário não autorizado que esteja escutando a comunicação, dessa forma aumentando relativamente a segurança. Mostramos teoricamente que o tempo de quebra da cifra multicanal cresce em ordem fatorial ou exponencial em função do número de canais utilizados na comunicação. Além disso, adaptado a um esquema de difusão binária herdado do AES (*Advanced Encryption Standard*) é possível incrementar a segurança da cifra multicanal para maior resiliência contra a criptanálise diferencial.

Palavras-chave: cifra, rede, segurança, multicanal, protocolo.

Abstract

AZEVEDO NETO, S. X. **Multichannel cipher for higher TCP/IP security**. 2013. 55 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

In this work we present a multichannel cipher inspired on telecommunications Spread Spectrum techniques to prevent eavesdropping in a TCP/IP network. We show that splitting any ciphertext into fixed size blocks and sending them randomly through multiple channels established between the agents (transmitter and receiver) it is possible to increase the complexity of cryptanalysis by a supposed not authorized adversary that might be eavesdropping the communication, in this way, relatively increasing the security. We show theoretically that the brake time of multichannel cipher grows in factorial or exponential order, in function of the number of channel used in the communication. Moreover adapted to binary spread scheme inherited from AES (Advanced Encryption Standard) it is possible to increase the cipher security for more resilience against differential cryptanalysis.

Keywords: cipher, network, security, multichannel, protocol.

Sumário

Lista de Abreviaturas **ix**

Lista de Símbolos **xi**

Lista de Figuras **xiii**

Lista de Tabelas **xv**

1 Introdução **1**
1.1 Objetivos . . . 1
1.2 Justificativa . . . 2
1.3 Hipóteses . . . 2
1.4 Contribuições . . . 2
1.5 Organização do Trabalho . . . 2

2 Espalhamento espectral **3**
2.1 A técnica FHSS . . . 5
2.2 A técnica DSSS . . . 6

3 A cifra multicanal **9**
3.1 Criptografia de chave secreta ou simétrica . . . 9
3.2 Criptanálise e seus tipos . . . 9
3.3 Transposição (Permutação) . . . 10
3.4 Composição . . . 11
3.5 A rede TCP/IP e a interceptação de pacotes . . . 11
3.5.1 O protocolo IP . . . 11
3.5.2 O protocolo TCP . . . 12
3.6 Criptanálise diferencial . . . 13
3.7 A cifra multicanal proposta . . . 15
3.7.1 Geração e troca da chave de sessão k e da sequência aleatória i . . . 16
3.7.2 A demultiplexação e envio de dados através dos múltiplos canais . . . 17
3.7.3 Multiplexação (montagem) dos dados recebidos . . . 18

4 Características da cifra multicanal **21**

5 Metodologia **25**

6 Implementação **29**
6.1 Aplicação de transferência de arquivos 29
6.1.1 Resultados da aplicação de transferência de arquivos 30
6.2 Segurança em relação à criptanálise diferencial 36
6.2.1 Resultados dos experimentos em relação à criptanálise diferencial 39

7 Conclusões **53**
7.1 Sugestões para Pesquisas Futuras 53

Referências Bibliográficas **55**

Lista de Abreviaturas

AES	Padrão de encriptação avançado (*Advanced Encryption Standard*)
BSON	JSON binário (*Binary JSON*)
DES	Padrão de encriptação de dados (*Data Encryption Standard*)
DSSS	Espalhamento espectral por sequência direta (*Direct Sequence Spread Spectrum*)
EOF	Fim de arquivo (*End Of File*)
FHSS	Espalhamento espectral por salto em frequência (*Frequency Hopping Spread Spectrum*)
FIPS	Padrão de processamento de informação federal (*Federal Information Processing Standard*)
FTP	Protocolo de Transferência de Arquivos (*File Transfer Protocol*)
GF	Corpo de Galois (*Galois Field*)
GSM	Sistema global para comunicações móveis (*Global System for Mobile Communications*)
HTTP	Protocolo de transferência de hipertexto (*Hypertext Transfer Protocol*)
HTTPS	Protocolo seguro de transferência de hipertexto (*Hypertext Transfer Protocol Secure*)
IID	Independente e Identicamente Distribuída (*Independent and Identically Distributed*)
IP	Protocolo de Internet (*Internet Protocolo*)
ISO	Organização Internacional para Padronização (*International Organization for Standardization*)
JSON	Notação de objetos javascript (*Javascript Object Notation*)
MP3	MPEG-1/2, camada de audio 3 (*MPEG-1/2, Audio Layer 3*)
MPEG	Grupo de Especialistas em Imagens com Movimento (*Moving Picture Experts Group*)
NIST	Instituto Nacional de Padrões e Tecnologia (*National Institute of Standards and Technology*)
OSI	(Modelo de) Interconexão de sistemas abertos (*Open Systems Interconnection*)
PDF	Formato de documento portátil (*Portable Document Format*)
RF	Frequência de Rádio (*Radio Frequency*)
SMP	*S-Box*/deslocamento de *Bytes*, mistura de colunas e permutação (*S-Box/ShiftBytes, MixColumns and Permutation*)
SMTP	Protocolo de transferência de correio simples (*Simple Mail Transfer Protocol*)
TCP	Protocolo de controle de transmissão (*Transmission Control Protocol*)
TLS	Proteção de camada de transporte (*Transport Layer Security*)
USRP	Periférico de rádio de software universal (*Universal Software Radio Peripheral*)
WCDMA	Acesso múltiplo por código-divisão de banda larga (*Wide-Band Code-Division Multiple Access*)
WWW	Rede mundial de computadores (*World Wide Web*)

Lista de Símbolos

$\oplus$	Operação binária ou-exclusivo
$f_k(x)$	Função de criptografia de um algoritmo qualquer
$f_k^{-1}(y)$	Função de decriptografia de um algoritmo qualquer
k	Chave criptográfica, segredo utilizado pelas funções de criptografia e decriptografia
x	Texto-legível qualquer, entrada da função criptográfica, ou resultado da operação de decriptografia
y	Texto-ilegível qualquer, resultado da operação de criptografia ou entrada da função decriptográfica
Δx	Número de diferença binária entre dois blocos de entrada distintos x' e x''
Δy	Número de diferença binária entre dois blocos de saída distintos y' e y''

Lista de Figuras

2.1 Representação de um sinal antes e depois da aplicação da técnica de espalhamento espectral em termos de densidade energética. 4
2.2 Representação da modulação FHSS em rádio-frequência. 5
2.3 Representação do FHSS - Frequência por Tempo. 6
2.4 Representação da modulação DSSS em rádio-frequência. 6
2.5 Simulação da operação de modulação do DSSS através da operação binária xor. . . . 6
2.6 Simulação da operação de demodulação do DSSS através da operação binária xor. . . 7

3.1 Representação de um canal de comunicação seguro, estabelecido pelo uso de um algoritmo de criptografia (cifra) simétrico. 9
3.2 Representação do cabeçalho IP. 12
3.3 Representação do cabeçalho TCP. 13
3.4 Ilustração do processo de abertura de múltiplas conexões *socket* TCP/IP. 14
3.5 Representação da obtenção dos deltas (Δx e Δy) para a criptanálise diferencial. . . . 15
3.6 Representação da interceptação normal de um canal seguro. 15
3.7 Representação da comunicação pela cifra multicanal, interceptada por um criptanalista. 16
3.8 Comunicação multicanal, com um canal específico para a troca de chaves. 17
3.9 Geração e troca da chave de sessão k e da sequência aleatória i. 18
3.10 Envio de dados em esquema multicanal. 19
3.11 Remontagem dos dados. 19

4.1 Distribuição de números aleatórios de 1 a 50, pela classe [java.security.SecureRandom] da plataforma Java. 23
4.2 Distribuição de números aleatórios de 1 a 500, pela classe [java.security.SecureRandom] da plataforma Java. 23
4.3 Crescimento de uma função fatorial 24

5.1 Alteração de *bits* seguindo a combinação através dos k agrupamentos. O valores mais claros na ilustração representam os *bits* alterados. 27

6.1 Tempos de inicialização para diferentes números de canais. 31
6.2 Tempos de transferência, *overhead* e total da comunicação para um arquivo de 100 *Bytes*. 32
6.3 Tempos de transferência, *overhead* e total da comunicação para um arquivo de 40 *kBytes*. 32

6.4 Tempos de transferência, *overhead* e total da comunicação para um arquivo de 500 *kBytes*. 33
6.5 Tempos de transferência, *overhead* e total da comunicação para um arquivo de 1.1 *MBytes*. 34
6.6 Tempos de transferência, *overhead* e total da comunicação para um arquivo de 6.6 *MBytes*. 35
6.7 Exemplo mostrando que a operação de permutação não causa difusão binária. 37
6.8 Representação da cifra SMP para a operação criptográfica. 39
6.9 Representação da operação inversa da cifra SMP. 40
6.10 Total de *bits* alterados para a permutação binária. 40
6.11 Total de *bits* alterados para a permutação de blocos de 4 *bits*. 41
6.12 Total de *bits* alterados para a permutação de blocos de 8 *bits*. 41
6.13 Total de *bits* alterados para a permutação de blocos de 16 *bits*. 42
6.14 Total de *bits* alterados para a permutação de blocos de 32 *bits*. 42
6.15 Total de *bits* alterados para a permutação de blocos de 64 *bits*. 43
6.16 Estatística dos *bits* alterados para a permutação binária. 44
6.17 Estatística dos *bits* alterados para a permutação de blocos de 4 *bits*. 44
6.18 Estatística dos *bits* alterados para a permutação de blocos de 8 *bits*. 45
6.19 Estatística dos *bits* alterados para a permutação de blocos de 16 *bits*. 45
6.20 Estatística dos *bits* alterados para a permutação de blocos de 32 *bits*. 45
6.21 Estatística dos *bits* alterados para a permutação de blocos de 64 *bits*. 46
6.22 Coeficiente de variação (desvio padrão pela média) para a permutação binária. . . . 47
6.23 Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 4 *bits*. 47
6.24 Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 8 *bits*. 48
6.25 Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 16 *bits*. 48
6.26 Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 32 *bits*. 48
6.27 Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 64 *bits*. 49
6.28 Média de *bits* trocados por posição em permutação binária. 50
6.29 Média de *bits* trocados por posição em permutação de blocos de 4 *bits*. 50
6.30 Média de *bits* trocados por posição em permutação de blocos de 8 *bits*. 51
6.31 Média de *bits* trocados por posição em permutação de blocos de 16 *bits*. 51
6.32 Média de *bits* trocados por posição em permutação de blocos de 32 *bits*. 51
6.33 Média de *bits* trocados por posição em permutação de blocos de 64 *bits*. 52

Lista de Tabelas

3.1 Exemplo de transposição. 11

6.1 Tempos de transmissão de dados de um arquivo de 100 *Bytes* (ms). 31
6.2 Tempos de transmissão de dados de um arquivo de 40 *kBytes* (ms). 31
6.3 Tempos de transmissão de dados de um arquivo de 500 *kBytes* (ms). 33
6.4 Tempos de transmissão de dados de um arquivo de 1,1 *MBytes* (ms). 33
6.5 Tempos de transmissão de dados de um arquivo de 6,6 *MBytes* (ms). 34
6.6 Tempos de transmissão de dados em um abordagem serializada normal (ms). 35
6.7 Tabela de constantes de deslocamento em função do tamanho do bloco de entrada do AES. 38

Capítulo 1

Introdução

No GSM (*Global System for Mobile Communications*), padrão de tecnologia para as redes de telefonia celular, além dos protocolos e algoritmos de segurança que autenticam os usuários à rede e criptografam os dados de voz e serviços que são transmitidos no ar, existe uma outra proteção não tão aparente que dificulta muito a interceptação e, consequentemente, a obtenção das informações que trafegam pelo ar, é a técnica FHSS (*Frequency Hopping Spread Spectrum*), espalhamento espectral por salto em frequência, uma técnica de transmissão onde os dados da informação são divididos e transmitidos separadamente em diversos canais de rádio frequência. Com a utilização dessa técnica, embora existam diversos trabalhos acadêmicos e técnicos que comprovam que os algoritmos criptográficos utilizados pelo GSM não sejam tão seguros, é, ainda, relativamente difícil encontrar ataques práticos que de fato consigam quebrar o fluxo cifrado das informações (voz e dados) do GSM.

Essa dificuldade é devida ao custo de se montar uma infraestrutura de escuta em diversos canais e pela posterior dificuldade de obtenção, identificação e ordenação dos blocos do fluxo de dados transmitido.

A ideia deste trabalho é estudar as características da técnica FHSS aplicada à criptografia no contexto da segurança de uma rede de computadores TCP/IP. Verificando se é viável o seu uso e qual seria o seu ganho em segurança.

Para esse estudo propomos uma cifra genérica de transmissão multicanal, inspirado nas estratégias principais de espalhamento de espectro existentes, dividindo uma informação (um texto-ilegível, criptografado) a ser transmitida, pelo número de canais (sockets TCP/IP) estabelecidos previamente entre um transmissor e receptor, e transmitindo seus blocos de dados aleatoriamente por estes canais.

Uma implementação de uma aplicação de transferência de arquivos utilizando esta cifra proposta é apresentada, e da qual é possível extrair informações de *overhead* [1], tempos de processamento, taxa de transferência e outras propriedades dessa cifra. Além disso, um estudo empírico é apresentado para testar a segurança da cifra multicanal em relação a criptanálise diferencial, uma técnica muito conhecida, apresentada pela primeira vez por Biham e Shamir (1991).

1.1 Objetivos

O objetivo dessa pesquisa é verificar as propriedades do espalhamento da informação em múltiplos canais como uma função criptográfica, de forma a aumentar a segurança de um sistema de comunicação.

[1] Dado a administração de múltiplos canais e encapsulamento de protocolo.

1.2 Justificativa

Até o momento, a utilização de múltiplos canais na criptografia tem se limitado ao uso em protocolos, em geral, utilizando canais heterogêneos (e.g. Takács (2007), Michtchenko e Vilnaski (2007)) para dificultar o acesso do adversário ao segredo (chave), utilizado na operação criptográfica. Não foram encontrados trabalhos fora do âmbito das telecomunicações que utilizasse o espalhamento da informação como forma de segurança.

Com a computação em nuvem (*cloud computing*), com o poder dos chips de múltiplos núcleos e máquinas virtuais cada vez mais populares, a cifra multicanal pode facilmente ser um vetor a mais de segurança entre a comunicação dos sistemas que rodam nesses ambientes.

1.3 Hipóteses

As seguintes hipóteses serão consideradas e verificadas ao longo desse trabalho:

- A possibilidade de simular as características do espalhamento espectral em um esquema multicanal em comunicações TCP/IP, em uma rede de computadores;
- A distribuição aleatória de blocos de uma informação (texto-ilegível) em múltiplos canais (cifra multicanal) pode aumentar a segurança da transmissão;
- Existe um gerador de números aleatórios de distribuição uniforme e de uso prático em computadores convencionais para a implementação da cifra multicanal;
- A cifra multicanal é mais resistente à criptanálise diferencial.

1.4 Contribuições

As principais contribuições deste trabalho:

- A proposta de uma cifra multicanal baseada em uma técnica de espalhamento espectral, para aumento de segurança nas comunicações TCP/IP;
- Uma análise da cifra multicanal proposta em relação à criptanálise diferencial;
- Um método de difusão binária que pode ser aplicado em outras cifras e sistemas criptográficos, incluindo o GSM, uma tecnologia de rede de celulares.

1.5 Organização do Trabalho

No Capítulo 2 apresentamos os conceitos e as características das principais técnicas de espalhamento espectral, nas quais se basearam a concepção da cifra multicanal. No Capítulo 3, fazemos uma revisão dos conceitos fundamentais de criptografia relacionados ao trabalho e detalhamos a nossa proposta da cifra multicanal. No Capítulo 4 mostramos as características e as propriedades da cifra multicanal proposta, introduzindo as questões do aumento de segurança esperado. No Capítulo 5 detalhamos a metodologia empregada neste trabalho. No Capítulo 6 detalhamos duas implementações experimentais, uma da cifra multicanal em uma aplicação de transferência de arquivos. E a outra, específica para a análise da cifra multicanal em relação a segurança contra a criptanálise diferencial. Apresentamos os dados resultantes dos experimentos realizados com essas implementações e fazemos uma análise de seu significado.

Finalmente, no Capítulo 7 discutimos algumas conclusões que podemos tirar dos resultados obtidos e apresentamos algumas propostas para trabalhos futuros.

Capítulo 2

Espalhamento espectral

Neste capítulo apresentaremos uma introdução sobre o espalhamento espectral e o funcionamento da técnica FHSS (*Frequency-Hopping Spread Spectrum*) que inspirou o desenvolvimento da cifra multicanal, devido à complexidade que causa na transmissão de dados, o que aumenta a dificuldade de interceptação por um possível adversário, interessado na comunicação.

Técnicas de espalhamento espectral são bem conhecidas e utilizadas em telecomunicações, seu desenvolvimento começa em meados de 1950, principalmente no uso militar, tratam-se de técnicas de transmissão onde um sinal é difundido sobre uma banda de frequência muito mais larga que a banda de fato necessária para a transmissão de uma determinada informação. O espalhamento espectral é realizado por meio de um código, um pseudo-ruído independente, aplicado ao sinal da informação. A aplicação desse código é uma modulação com o objetivo de difundir a energia do sinal de informação, assim difundindo-o em uma largura de banda muito maior do que a do original, segundo Meel (1999). A recepção sincronizada pelo mesmo código da modulação é a forma de remontar a mensagem (*de-spread*) e recuperar a informação no receptor.

São várias as razões para a utilização dessas técnicas, as aplicações iniciais foram as comunicações táticas anti-interferência (*anti-jamming*), sistemas de orientação e sistemas anti-múltiplos-caminhos (*anti-multi-path systems*). Hoje são muitas as aplicações das técnicas de espalhamento espectral, são elas (Fazel e S.Kaizer (2008)):

- Resistência contra ruídos e interferências;
- Baixa probabilidade de interceptação, dificuldade de detecção do sinal de transmissão;
- Múltiplo acesso;
- Recepção multi-caminho (*multi-path*);
- Diversidade de recepção;
- Variação de alta resolução (*high resolution ranging*);
- Tempo universal preciso (*accurate universal timing*).

A figura 2.1 ilustra a difusão da energia empregada em uma transmissão normal e em uma transmissão que utiliza uma técnica de espalhamento espectral qualquer. Na cifra multicanal não nos preocuparemos com a questão energética da transmissão e o aspecto do domínio de frequências será transformado em uma comunicação paralela.

Existem algumas técnicas de espalhamento espectral, as duas mais importantes [1] utilizadas são:

[1] Em termos das tecnologias mais utilizadas, já que são utilizadas nos padrões de comunicação sem fio: *Bluetooth*, *Wifi* e nas redes digitais modernas de telefonia celular. Existem outras técnicas, como por exemplo, o THSS (*Time-Hopping Spread Spectrum*) e o CSS (*Chirp Spread Spectrum*), mas essas não serão consideradas nesse trabalho, pois são utilizadas em tecnologias muito específicas.

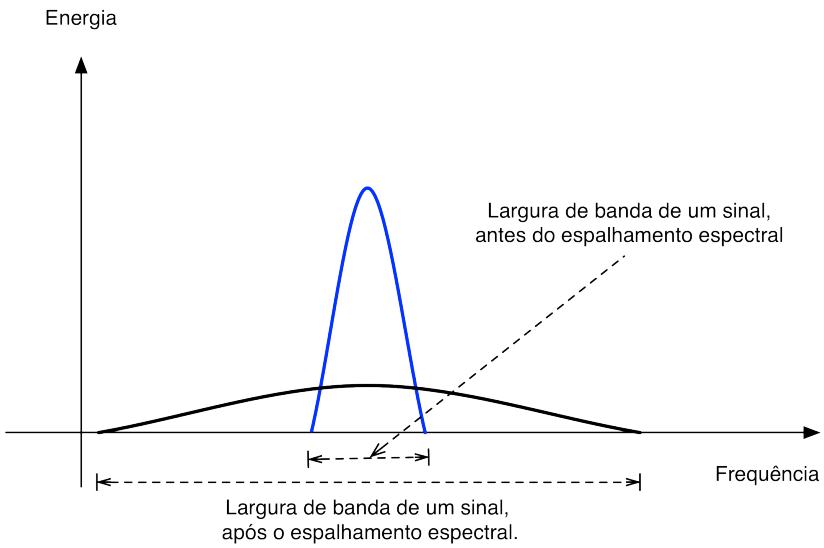

Figura 2.1: *Representação de um sinal antes e depois da aplicação da técnica de espalhamento espectral em termos de densidade energética.*

- DSSS (*Direct Sequence Spread Spectrum*) - Neste método um código pseudo-ruído é multiplicado diretamente no sinal da informação antes da modulação. Esse mecanismo é utilizado pelas redes *Wifi* 802.11a/b/g e nas redes de 3ª geração das redes de telefonia com a tecnologia WCDMA (*Wide-band Code Division Multiple Access*).

- FHSS (*Frequency-Hopping Spread Spectrum*, do inglês Espalhamento Espectral por Salto em Frequência) - Neste método o código pseudo-ruído é utilizado na etapa de modulação alterando a frequência da portadora[2], selecionando o uso dos os canais de comunicação. Essa técnica é utilizada no padrão de comunicação *Bluetooth* e nas redes de telefonia GSM.

A tecnologia de telefonia celular GSM utiliza um método de FHSS para evitar a interferência entre antenas e também como método de *interleaving*[3], de acordo com Eberspacher *et al.* (2001). O emprego de tal método dificulta em muito a escuta (*eavesdropping*) de uma comunicação por um adversário mal intencionado, pois ele deve manter toda uma infraestrutura para cada um dos canais envolvidos, identificar e montar na sequência certa (que ele desconhece) o fluxo (*stream*) de comunicação cifrado (texto-ilegível) e, por fim quebrar a cifra da família A5 em que o fluxo foi cifrado, para obter a mensagem original.

A família de algoritmos A5 é a responsável pela privacidade na comunicação das redes GSM, é atualmente composta por três membros: A5/1, A5/2 e A5/3. Todos os três algoritmos tem sido estudados pela academia, com vários trabalhos publicados de criptanálise e de melhoria de segurança.

O A5/2 foi totalmente quebrado pelo trabalho de Barkan *et al.* (2006), já o A5/1 tem chave muito pequena, e através do poder computacional dos tempos atuais e associado às estratégias de ataque como as *rainbow tables* (*time-tradeoff*) é possível encontrar a chave em tempo relativamente curto, conforme Nohl (2009). E mesmo para o A5/3, o padrão mais moderno, já existem alguns ataques relevantes, segundo Dunkelman *et al.* (2010), embora ainda sem implementações práticas.

Mas a questão é que embora haja uma certa fragilidade nos protocolos de segurança GSM (utilizando os algoritmos A5/1 ou A5/3), ainda é muito difícil realizar uma quebra prática e real do padrão, contras as operadoras de telefonia celular existentes, os trabalhos baseiam-se em ambientes

[2]Sinal base que possibilita a transmissão, onde o sinal de informação é modulado. Nas transmissões de rádio é uma onda eletromagnética na faixa aproximada de 3kHz até 240GHz, a radiofrequência.

[3]Técnica usada principalmente em telecomunicações de intercalação aleatória dos dados para evitar a perda de contexto de uma mensagem transmitida por rádio.

simulados, e não existem muitas ferramentas implementadas e difundidas, a ferramenta que mais se aproxima de tal objetivo é o projeto *Airprobe* [4], um *framework* para a interceptação e escuta do GSM, mas está descontinuado desde 2009 e apresenta sérias limitações, foi desenvolvido para uma plataforma de hardware muito específica, documentação desatualizada e insuficiente e que requerem um conhecimento razoável e específico de rádio telecomunicações.

Essas limitações se explicam pela dificuldade em se lidar com altas frequências de rádio da faixa em que o GSM está localizado, normalmente com o uso de equipamentos específicos, e com um custo considerável[5]; pela complexidade da demodulação do sinal GSM, que requer um conhecimento em telecomunicações e em eletrônica considerável; pelo caráter estritamente acadêmico das maioria dos trabalhos publicados; e, pelo que nos interessa aqui, a estratégia de FHSS empregada em seu canais de comunicação, que ao quebrar e espalhar o fluxo da comunicação, aumenta considerável e diretamente os custos e a complexidade da interceptação e da reconstrução do fluxo de dados de uma determinada comunicação. É o caso justamente do projeto *Airprobe*, que analisa somente os canais que não sofrem a ação dessa estratégia, normalmente os canais de controle, ou comunicações em ambiente controlado (laboratório com estações rádio-base no controle do pesquisador).

A seguir detalhamos as técnicas FHSS e o DSSS.

2.1 A técnica FHSS

Na técnica FHSS a informação é dividida em pacotes e transmitida ao longo do espectro do domínio de frequências. O salto (*hopping*) é a troca da frequência do sinal da portadora, onde o sinal da informação é modulado. O código (pseudo-ruído) é utilizado para selecionar a frequência do sinal da portadora. A figura 2.2 ilustra o processo de modulação de um sinal utilizando a técnica FHSS.

Os pacotes de informação a cada momento são modulados em frequências diferentes, em uma representação da transmissão do tempo pela frequência, parecem saltar de um canal a outro, daí o seu nome (figura 2.3).

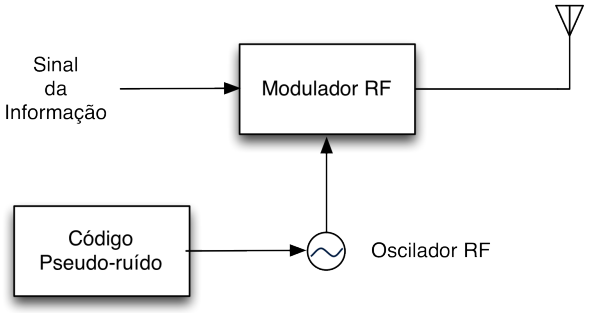

Figura 2.2: *Representação da modulação FHSS em rádio-frequência.*

Essa característica de salto, pode ser simulada na rede TCP/IP através do estabelecimento de múltiplos canais de comunicação (*sockets*) e da divisão e transmissão aleatória da informação por esses canais. Repare que a técnica FHSS não causa um embaralhamento no tempo, somente na de frequência, na simulação TCP/IP, o salto é feito pela seleção do *socket* onde o pacote será transmitido, no lugar da frequência. Essa simulação, utilizando pacotes TCP/IP, é a base da cifra multicanal apresentada nesse trabalho.

[4]Fonte: https://svn.berlin.ccc.de/projects/airprobe/, acessado pela última vez em 30/07/2013

[5]Embora existam hoje em dia algumas opções mais acessíveis - o USRP *Universal Software Radio Peripheral*, http://www.ettus.com, com custo em torno de US$ 1.000,00.

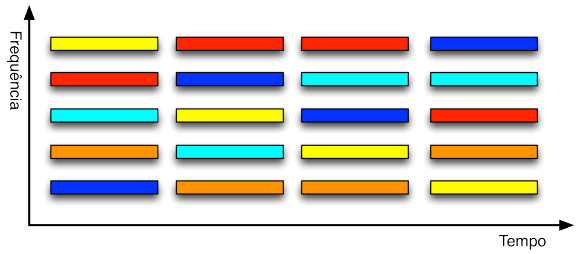

Figura 2.3: *Representação do FHSS - Frequência por Tempo.*

2.2 A técnica DSSS

Na técnica DSSS, um sinal aleatório, o código pseudo-ruído, é multiplicado ao sinal da informação diretamente ao longo do tempo[6]. Esse sinal aleatório representa uma sequência de valores 1 e -1, em uma frequência muito maior que o sinal da informação. A informação pode ser recuperada aplicando um sinal que representa o código de modulação inverso, uma sequencia inversa de valores 1 e -1. A figura 2.4 ilustra o esquema dessa técnica de modulação.

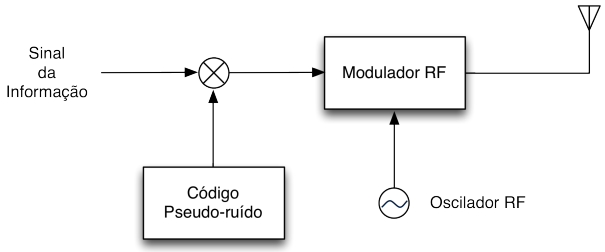

Figura 2.4: *Representação da modulação DSSS em rádio-frequência.*

A característica da aplicação de um código ruído pode ser simulada em uma operação binária *xor* ($\oplus$, ou-exclusivo), combinando um vetor de bits x, representando a informação, com um outro vetor de bits k de tamanho $n_k = v.n_x$, onde v representa a proporção do aumento da "frequência" desejada e n_x o tamanho do vetor x. Nas figuras 2.5 e 2.6, exemplificamos isso nas operações de modulação e demodulação, respectivamente.

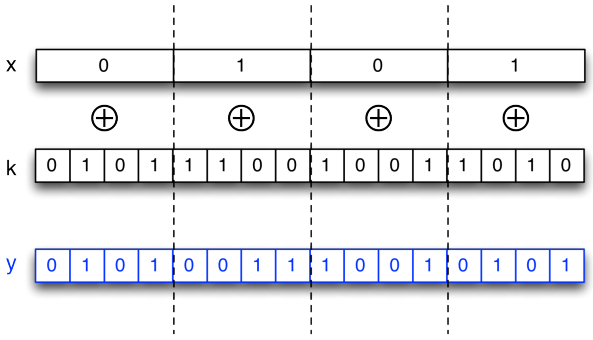

Figura 2.5: *Simulação da operação de modulação do DSSS através da operação binária xor.*

A princípio consideramos utilizar essa técnica também na cifra multicanal, mas nos exemplos do DSSS pode-se reparar que há um aumento expressivo da informação e também de geração de redundância dos valores binários, por esse motivo, apesar de interessante, ignoramos essa técnica nesse trabalho.

[6]Em sinais de rádio audíveis, o sinal resultante lembra um ruído branco, um som de estática.

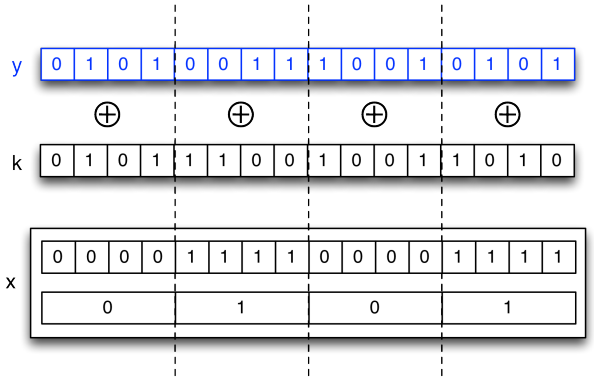

Figura 2.6: *Simulação da operação de demodulação do DSSS através da operação binária xor.*

Neste capítulo apresentamos uma introdução sobre as técnicas de espalhamento espectral, detalhamos o funcionamento das duas principais o DSSS e o FHSS. Em particular, a técnica FHSS, possui uma característica interessante que queremos reproduzir na cifra multicanal.

Capítulo 3

A cifra multicanal

Neste capítulo fazemos uma revisão sobre os conceitos de criptografia, dos tipos de criptanálise e das redes TCP/IP, na sequência, apresentamos a nossa proposta da cifra multicanal e seu esquema geral de funcionamento.

Os conceitos de criptografia, os tipos de criptanálise e as definições de transposição e composição de funções criptográficas detalhados a seguir foram extraídos do livro Terada (2008).

3.1 Criptografia de chave secreta ou simétrica

Em um modelo de comunicação onde um agente A (Alice) comunica-se de forma segura com um agente B (Beto) (figura 3.1), conforme representação da figura 3.1. Alice com a chave k, criptografa um *texto legível* x (texto original, ou em inglês *plaintext* obtendo um *texto ilegível*, $f_k(x) = y$. O texto y é transmitido para o computador de destino do Beto, onde y é decriptografado pelo algoritmo inverso do utilizado por Alice, $f_k^{-1}(y)$, obtendo-se x se e só se o destinatário Beto conhecer a chave k. Para quem desconhece a chave k, um possível adversário, um agente C interceptador (o Criptanalista Carlos), é computacionalmente difícil obter-se x somente a partir do conhecimento de y, isso se o algoritmo $f(x)$ for bem projetado, seguro.

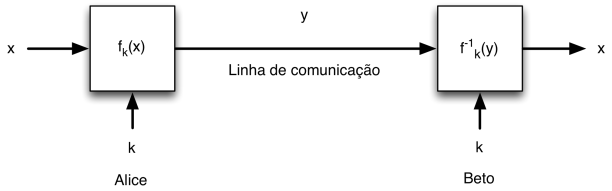

Figura 3.1: *Representação de um canal de comunicação seguro, estabelecido pelo uso de um algoritmo de criptografia (cifra) simétrico.*

3.2 Criptanálise e seus tipos

Ao analisar a segurança de um algoritmo criptográfico, o criptanalista Carlos tem o objetivo de quebrar [1] um texto ilegível y interceptado, ou quebrar a chave secreta k.

Os tipos de ataque que Carlos pode realizar são:

1. *Ataque por só-texto-ilegível*: O criptanalista Carlos tenta adquirir conhecimento útil à quebra analisando um ou mais ilegíveis y. Se este tipo de ataque for computacionalmente viável o algoritmo em questão é considerado totalmente inseguro e inútil, quebrado.

[1] O termo "quebrar"aqui tem o significado de atingir o objetivo, contra um texto-ilegível, significa descobrir o texto-legível original. Contra a chave secreta, significa descobrí-la.

2. *Ataque por texto legível conhecido*: O criptanalista Carlos [2] possui e analisa pares (x, y) de texto legível e ilegível correspondentes.

3. *Ataque por texto legível escolhido*: Além do descrito no ataque anterior (texto legível conhecido), o criptanalista Carlos pode escolher os legíveis x e obter os y correspondentes. Ele vai escolher x que apresente alguma característica estrutural que aumente o seu conhecimento do algoritmo e da chave em uso. Com o conhecimento adquirido ele pode deduzir o legível correspondente a um ilegível novo.

4. *Ataque adaptativo por texto legível escolhido*: Além do descrito no ataque anterior (texto legível escolhido), a escolha de um novo x pelo criptanalista Carlos pode depender dos ilegíveis $\overline{y}$ analisados anteriormente. Desta forma a escolha de um novo x é condicionada ao conhecimento já adquirido pela análise dos $\overline{y}$ já analisados.

5. *Ataque por texto ilegível escolhido*: O criptanalista Carlos escolhe inicialmente o ilegível y e então obtém o legível x correspondente. Supõe-se que Carlos só tenha acesso ao algoritmo de decriptografia (sem acesso à chave secreta k) e o seu objetivo é, mais tarde, sem ter mais acesso à decriptografia, ser capaz de deduzir x correspondente a um y novo.

6. *Ataque adaptativo por texto ilegível escolhido*: Além do descrito no item anterior (texto ilegível escolhido), a escolha de um novo y pelo criptanalista Carlos pode depender dos ilegíveis $\overline{y}$ analisados anteriormente. Dessa forma a escolha de um novo y é condicionada ao conhecimento já adquirido pela análise dos $\overline{y}$ já analisados.

Ainda, segue abaixo, outros tipos de ataques que podem ser aplicáveis em outros cenários e que podem ser compostos com os ataques descritos acima ou entre si:

1. *Ataque por chaves conhecidas*: O criptanalista Carlos conhece algumas chaves já usadas e utiliza tal conhecimento para deduzir chaves novas.

2. *Ataque por replay*: O criptanalista Carlos grava as comunicações legítimas entre Alice e Beto, e depois usa parte da gravação para o seu proveito. Se a parte da gravação usada não é alterada, este ataque é chamado de *passivo*, caso contrário, de *ativo*.

3. *Personificação*: O criptanalista Carlos simula um outro usuário legítimo. Por exemplo, toma o lugar de Alice sem que Beto consiga notar qualquer diferença.

4. *Ataque por dicionário*: Muito utilizado para deduzir senhas quando o criptanalista Carlos tem acesso ao arquivo de senha legítimas criptografadas; Carlos calcula antecipadamente as senhas criptografadas y correspondentes às senhas x mais comuns (Ex: nome de pessoas, números de telefones, data de nascimento, e suas combinações, etc.) e compara com os y calculados e armazenados no arquivo de senhas legítimas; se encontrar um y igual no arquivo, ele já possui a senha x válida correspondente.

3.3 Transposição (Permutação)

Trata-se de uma cifra onde uma chave secreta k é uma permutação $c_1, c_2, ..., c_n$ dos inteiros de 1 a n. Seja $x = \{x_1, x_2, ..., x_n\}$ um texto legível , onde x_j representa cada dígito ou letra de x. A transposição consiste em transpor cada x_j na posição j para a posição c_j.

Por exemplo, para $n = 4$ o texto legível $x =$ "$SATURNOS$" é criptografado para $y =$ "$AUSTNSRO$", conforme a tabela 3.1. Observer que a chave $\{3, 1, 4, 2\}$ é utilizada duas vezes.

[2] Nesse e nos tipos de ataque a seguir, supomos que o criptanalista tem pleno acesso ao algoritmo (sem conhecer a chave secreta k) e não é necessariamente um adversário, uma pessoa mal-intencionada ou um intruso, mas pode ser um especialista que objetiva descobrir as vulnerabilidades do algoritmo, objeto do estudo, que fora projetado por outra pessoa, eventualmente.

posição j	1	2	3	4	1	2	3	4
legível	S	A	T	U	R	N	O	S
chave	3	1	4	2	3	1	4	2
ilegível	A	U	S	T	N	S	R	O

Tabela 3.1: *Exemplo de transposição.*

O número total de chaves possíveis é $n!$ (i.e. fatorial de n). Mas essa cifra preserva a frequência das letras se x for uma língua natural, é então sensível a inferências estatísticas.

Para um texto legível x binário, x_j é no mínimo um bit ou um conjunto de bits previamente estabelecidos. Neste caso a inferência estatística é mais difícil desde que o x_j binário não apresente um padrão de codificação conhecido (ASCII ou EBCDIC, por exemplo).

A operação de transposição está associada com a ideia de difusão, que segundo Menezes *et al.* (1996), refere-se a reorganização ou espalhamento dos *bits* em uma mensagem de tal forma que qualquer redundância no *texto legível* seja difundida no *texto ilegível*.

3.4 Composição

Sejam dadas duas funções f e g tais que $f(x) = t$ e $g(t) = y$, com $x \in A, t \in B, y \in C$.

A composição gf das funções f e g é definida por $gf(x) = g(f(x)) = y$.

A maioria dos algoritmos criptográficos modernos consiste em uma composição de várias funções f_j para obter $y = f_n f_{n-1} ... f_2 f_1 = F(x)$. Assim, mesmo que cada uma das f_j não seja isoladamente segura, a função composta $F(x)$ é relativamente mais segura.

Em particular, a cifra multicanal proposta nesse trabalho pode ser considerada como a aplicação uma segunda função criptográfica no sistema, com o objetivo de deixá-lo mais seguro, até mesmo em aqueles que usam cifras já fragilizadas por uma criptanálise qualquer.

3.5 A rede TCP/IP e a interceptação de pacotes

O par de protocolos TCP/IP é hoje um dos mais importantes do mundo, está presente no dia a dia de bilhões de pessoas e é parte integrante da economia global. É sobre esse par que estão alguns dos mais utilizados e famosos protocolos, é o caso do HTTP(S) (*HyperText Transfer Protocol (Secure)*), o protocolo da WWW (*World Wide Web*), o SMTP (*Simple Mail Transfer Protocol*) para o envio de e-mails, o FTP (*File Transfer Protocol*) para a troca de arquivos, e muitos outros.

Não é escopo desse trabalho detalhar o funcionamento da rede TCP/IP, só apresentaremos o fundamental para o entendimento da interceptação dos dados dos canais de comunicação estabelecidos por esse protocolo.

Em relação ao modelo OSI (*Open System Interconnection*, modelo de rede conceitual, ISO/IEC 7498-1), os protocolos TCP e IP, correspondem às camadas 4, de transporte, e 3, de rede, respectivamente. A seguir, um pouco mais de detalhe sobre os protocolos.

3.5.1 O protocolo IP

O IP (*Internet Protocol*) é um protocolo de rede responsável por entregar pacotes de dados de uma fonte (*source*) para a um destinatário (*target*), agentes dessa rede de comunicação que são identificados por um endereço IP (*IP Address*[3]), uma das principais informações do procolo. O

[3]Número de identificação das máquinas possui atualmente duas versões: IPV4 e IPV6, na primeira os números são organizados por 4 grupos de octetos (1 *byte*) representados em decimal no intervalo de 0 a 255, e na segunda, a versão mais moderna, os números de identificação são organizados em oito grupos de 2 *bytes*, representados em 4 dígitos em hexadecimal. A revisão do protocolo IP, que acabou gerando a versão IPv6, foi necessária devido ao explosivo crescimento da internet, que alcançou o limite de combinações de números possíveis da versão anterior IPv4, à saber,

protocolo IP não garante uma entrega assegurada dos pacotes, não possui controle do fluxo de dados, não tem um mecanismo para de recuperação de erros (frequentes, principalmente em *links* de rádio) e não é orientado a conexão, em outras palavras, sem controle de estabelecimento e persistência da conexão. A figura 3.2 representa um cabeçalho IP.

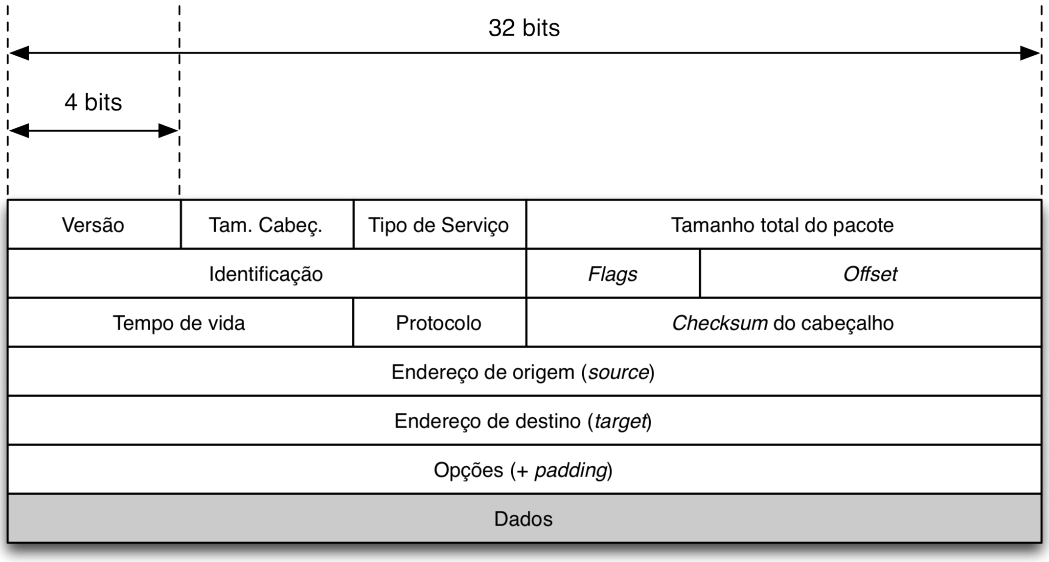

Figura 3.2: *Representação do cabeçalho IP.*

3.5.2 O protocolo TCP

O TCP (*Transmission Control Protocol*, RFC - *Request For Comments* 793) como o nome sugere refere-se a um protocolo de controle da transmissão, isso implica na garantia de entrega dos pacotes (confiança), de forma ordenada e íntegra, garantindo que não há erros nos dados transmitidos (verificação e correção de erros). O TCP é um protocolo de transporte orientado a conexão, uma característica complementar ao IP, que traz robustez ao esquema (TCP/IP), e além disso funciona como um serviço intermediário entre a aplicação e o protocolo IP, pois este último, trabalha com tamanhos de dados específicos de 32 bits, o TCP, quando na transmissão de dados maiores, encapsula as operações de divisão de dados para o encaixe nos pacotes IP. A figura 3.3 representa um cabeçalho TCP.

A cifra multicanal proposta a seguir, utiliza múltiplos canais TCP/IP. A maioria das linguagens de programação modernas provê acesso fácil para a criação desses canais. Cada canal estabelecido é um *socket* (soquete, encaixe, conexão).

Para que ocorra uma comunicação é necessário duas pontas, normalmente programas distintos, um cliente e um servidor.

No desenvolvimento normal do servidor abre-se uma porta de comunicação (*socket*), estado conhecido como *listening* (escuta), esperando que um cliente nela conecte, quando há uma conexão, uma nova *thread* (processo leve, uma nova corrente de processamento) é aberta e o *socket* é transferido para essa *thread*, enquanto servidor volta a ficar em *listening*. Dessa maneira o programa servidor consegue tratar múltiplas conexões. A figura 3.4 ilustra o processo de conexão de múltiplos clientes ao servidor.

O termo multicanal utilizado é justamente por essa característica, trabalha com conexões TCP/IP múltiplas, simultâneas e homogêneas; estabelecidas da maneira descrita acima. O programa cliente mantém normalmente uma só comunicação com o servidor, mas nada impede de haver um programa que controle vários clientes conectados com o servidor.

o número aproximado de 4,3 bilhões de endereços.

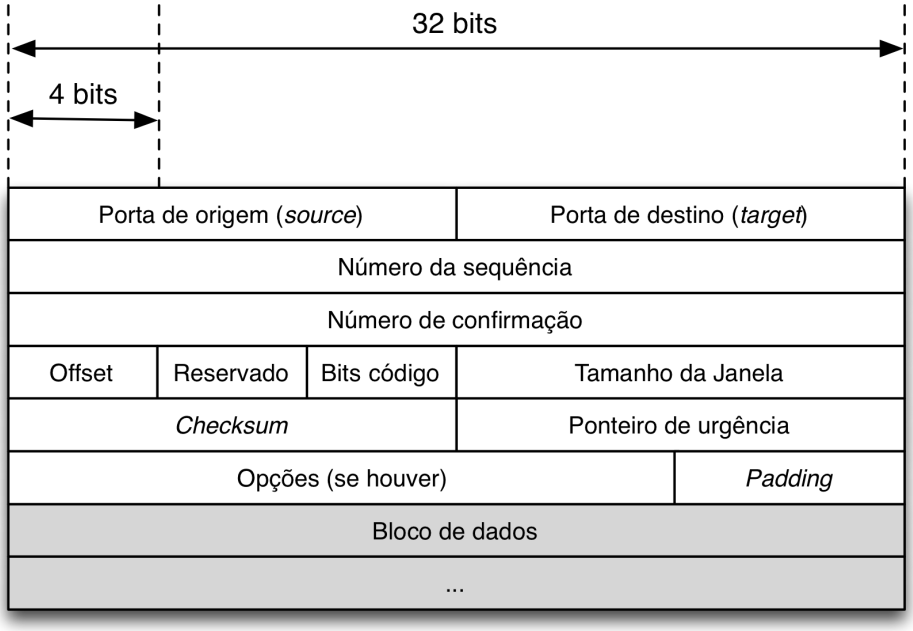

Figura 3.3: *Representação do cabeçalho TCP.*

Nos protocolos de segurança multicanais verificados, onde um exemplo bem concreto é o proposto por Wong e Stajano (2005), normalmente trabalha-se com poucos canais e que são heterogêneos, onde o objetivo é transmitir partes da chave secreta em meios diferentes, de modo a dificultar a obtenção integral dela, por parte de um adversário.

3.6 Criptanálise diferencial

A criptanálise diferencial foi apresentada pela primeira vez por Biham e Shamir na CRYPTO 90 para atacar a cifra DES (*Data Encryption Standard*) que posteriormente virou um livro Biham e Shamir (1991), embora originalmente apresentado para o DES, juntamente com a criptanálise linear, é hoje uma das principais ferramentas de criptanálise para a cifra de bloco. Alguns artigos a apresentam de forma mais genérica, não relacionada uma cifra de bloco em particular, é o caso do tutorial Heys (2001), no qual nos basearemos para explicar esse ataque.

A criptanálise diferencial explora a alta probabilidade de ocorrência de que certas diferenças do texto-legível gerem certas diferenças no texto-ilegível.

Por exemplo, considerando um sistema que tenha como entrada $x = \{x_1, x_2, ...x_n\}$, onde x_i é um bit na posição i de x, e uma saída $y = \{y_1, y_2, ..., y_n\}$, onde y_j é um bit na posição j de y. Considerando duas entradas x' e x'' e duas respectivas saídas y' e y''. A diferença de entrada é dada por $\Delta x = x' \oplus x''$, assim $\Delta x = \{\Delta x_1, \Delta x_2, ..., \Delta x_n\}$, onde cada $\Delta x_i = x'_i \oplus x''_i$, representa a diferença de cada bit de x' e x''. De mesmo modo, $\Delta y = y' \oplus y''$ é a diferença de saída, sendo $\Delta y = \{\Delta y_1, \Delta y_2, ..., \Delta y_n\}$, onde $\Delta y_i = y'_i \oplus y''_i$, representa a diferença binária das saídas.

Em uma cifra aleatória ideal, a probabilidade que uma diferença de saída particular Δy aconteça dado uma particular diferença de entrada Δx é de $\frac{1}{2^n}$, onde n é o número de *bits* de x. A criptanálise diferencial procura explorar um cenário onde um particular Δy ocorre para uma particular diferença de entrada Δx com probabilidade $P_d \frac{1}{2^n}$.

O par de diferenças $(\Delta x, \Delta y)$ é definido como *diferencial* (*differential*).

A criptanálise diferencial é um tipo de ataque de somente texto-legível escolhido, porque supõe que o adversário (criptanalista Carlos) consegue selecionar as entradas e examinar as saídas na tentativa de derivar (encontrar) a chave secreta k. O adversário seleciona dois pares x' e x'', para satisfazer uma particular diferença Δx, sabendo que para aquele Δx, uma Δy ocorre com grande

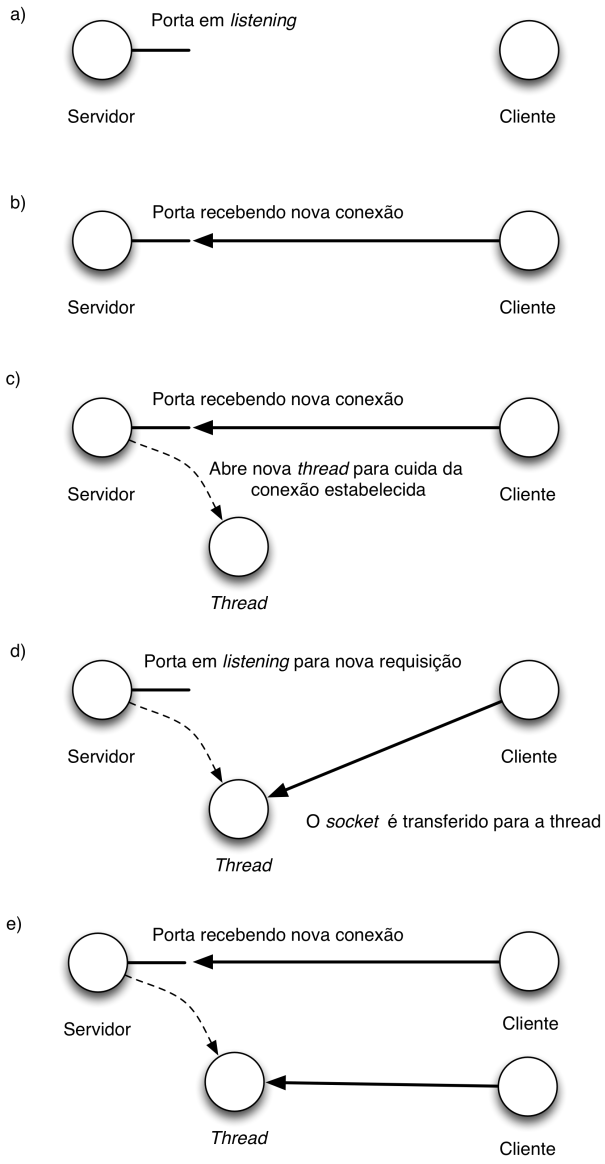

Figura 3.4: *Ilustração do processo de abertura de múltiplas conexões* socket *TCP/IP.*

probabilidade. Eventualmente, montando-se uma tabela de relações entre Δx, uma certa quantidade de chaves k e Δy e fazendo essa análise várias vezes é possível inferir a chave secreta k, que foi utilizada na criptografia, realizando-se muito menos tentativas que a força bruta. A criptanálise diferencial é um procedimento estatístico que procura maximizar a probabilidade de que a chaves escolhidas pelo criptanalista seja a correta.

A força contra a criptanálise diferencial é medida pela relação de números de *bits* alterados entre Δx e Δy. No nosso caso, 4 *bits*[4] são alterados na entrada (Δx) quer se descobrir quantos *bits* são alterados na saída Δy e em quais posições em relação a y'. Se o Δy for baixo, e se os *bits* alterados, mostrarem algum viés, há uma vantagem probabilística para o adversário. Queremos, portanto, que Δy seja tão grande quanto possível.

O trabalho de Biham e Shamir (1991) mostra que aplicado ao DES, o tempo médio necessário para o ataque usando essa cifra é da ordem de 2^{47}, onde pela força bruta o tempo necessário é da ordem de 2^{56}, uma redução considerável.

Testamos a cifra multicanal contra a criptanálise diferencial, onde mostrou-se extremamente frágil, no capítulo 6 apresentaremos os detalhes dessa fragilidade e a forma proposta para superá-la.

[4]Todas as combinações possíveis dos 4 bits.

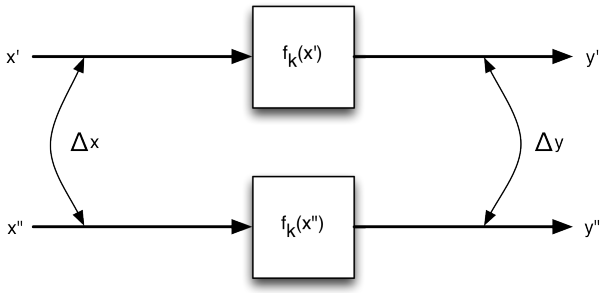

Figura 3.5: *Representação da obtenção dos deltas (Δx e Δy) para a criptanálise diferencial.*

3.7 A cifra multicanal proposta

Uma comunicação TCP/IP segura normalmente é estabelecida criptografando-se um fluxo de dados x (*data stream*) por um algoritmo criptográfico $f_k(x)$, utilizando-se uma chave k, gerando um fluxo de dados criptografado y (texto ilegível, ou cifrado) que irá trafegar em um canal *socket* entre o transmissor (Alice - A) e o receptor (Beto - B). O trabalho do criptanalista (Carlos - C) consiste em capturar esse fluxo de dados criptografado y e, aplicando técnicas de criptanálise, tentar encontrar maneiras de decifrá-lo[5] (figura 3.6).

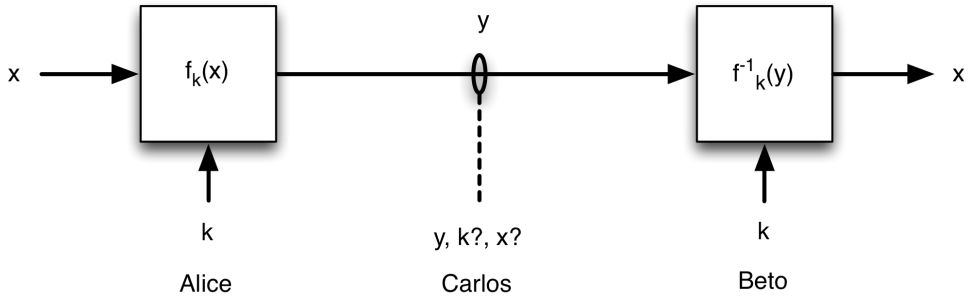

Figura 3.6: *Representação da interceptação normal de um canal seguro.*

O modelo proposto é criar uma comunicação segura com múltiplos canais, n, estabelecidos entre Alice e Beto, enviando nesses canais os blocos criptografados, p, do fluxo de dados, o texto-ilegível y, onde $p_c = y/n$, para $1 \leq c \leq n$. Cada um dos blocos é enviado seguindo uma sequência aleatória i, combinada anteriormente entre Alice e Beto, na figura 3.7 temos uma representação dessa comunicação.

Podemos notar que a dificuldade do criptanalista aumenta, pois agora, além de ter que quebrar o fluxo cifrado y, tem que "adivinhar" a sequência aleatória utilizada na transmissão para a remontagem à forma original dos blocos y interceptados[6].

É esse embaralhamento dos blocos de y enviados por múltiplos canais que estamos chamando de cifra multicanal.

Evidentemente, se pensarmos na cifra de forma serializada, ou seja, após a recuperação do texto-ilegível "y", podemos pensar na sequência i empregada como a aplicação de uma segunda chave, em uma composição de funções. Sendo $g_i(y) = z$ a cifra multicanal e $f_k(x) = y$ uma cifra criptográfica qualquer (vide página 28, Terada (2008)). Nesse caso a cifra $g_i(y)$ representando uma permutação dos blocos y.

Para a realização dos experimentos, na implementação do mecanismo proposto, consideramos ainda a inclusão de um canal seguro TLS (*Transport Layer Security*)[7], para a troca de uma chave

[5] Em Terada (2008) há uma relação dos tipos de ataques possíveis

[6] A interceptação de pacotes é trivial em redes TCP/IP, utilizando-se um *sniffer*.

[7] Para maiores informações acesse a descrição do protocolo no caminho: http://tools.ietf.org/html/rfc5246.

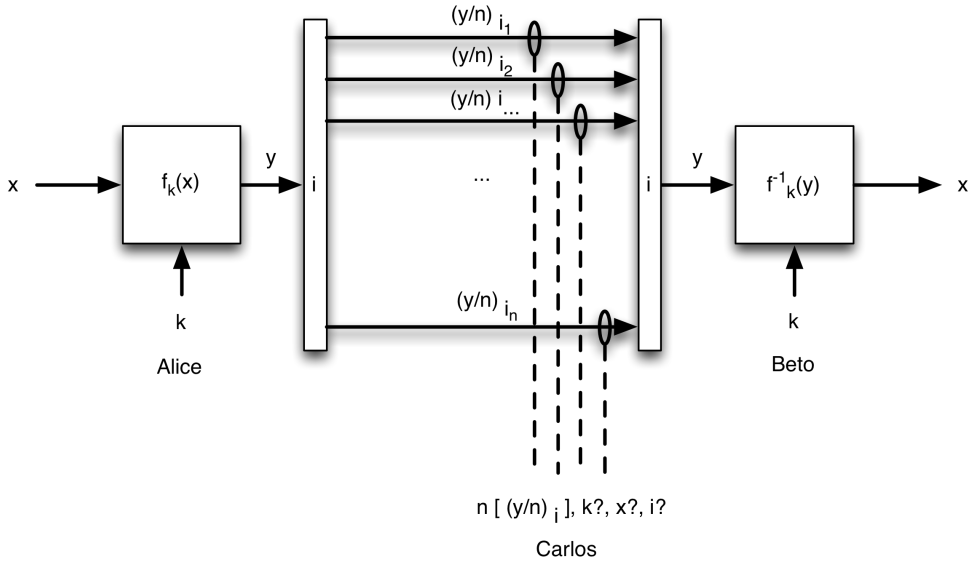

Figura 3.7: *Representação da comunicação pela cifra multicanal, interceptada por um criptanalista.*

de sessão k e a sequência aleatória i, conforme a figura 3.8.

Eventualmente, essa troca de chaves poderia ser concebida e realizada pela própria comunicação multicanal, não foi o caso aqui para facilitar os experimentos.

O esquema mais simples para a implementação de uma comunicação multicanal é o envio direto e sequencial dos blocos p_c do texto ilegível y, a medida que vão sendo criadas.

Na sequência detalhamos o funcionamento, através da sequência de operações e comunicações realizadas pelos agentes Alice e Beto.

3.7.1 Geração e troca da chave de sessão k e da sequência aleatória i

A primeira etapa é a de geração e troca das chave de sessão k e da sequência aleatória i utilizada no envio dos blocos p_n. Aqui utilizamos o canal seguro TLS para enviar as chaves, normalmente essa operação é realizada através de um protocolo de acordo de chaves específico, mas tal como protocolo não é foco desse trabalho, apenas supomos que a transferência foi feita de forma segura e que Alice é capaz de identificar Beto de forma inequívoca.

Abaixo a sequência detalhada dessa etapa:

1. Alice gera uma chave de sessão k, através de uma função geradora *SessionKey()*, com tamanho necessário ao uso da cifra criptográfica $f_k(x)$;

2. Beto conecta-se à Alice através de n canais e envia através do canal TLS seu código de identificação único ($UUID$);

3. Considerando que haja repetição de canais, a partir das n conexões estabelecidas com Beto, Alice gera através da função *RandomSequence()*, uma sequência aleatória, $i = X_1, ..., X_n$ onde X é uma variável aleatória de distribuição uniforme e independente (IID) com valores representando a identificação dos canais no intervalo de $\{0, 1, ..., n-1\}$. Se considerarmos a não repetição de canais, a sequência aleatória i gerada pela função *RandomSequence()*, deve ser composta de n valores aleatórios v distintos, onde $1 \leq v \leq n$ e uniformemente prováveis. A geração dessa sequência pode ser feita através do seguinte algoritmo:

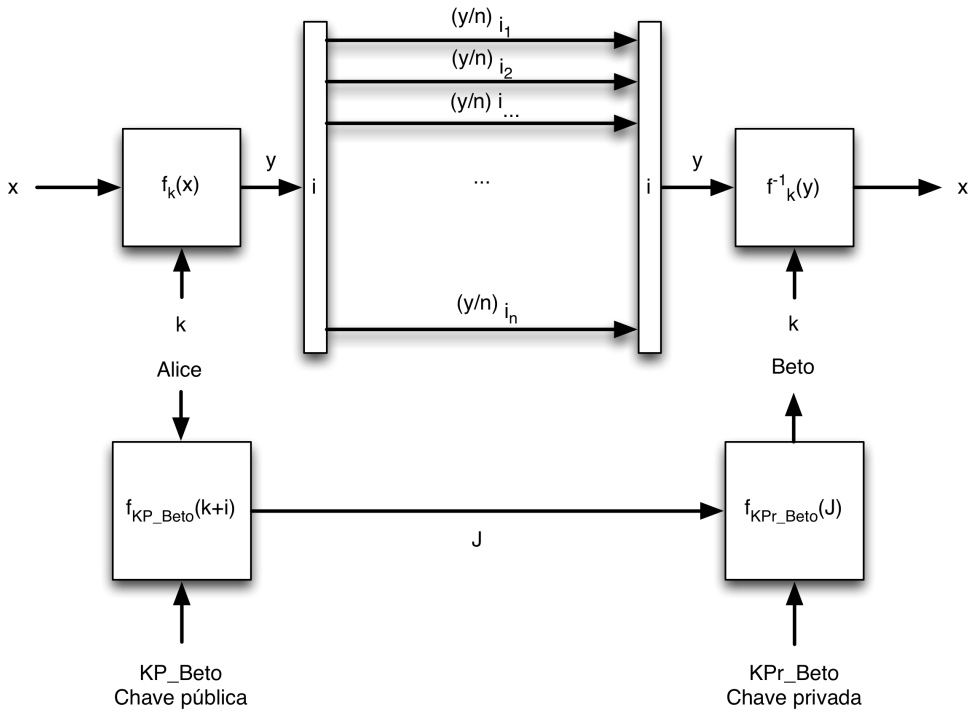

Figura 3.8: *Comunicação multicanal, com um canal específico para a troca de chaves.*

RandomSequence(n)

```
1 i[n]
2 Para p = 1 Até n
3    v = Random(p)
4    i[p] = i[v]
5    i[v] = p
```

A função *RandomSequence(n)* retorna um valor aleatório v de 1 até n. Provar que esse algoritmo gera uma sequência aleatória uniforme não é trivial, mas ele é baseado em um algoritmo conhecido na literatura como *PermuteInPlace* detalhado em Cormen *et al.* (2009), onde está definido o lema 5.5. que prova que o algoritmo realiza uma permutação aleatória uniforme. No algoritmo *RandomSequence* somente acrescentamos ao *PermuteInPlace* a inserção de valores v no vetor de saída i, ao invés de só permutarmos o vetor.

4. Alice então envia a chave de sessão e a sequência aleatória i [8] gerada para Beto.

A figura 3.9 ilustra essa etapa.

3.7.2 A demultiplexação e envio de dados através dos múltiplos canais

Com os canais estabelecidos e chaves trocadas é possível realizar a transmissão dos dados. A transferência é iniciada por Beto que informa a Alice a informação que deseja. Alice, então, com a chave k, criptografa o texto-legível (informação) x requerido por Beto, divide o tamanho do texto ilegível y resultante a ser transmitido pelo número de canais estabelecidos n, e então, envia os blocos p_c para Beto.

Abaixo a sequência detalhada dessa etapa:

1. Beto faz a requisição de dados (*DataRequest*()) a Alice, no canal SSL;

[8] Podemos pensar nessa sequência como uma segunda chave criptográfica. Supondo que haja 256 canais estabelecidos, o tamanho dessa chave seria de 256 *bytes*, ou 2048 bits.

2. Alice criptografa os dados requeridos com uma função criptográfica $f_k(x) = y$;

3. Alice envia então, sequencialmente, em cada um dos canais estabelecidos C_n com Beto, um dos blocos do texto-ilegível p_d, onde d é a identificação de uma parte de y escolhida segundo a sequência aleatória estabelecida i;

4. No final da transmissão, Alice sinaliza Beto do término ($EndOfData()$), no canal SSL.

A figura 3.10 ilustra essa etapa.

3.7.3 Multiplexação (montagem) dos dados recebidos

Uma vez recebidos todos os blocos, o papel de Alice na comunicação encerra-se. Beto com a sequência aleatória i e com os blocos obtidos p_n, reordena os blocos no texto-ilegível y, e com a chave k, o decriptografa em x.

Neste ponto todas as conexões podem ser terminadas, aí depende somente o uso que se queira dar à aplicação.

Abaixo a sequência detalhada dessa etapa:

1. Beto serializa a informação recuperando os dados dos canais utilizando a sequência aleatória i estabelecida em conjunto com Alice;

2. Beto utiliza a função inversa do algoritmo criptográfico $f_k^{-1}(y) = x$, e assim, recupera os dados originais requeridos x.

A figura 3.11 ilustra essa etapa.

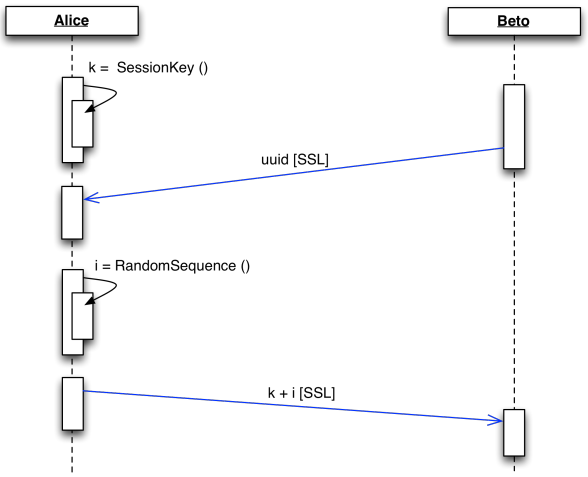

Figura 3.9: *Geração e troca da chave de sessão k e da sequência aleatória i.*

Neste capítulo fizemos uma introdução dos conceitos necessários para o entendimento da cifra multicanal e apresentamos suas características, principalmente o esquema básico de funcionamento entre os agentes de comunicação Alice e Beto.

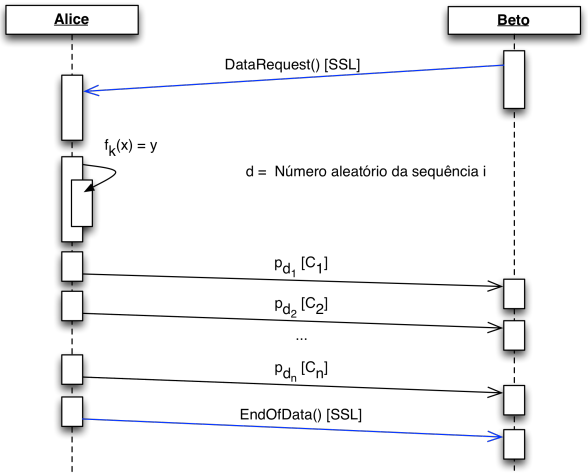

Figura 3.10: *Envio de dados em esquema multicanal.*

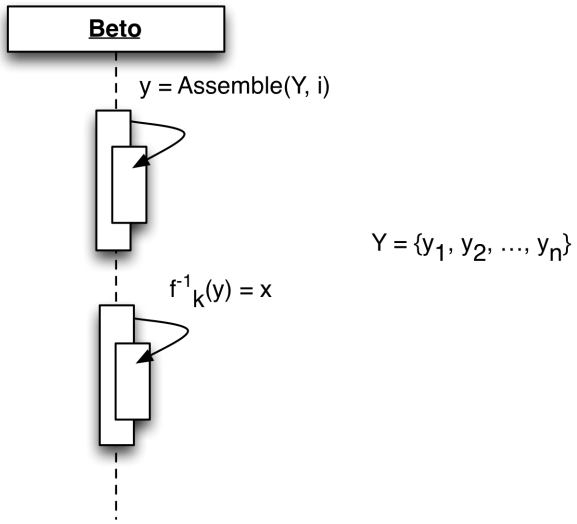

Figura 3.11: *Remontagem dos dados.*

Capítulo 4

Características da cifra multicanal

Após a apresentação da cifra no capítulo anterior, apresentamos aqui as suas características e propriedades, mostramos que pode ser considerada como uma permutação aleatória, onde o número de combinações é razoavelmente grande, de forma que há considerável ganho de segurança em sua utilização.

Se removida a parte da comunicação, a cifra multicanal pode ser reduzida em uma segunda função, em uma composição de funções criptográficas, a aplicação de uma segunda chave, a sequência aleatória. A aplicação dessa segunda chave é meramente uma operação de permutação aleatória que apesar de sua simplicidade permite aumentar a segurança de todo os sistema, e com uma pequena alteração pode aumentar relativamente a segurança contra a criptanálise diferencial conforme estabelecido por Biham e Shamir (1991) (detalhado no capítulo 3).

Supondo que o criptanalista Carlos consiga todas os blocos p do texto-ilegível y transmitidos, o que em uma rede TCP/IP é relativamente fácil, principalmente se considerarmos apenas uma única interface de rede [1] entre Alice e Beto, é trivial que a segurança da cifra multicanal seja toda relacionada com a sequência aleatória i combinada entre as partes (Alice e Beto), pois Carlos teria que verificar cada uma das combinações possíveis dos blocos que obteve até encontrar a correta que representa o texto-ilegível y antes que possa, enfim, analisar para tentar descobrir o texto-original.

Assim, é importante que o gerador da sequência aleatória gere realmente números aleatórios em uma distribuição uniforme. Do contrário, Carlos, poderia ter alguma vantagem estatística para obter a sequência aleatória i.

Supondo que o gerador é realmente aleatório, o número de combinações que o criptanalista Carlos precisa analisar, no pior caso, é de $n!$, se não considerarmos a repetição dos canais envolvidos na comunicação. E n^n, se houver repetição dos canais. Evidentemente a probabilidade de Carlos encontrar a sequencia certa é de $1/n!$ e $1/n^n$, respectivamente, à questão da repetição de canais.

Para o número de 100 canais Carlos tem $7,15.10^{-150}\%$ (considerando a não repetição de canais) como probabilidade de acerto ao acaso da sequência correta de montagem do texto-ilegível.

Desde o início de 2012, já podemos contar com um gerador realmente aleatório em chips convencionais da Intel, através da instrução *RdRand*, nos chips da plataforma *ivy-bridge*, segundo Hofemeier (2012), portanto podemos considerar a hipótese do gerador de sequência aleatória como verdadeira e funcional. Neste trabalho não tivemos acesso a um chip com essa instrução (*RdRand*), trabalhamos que essa hipótese seja verdadeira e nas implementações dos experimentos utilizamos um gerador relativamente seguro da linguagem Java, implementado pela classe, *java.security.SecureRandom*[2], esse gerador aleatório atende as especificações estatísticas do NIST (*National Institute of Standards and Technology*) estabelecidas pela *FIPS 140-2, Security Requirements for Cryptographic Modules*[3], na sessão 4.9.1.

[1] Para tal basta utilizar um programa para capturar todos os pacote de comunicação que passam na interface de rede (um *sniffer*, o *Wireshark* (http://www.wireshark.org/) é um do mais conhecidos programas para tal).

[2] A documentação sobre esse gerador está disponível no javadoc, através do endereço: http://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html

[3] Disponível em csrc.nist.gov/cryptval/140-2.htm

Sendo T_{quebra} o tempo total de quebra da chave de sessão k utilizado por um algoritmo criptográfico qualquer $f_k(x) = y$, no melhor caso, o adversário tem sorte, e consegue, ao acaso, escolher a sequência correta i com a qual os blocos foram permutados, utilizando a cifra multicanal. Nesse caso, o tempo total de quebra é exatamente o tempo de quebra do algoritmo criptográfico utilizado, $T_{total} = T_{quebra}$. A chance do adversário conseguir tal façanha é dada pela probabilidade $P(quebra) = \frac{1}{n!}$ no caso da não repetição de canais e $P(quebra) = \frac{1}{n^n}$, caso haja repetição, ou seja, bem baixa, para um número razoável de canais.

No pior caso, o adversário tem azar, e tem que avaliar cada uma das combinações possíveis para a sequência i. Nesse caso, T_{total} pode ser escrito como $T_{total} = T_{quebra}.n!$, sem repetição de canais, ou $T_{total} = T_{quebra}.n^n$, caso haja reutilização dos canais.

Considerando a probabilidade de quebra $P(quebra) = \frac{1}{n!}$ (sem repetição de canais), podemos realizar uma análise de tempo médio, T_{medio} para a quebra da cifra multicanal, da seguinte forma[4]:

$$T_{medio} = \frac{1}{n!}.T_{quebra} + \frac{2}{n!}.T_{quebra} + ... + \frac{n!}{n!}.T_{quebra} \tag{4.1}$$

$$T_{medio} = \frac{1}{n!}.(1 + 2 + ... + n!).T_{quebra} \tag{4.2}$$

$$T_{medio} = \frac{T_{quebra}}{n!}.\sum_{i=1}^{n!} i \tag{4.3}$$

$$T_{medio} = \frac{T_{quebra}}{n!}.(\frac{n!.(n! + 1)}{2}) \tag{4.4}$$

$$T_{medio} = T_{quebra}.(\frac{(n! + 1)}{2}) \tag{4.5}$$

Assintoticamente para um número grande de n, percebemos que o tempo médio T_{medio} é igualmente explosivo, como no pior caso.

$$\lim_{n\to\infty} T_{medio} = \infty \tag{4.6}$$

Portanto, podemos observar que como o crescimento das combinações é no mínimo fatorial, com poucos canais estabelecidos aumenta-se consideravelmente o esforço da criptanálise.

Ainda no caso da obtenção total dos blocos do texto-ilegível pelo adversário (Carlos), se serializarmos os blocos p obtidos na mesma ordem da obtenção, notamos que a operação resultante trata-se, na realidade, de uma permutação aleatória em blocos do texto-ilegível y, definidos pelo emprego de uma segunda chave, a sequência aleatória i. Conforme já mencionado (capítulo 3) trata-se uma composição de funções, onde a função composição c pode ser definida como $c_{k1k2}(x) = g_{k2}(g_{k1}(x))$. A demonstração que $c_{k,i}(x)$, no nosso caso a cifra multicanal, é mais forte (mais seguro), que uma outra cifra qualquer, $f_k(x)$, em termos assintóticos é razoavelmente difícil, no entanto, podemos testar o aumento de segurança para alguns casos de criptanálise mais utilizados, é o caso, em relação a criptanálise diferencial que apresentamos no capítulo 3.

Neste capítulo apresentamos as características da cifra multicanal e suas propriedades. Mostramos que podemos considerar um gerador aleatório real e disponível nas próximas gerações de computadores, através da instrução *RdRand* dos novos *chips* da Intel. Mostramos que as combinações geradas pela cifra multicanal são grandes o suficiente para um aumento considerável de segurança.

[4]Consideramos na demonstração o caso sem repetição de canais, para o caso da repetição de canais, basta trocar o $n!$, por n^n

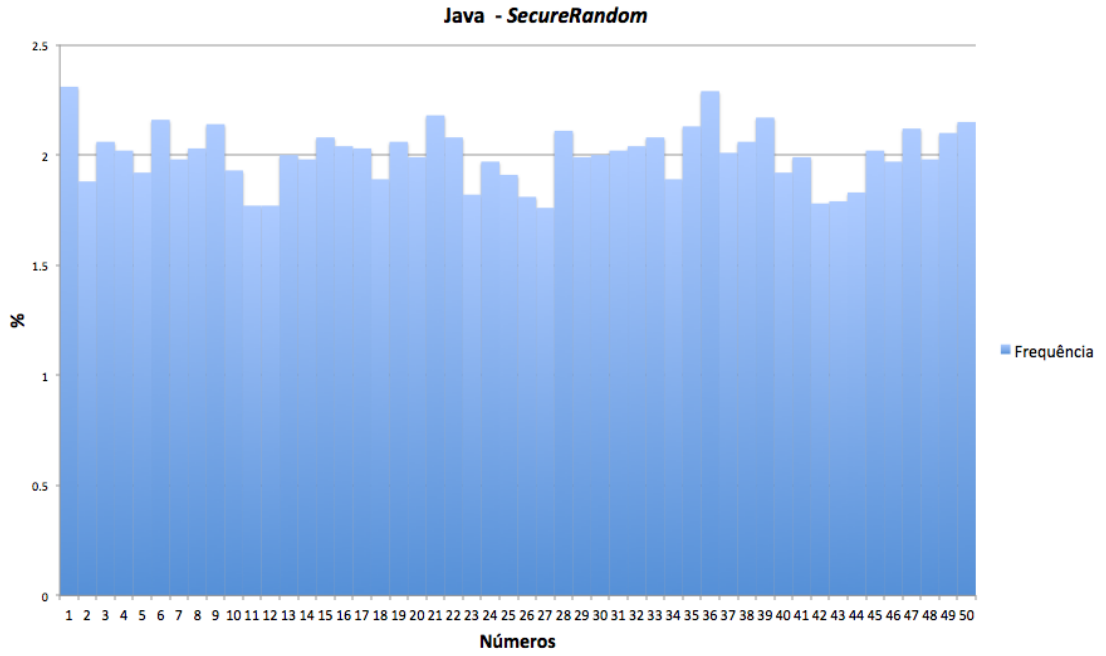

Figura 4.1: *Distribuição de números aleatórios de 1 a 50, pela classe [java.security.SecureRandom] da plataforma Java.*

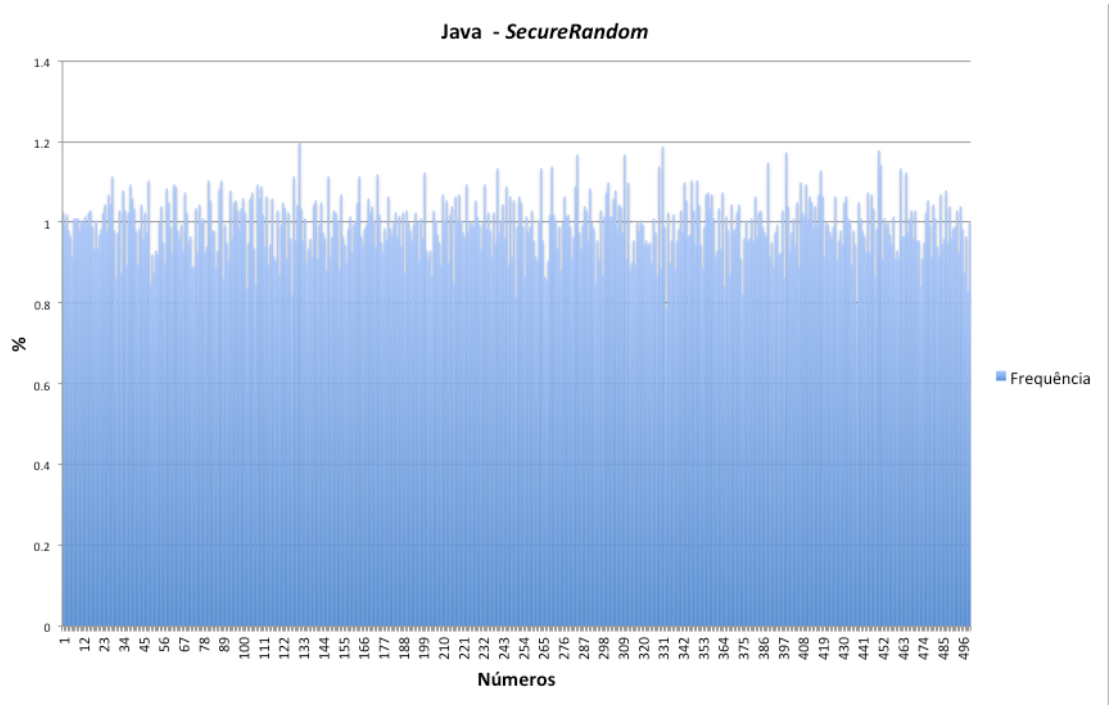

Figura 4.2: *Distribuição de números aleatórios de 1 a 500, pela classe [java.security.SecureRandom] da plataforma Java.*

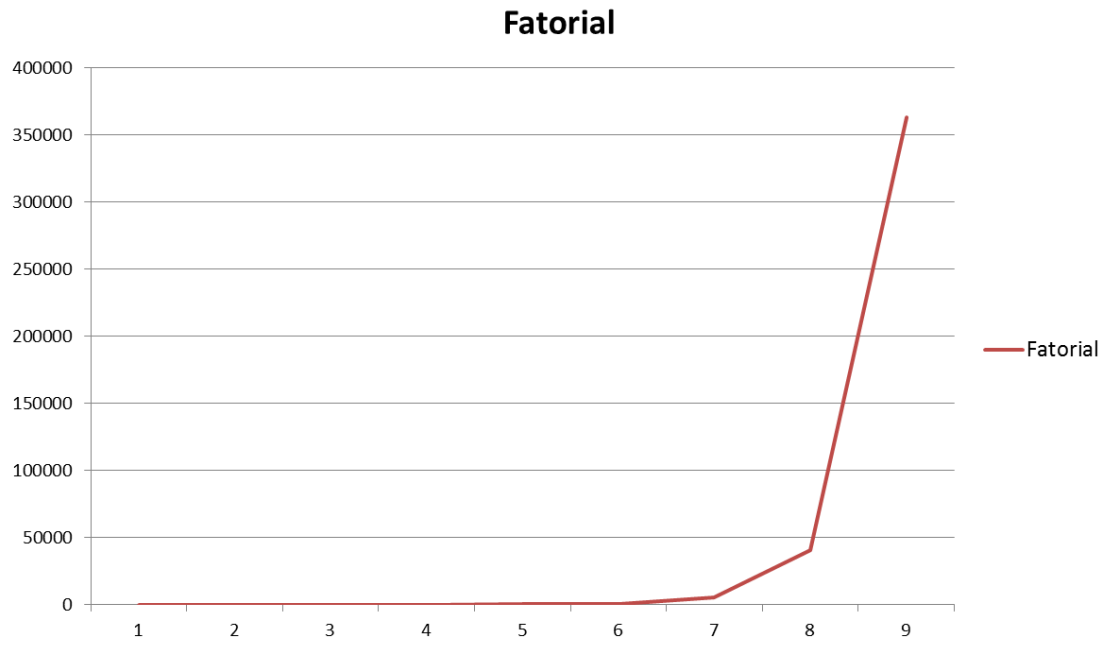

Figura 4.3: *Crescimento de uma função fatorial.*

Capítulo 5

Metodologia

Neste capítulo apresentamos a metodologia utilizada nesse trabalho para aferir a viabilidade da cifra multicanal como uma alternativa de uso prático, e, também, como medir a segurança da cifra em relação à criptanálise diferencial, através do cálculo dos valores Δx e Δy.

Para validar a cifra multicanal proposta, foi desenvolvido um sistema "cliente-servidor", inspirado na técnica FHSS, com agentes comunicando-se entre si através de múltiplos canais TCP/I que simulando computacionalmente uma comunicação real, nos permitiu gerar dados empíricos de análise.

Sobre a transferência de dados da cifra, nessa implementação, temos os seguintes pontos de avaliação:

1. Como múltiplos canais são estabelecidos, o tempo de administração desses canais, inviabilizam a transmissão?
2. Qual é relação entre o tempo de administração e o tempo total da comunicação?
3. Qual é a sua evolução em função do número de canais estabelecidos?
4. Em uma comunicação serial os dados são empacotados pelo protocolo TCP uma única vez, no caso da cifra multicanal, por conta da divisão da informação em vários blocos, qual é o tempo a mais (*overhead*) gasto pelo TCP para a transmissão de cada um desses blocos?
5. Qual a relação do tempo gasto com a cifra multicanal e em relação a apenas um canal seguro estabelecido, o meio normal de comunicação?

Para responder a tais pontos, os tempos da comunicação foram aferidos com precisão de milissegundos e segregados por função: inicialização, estabelecimento dos múltiplos canais de comunicação, divisão da informação, transmissão e tempo de remontagem da informação.

Para a relação do tempo com o número de canais, foram efetuados vários experimentos com diferentes números de canais.

E para o estudo da relação entre a cifra multicanal e uma comunicação segura normal (com apenas um canal estabelecido). Uma nova implementação foi realizada, com apenas um canal seguro (TLS), e os mesmos dados utilizados nos experimentos com a cifra multicanal foram utilizados na transmissão nesse único canal, os tempos foram aferidos e comparados ao tempos obtidos com a transmissão multicanal.

A respeito da segurança da cifra, sabemos que há uma relação fatorial, ou exponencial[1], entre o número de canais e o número combinações com a qual o adversário terá que lidar. Um ponto importante é saber o número máximo de canais que é possível ser estabelecido, evidentemente, essa informação define o máximo da complexidade obtida no sistema. Para obter essa informação,

[1]Por conta da combinações possíveis na permutação, fatorial, sempre que não houver repetição de canal, e exponencial, caso o canal possa se repetir.

elevamos progressivamente o número de canais até o momento que o sistema se torna inviável, ou seja, não realiza mais a transmissão dos dados.

Quanto ao ganho de segurança dessa cifra em relação às demais, é uma questão muito difícil de ser respondida e depende de diversos fatores, neste trabalho consideramos a segurança da cifra em relação a criptanálise diferencial (capítulo 3), para essa avaliação um outro experimento foi criado, uma outra implementação, dessa vez tratando a cifra multicanal apenas como a aplicação de uma segunda chave criptográfica, como um algoritmo aleatório de permutação.

Para aferir os diferenciais, os experimentos foram realizados com um texto-legível x gerado aleatoriamente com um tamanho de 64 *bits* (8 *bytes*). A cifra multicanal (como permutação) foi utilizada com diferentes tamanhos de bloco, $p = \{1, 4, 8, 16, 32\}$.

Para a análise dos deltas de entrada (Δx) e saída (Δy) foi utilizada a medida da Distância de Hamming, Hamming (1950), que mede a quantidade de *bits* alterados entre dois vetores binários[2]. Para gerar o Δx um bloco de 64 *bits* x' é gerado aleatoriamente e duplicado, gerando o bloco x", neste último, são utilizados somente os primeiros 4 *bits*, onde em um laço são trocados os *bits*, varrendo todas as possibilidades possíveis, primeiramente em 1 *bit* de cada vez, depois em grupos de dois *bits*, em grupos de 3 e por fim um grupo de 4 *bits*. O número de combinações possíveis de cada passo do experimento pode ser calculado através do binômio de Newton. Cada passo através da equação:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{5.1}$$

Onde n é o tamanho do conjunto e k é agrupamento utilizado. No caso da geração de x", $n = 4$, e $1 \leq k \leq 4$.

Para o cálculo de combinações totais do experimento, basta seguir:

$$C_{total} = \binom{4}{1} + \binom{4}{2} + ... + \binom{4}{4} \tag{5.2}$$

A figura 5.1 ilustra o processo de geração do x'' em relação a um x'.

A função de troca de *bits* é bem genérica, poderia ser utilizada com qualquer valor n, mas para a criptanálise diferencial, interessa valores baixos de n (no nosso caso igual a 4), porque não se deseja uma mapeamento muito de grande Δx, chaves k e Δy.

Para o Δy medimos a diferença de saída após a permutação (de diferentes tamanhos de blocos) entre o texto-ilegível y' gerado pelo texto-legível original x' (sem a troca de *bits*) e o texto ilegível y" gerado pelo texto legível alterado x".

Os diversos resultados dos Δx e Δy foram tabulados e os dados organizados graficamente para melhor visualização e são apresentados no capítulo 6. Com tais valores, Δx e Δy, é possível afirmar a segurança da cifra em relação à criptanálise diferencial, onde para a consideração de forte e seguro, Δy deve ser relativamente "grande" em relação a Δx.

Neste capítulo propomos duas implementações, a primeira para responder as questões de viabilidade prática da cifra multicanal e a segunda para testar a segurança da cifra em relação à criptanálise diferencial. Detalhamos o processo de geração dos Δx e Δy, variáveis necessárias para a verificação de segurança, mostrando como obter tais valores, a partir da alteração de binária das combinações possíveis dos quatro primeiros *bits* do bloco de entrada.

[2] A medida é bem genérica, pode ser utilizada em outros tipos de vetores, como *strings*, por exemplo. O algoritmo utilizado para o cálculo da Distância de Hamming binário foi o publicado por Wegner (1960), muito elegante por sinal.

Figura 5.1: *Alteração de* bits *seguindo a combinação através dos k agrupamentos. O valores mais claros na ilustração representam os* bits *alterados.*

Capítulo 6

Implementação

Neste capítulo detalhamos as duas implementações nos quais se baseiam os experimentos realizados, apresentamos os resultados e fazemos uma análise do significado dos dados obtidos.

Duas aplicações distintas foram desenvolvidas, conforme descrito no capítulo 5. A primeira que detalhamos é um experimento de transferência de dados e a segunda um experimento para a verificação da segurança em relação a criptanálise diferencial.

Todos os programas foram desenvolvidos na plataforma Java (J2SE - *Java 2 Standard Edition*), versão JDK 1.7. Duas máquinas foram utilizadas no desenvolvimento e execução. A primeira máquina, a ponta cliente da comunicação, o Beto, um *Macbook Pro*, com CPU *Intel* Core 2 Duo, de 2.53 GHz, e 8 GBytes de memória RAM, rodando o MacOSX, versão 10.8.4, com HD de 250 GBytes. E a segunda, a ponta servidora da comunicação, a Alice, um *desktop*, com CPU Intel Core i7, de 2.8Ghz, e 8 Gbytes de memória, rodando o *Windows 7*, com HD de 128 GBytes SSD (*Solid-State Drive*). Ambas máquinas de 64 *bits*, com interface de rede *giga bit*. A comunicação entre as máquina foi realizada por cabo através de um roteador *TP-Link*, modelo TL-WR841ND.

O código-fonte está disponível no seguinte endereço do *github*[1] : https://github.com/sylvioazevedo/multichannel-security, é um repositório privado, para ter acesso, envie um e-mail para: sylvio@ime.usp.br.

6.1 Aplicação de transferência de arquivos

A cifra multicanal foi testada em uma aplicação simples de transferência de arquivos no modelo cliente-servidor clássico. Seguimos exatamente a descrição da cifra apresentada no capítulo 3.

Para detalharmos a aplicação desenvolvida, trataremos o agente da comunicação Beto como o cliente, e a agente Alice, como servidor.

O cliente inicia a comunicação, estabelece primeiramente as conexões no canal seguro (TLS) e nos múltiplos canais de comunicação, após, com o servidor troca a chave de sessão e a sequência aleatória da transmissão de dados.

O cliente informa o arquivo que quer receber através do canal TLS, o servidor localiza o arquivo em seu diretório, o abre, lê e criptografa os dados do arquivo utilizando a chave de sessão, divide os dados em pacotes de transmissão, usando o tamanho do arquivo pelo número de canais estabelecidos (em *bytes*), e então, envia os pacotes através dos canais, utilizando-os sequencialmente e selecionando os pacotes com a sequência aleatória. Após a transmissão de todos os blocos do arquivo o servidor envia uma sinalização de término de arquivo (EOF - *End of File*).

O cliente vai recebendo através dos canais cada um dos blocos de dados do arquivo requerido, na ordem em que estão sendo enviados, após o recebimento de todos os blocos e a sinalização de término de arquivo (EOF), ordena os dados utilizando a sequência aleatória. Com os dados ordenados, decriptografa os dados utilizando a chave de sessão, cria um arquivo e escreve os dados decriptografados nele.

[1] Sistema web de controle de projetos versionados pelo *git*, uma ferramenta moderna de controle de versão - (http://git-scm.com).

Os tamanhos de arquivos utilizados no experimento foram: 100 *bytes*, 44 *Kbytes*, 100 *Kbytes* e 1,7 *MBytes*. Os dados dos arquivos são mantidos em memória nesta implementação, por isso arquivos maiores não foram testados, entretanto nada impede que uma outra aplicação seja criada para trabalhar com arquivos gigantescos.

Para realizar as conexões TCP/IP foi utilizado o *framework* de infraestrutura de rede Apache Mina[2], versão 2.0.7. O projeto Apache Mina facilita muito o desenvolvimento de aplicações de rede, restando para o desenvolvedor, somente a definição do esquema de serialização[3] e o protocolo da comunicação a ser utilizado. O método de serialização escolhido foi o BSON [4](*Binary JSON*), um método de serialização binário baseado no JSON [5] (*Javascript Object Notation*), uma representação de objetos *javascript* em um mapa de texto (chave-valor). O método de serialização é desenvolvido através da criação de *codecs* (acrônimo de "codificador" e "decodificador") implementando interfaces do Mina (à saber, *ProtocolEncoder* e *ProtocolDecoder*) e o protocolo estendendo uma classe padrão (*IoHandlerAdapater*), duas dessas classes foram desenvolvidas, uma para o protocolo do canal seguro (TLS) e outra para o protocolo os múltiplos canais de comunicação. No detalhamento da cifra multicanal (capítulo 3) os dois foram apresentados juntos.

6.1.1 Resultados da aplicação de transferência de arquivos

Apresentamos nessa sessão os resultados da aplicação de transferência de arquivos. Para esse experimento foram utilizados tamanhos de arquivos típicos[6] de 100 *Bytes*, representando uma linha texto; 40 *kBytes*, uma página html normal; 500 *kBytes*, um arquivo de texto *Microsoft Word* com 5 páginas ou um arquivo PDF (*Portable Document Format*); 1,1 *MBytes*, uma foto em alta definição ou 1 minuto de música em MP3 (*MPEG-1/2, Audio Layer 3*); e 6,6 *MBytes*, 3 minutos de MP3 em um taxa de bits muito altas (256 *kbps*) ou um 1 minuto de fluxo de video.

Foram utilizados diferentes números de canais conectados simultaneamente, à saber, $n_{canais} = \{1, 3, 6, 12, 25, 50, 100, 200, 500\}$ (repare que o número de canais é dobrado em cada experimento, propositalmente quadrático, para verificar se há explosão de tempo em função a um número expressivo de canais simultâneos estabelecidos). Na figura 6.1, temos os tempos de inicialização, o tempo necessário para levantar a infraestrutura da comunicação com o estabelecimento dos n_{canais}, entre o cliente e o servidor, apesar de estarem no gráfico relacionado com os diferentes arquivos, o estabelecimento dos canais de comunicação é um processo independente, e por isso as curvas apresentam um mesmo comportamento. O tempo necessário para inicializar os canais teve um comportamento explosivo, principalmente depois de 50 canais, onde o tempo passa a praticamente dobrar. O tempo de inicialização é fator determinante para o número de canais que se deseja utilizar na cifra multicanal, no entanto, dependendo da aplicação, esse tempo pode ser gasto apenas uma vez se as conexões forem mantidas não apenas uma transferência de dados, mas várias.

A seguir, apresentaremos os experimento com o tempo da comunicação para a transferência de dados dos múltiplos arquivos, começando pelo menor de 100 *Bytes* até o maior de 6,6 *MBytes*.

Nas tabelas apresentamos o número de canais utilizados (No. canais), o tempo de estabelecimento das conexões dos canais (Inicialização), o tempo de transferência dos dados pelos múltiplos canais (Transferência), o tempo da administração dos canais e manipulação dos arquivos (inclui, a abertura, leitura e escrita) que é o tempo de *overhead* [7] (*Overhead*), e o tempo total da comunicação (Total).

Como apresentamos em um gráfico só o tempos de inicialização dos canais, nos gráficos dos tempos da comunicação dos arquivos subtraímos essa informação, nestes, portanto, estão as informações

[2]Para maiores informações a respeito da tecnologia, acesse: http://mina.apache.org

[3]Termo utilizado para a definição de como os dados de um determinado objeto (no modelo de programação orientado a objetos) será transformado em um sequência binária para a transmissão em um canal de comunicação. Já a deserialização é a operação inversa, que transforma uma sequência binária novamente em um objeto.

[4]Para saber mais sobre o BSON, acesse: http://bsonspec.org/

[5]Para saber mais sobre o JSON, acesse: http://www.json.org/

[6]Na internet existem algumas referências de quais são os arquivos típicos e mais utilizados, para referência, utilizamos, http://www.gn.apc.org/support/understanding-file-sizes

[7]Esse tempo foi calculado subtraindo do tempo total o tempo de transferência de dados

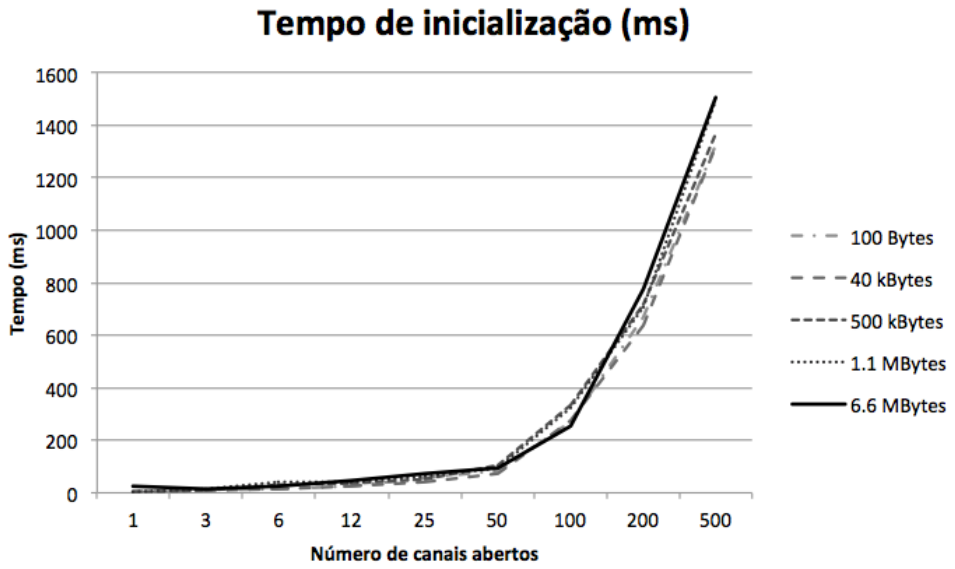

Figura 6.1: *Tempos de inicialização para diferentes números de canais.*

dos canais (nas abscissas) e os tempo de transferência, *overhead* e total da comunicação.

Na tabela 6.1 temos os tempos obtidos para a transmissão de dados de um arquivo de 100 *Bytes*:

No. canais	Inicialização (ms)	Transferência (ms)	*Overhead* (ms)	Total (ms)
1	3	32	13	45
3	8	34	9	43
6	16	33	12	45
12	27	30	22	52
25	56	33	40	73
50	85	25	31	56
100	269	18	41	59
200	671	21	35	56
500	1325	35	45	80

Tabela 6.1: *Tempos de transmissão de dados de um arquivo de 100* Bytes *(ms).*

Na figura 6.2 temos a representação gráfica para os dados da tabela 6.1.

Na tabela 6.2 temos os tempos obtidos para a transmissão de dados de um arquivo de 40 *kBytes*:

No. canais	Inicialização (ms)	Transferência (ms)	*Overhead* (ms)	Total (ms)
1	2	483	7212	7695
3	10	704	871	1575
6	13	813	650	1463
12	25	817	788	1605
25	39	857	777	1634
50	75	790	876	1666
100	278	662	1118	1780
200	640	692	1585	2277
500	1319	790	1302	2092

Tabela 6.2: *Tempos de transmissão de dados de um arquivo de 40* kBytes *(ms).*

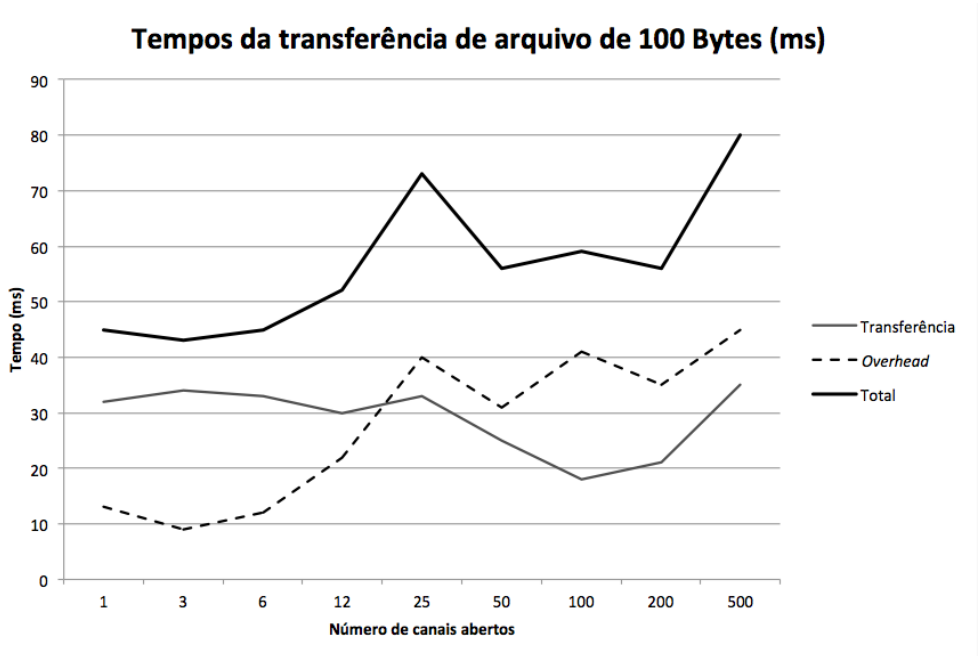

Figura 6.2: *Tempos de transferência,* overhead *e total da comunicação para um arquivo de 100* Bytes.

Na figura 6.3 temos a representação gráfica para os dados da tabela 6.2.

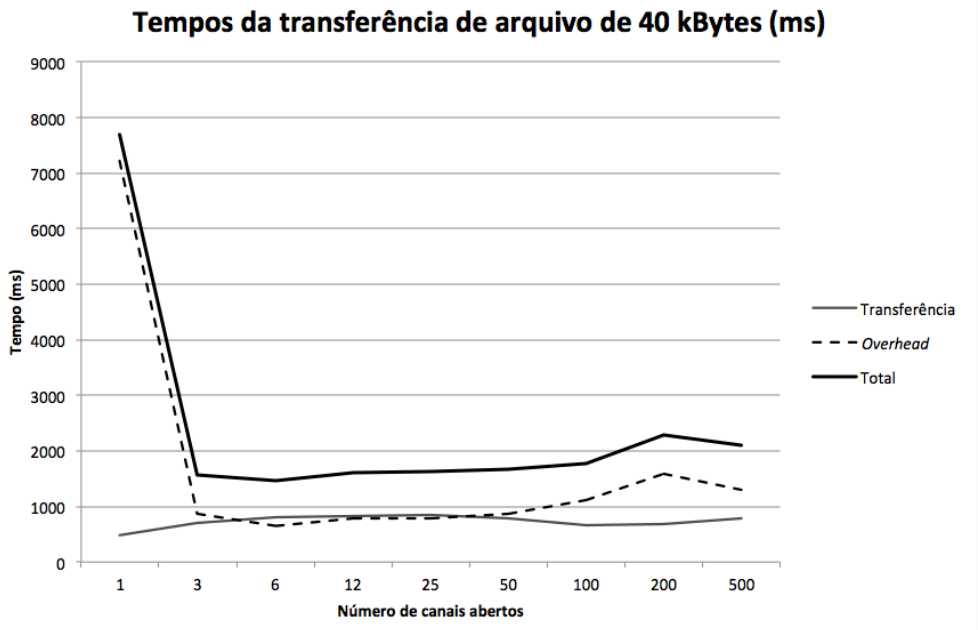

Figura 6.3: *Tempos de transferência,* overhead *e total da comunicação para um arquivo de 40* kBytes.

Na tabela 6.3 temos os tempos obtidos para a transmissão de dados de um arquivo de 500 *kBytes*:

Na figura 6.4 temos a representação gráfica para os dados da tabela 6.3.

Na tabela 6.4 temos os tempos obtidos para a transmissão de dados de um arquivo de 1,1 *MBytes*:

Na figura 6.5 temos a representação gráfica para os dados da tabela 6.4.

Na tabela 6.5 temos os tempos obtidos para a transmissão de dados de um arquivo de 6,6 *MBytes*:

Na figura 6.6 temos a representação gráfica para os dados da tabela 6.5.

No. canais	Inicialização (ms)	Transferência (ms)	*Overhead* (ms)	Total (ms)
1	4	48927	75096	124023
3	11	6309	15892	22201
6	28	7655	12769	20424
12	36	7377	10992	18369
25	54	6771	13200	19971
50	106	8003	13865	21868
100	332	8132	19815	27947
200	718	7744	24779	32523
500	1369	11683	29064	40747

Tabela 6.3: *Tempos de transmissão de dados de um arquivo de 500* kBytes *(ms).*

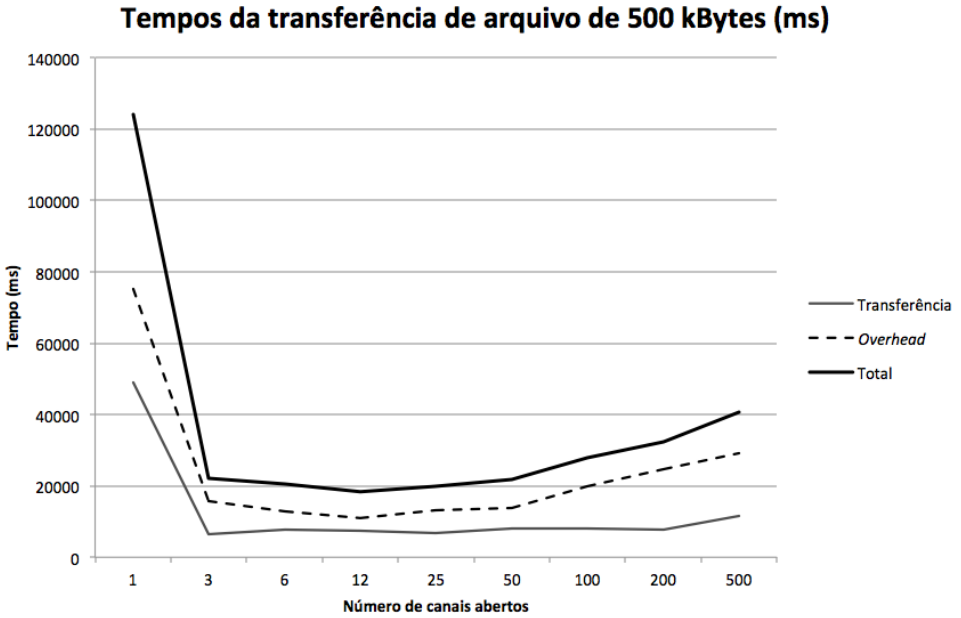

Figura 6.4: *Tempos de transferência,* overhead *e total da comunicação para um arquivo de 500* kBytes*.*

No. canais	Inicialização (ms)	Transferência (ms)	*Overhead* (ms)	Total (ms)
1	2	161212	63938	225150
3	16	18753	19394	38147
6	41	11233	21906	33139
12	42	12188	25279	37467
25	65	9522	28453	37975
50	92	12607	28610	41217
100	326	14733	40804	55537
200	708	15062	45152	60214
500	1499	23826	52763	76589

Tabela 6.4: *Tempos de transmissão de dados de um arquivo de 1,1* MBytes *(ms).*

Os tempos apresentados, foram obtidos da média de 100 execuções com cada um dos arquivos, com cada número de canais. Podemos reparar nesse experimento, que uma característica importante nesses gráficos se sobressai, não há um crescimento explosivo do tempo em função do número de

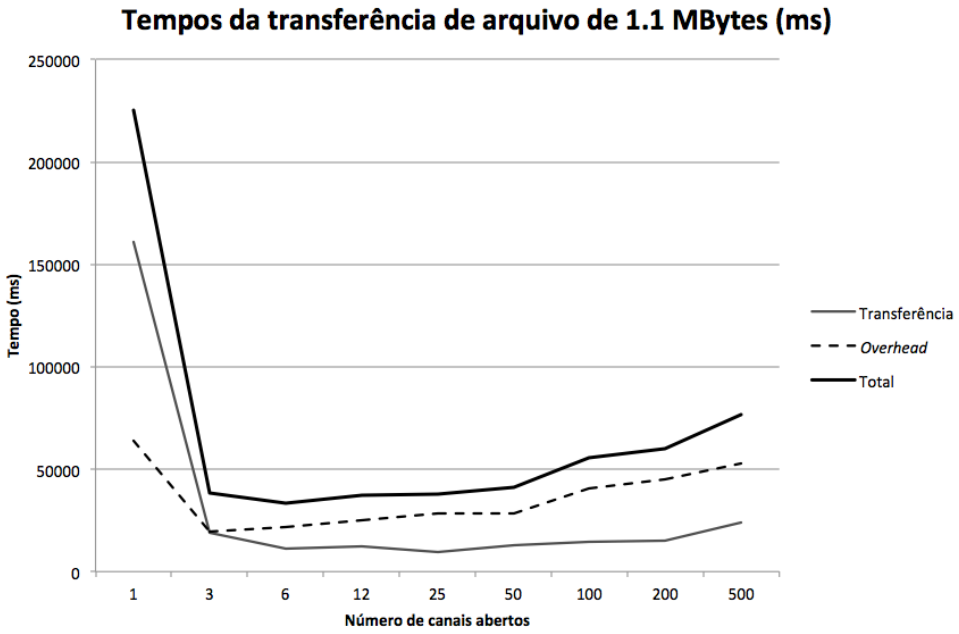

Figura 6.5: *Tempos de transferência,* overhead *e total da comunicação para um arquivo de 1.1* MBytes.

No. canais	Inicialização (ms)	Transferência (ms)	*Overhead* (ms)	Total (ms)
1	27	1526952	56639	1583591
3	13	295801	47389	343190
6	25	225736	44290	270026
12	47	176473	81579	258052
25	75	144215	100876	245091
50	93	134894	127277	262171
100	252	145801	6595	152396
200	775	156481	249484	405965
500	1507	197712	323110	520822

Tabela 6.5: *Tempos de transmissão de dados de um arquivo de 6,6* MBytes *(ms).*

canais antes dos 200 canais, e essa é a informação que mais nos interessa. Quanto maior o número de canais, maior é a segurança, pois com um valor relativamente grande de canais, sabendo que a sequência de transmissão dos pacotes é aleatória (a chave da cifra multicanal) e seu efeito final é uma permutação, podemos considerar que temos uma alta entropia na chave, o que garante a segurança contra o ataque do homem-do-meio, conforme a sessão 2.3.2, em Terada (2008). No contexto TCP/IP essa propriedade é desejável. Os gráficos mostram que na comunicação com as máquina envolvidas, 100 canais poderiam ser utilizados sem problemas.

O expressivo aumento de tempo a partir de 200 canais os altos tempos pode ser explicado pela necessidade de maior tempo para administrar os canais, portanto, um possível *trade off* que deve ser avaliado pelo projetista do sistema.

Outras implementações podem ser realizadas para aproveitar melhor os limites da máquina, mas não é o foco neste trabalho. Aqui é suficiente saber que podemos ter um número razoável de canais para aumentar a segurança da comunicação.

Os tempos para a transmissão em geral são significativamente maiores do que uma transmissão em série normal em um canal TLS seguro, algo em torno de 10 vezes mais lento. Em uma implementação de referência de transferência de arquivos temos os tempos relacionados na tabela 6.6. Não chega a ser impeditivo, mas fica claro que o estabelecimento e administração dos canais tem

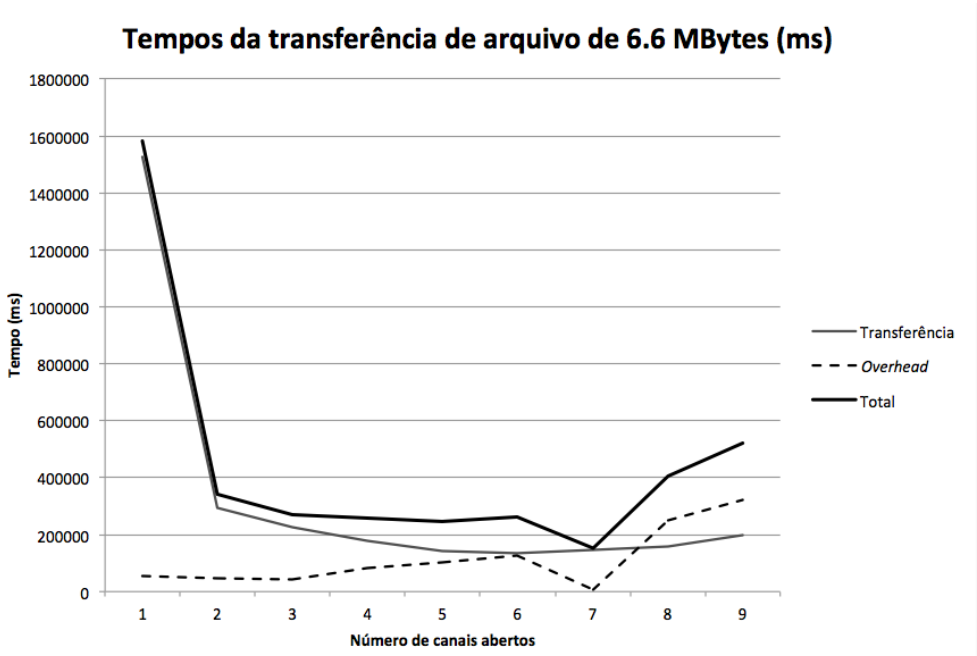

Figura 6.6: *Tempos de transferência,* overhead *e total da comunicação para um arquivo de 6.6* MBytes.

um custo que deve ser considerado.

Tamanho do arquivo	Transferência de dados (ms)
100 Bytes	12
40 kBytes	73
500 kBytes	179
1.1 MBytes	186
6.6 MBytes	1201

Tabela 6.6: *Tempos de transmissão de dados em um abordagem serializada normal (ms).*

A cifra multicanal não estabelece número máximo de canais que possam ser utilizados, entretanto além do *overhead* da administração dos canais, nos experimentos, ao ir aumentando o número de canais, encontramos uma outra limitação, dessa vez, em relação ao sistema de arquivos dos sistemas operacionais. Nos sistemas Unix uma conexão de rede é mapeada em um arquivo temporário no sistema de controle de arquivos e há um limite máximo de arquivos abertos concorrentemente. No *MacOSX* esse limite é de 710, nas distribuições Linux normalmente está configurado em 1024 e encontramos que o sistema operacional Windows trabalha com no máximo em torno de 8.000 conexões simultâneas, embora esse último não tenha relação direta com o sistema de arquivos. Apesar dessa restrição, nos sistemas Unix, é possível configurar o número máximo de arquivos aberto simultaneamente, normalmente através do comando "ulimit". Para a construção de servidores esse ponto merece especial atenção e a consulta do manual da distribuição do sistema operacional é necessária para a correta configuração desse limite.

6.2 Segurança em relação à criptanálise diferencial

A cifra multicanal foi transformada em uma função de permutação aleatória, dessa forma removemos totalmente a características de comunicação e damos a atenção somente as propriedades da cifra, que nesse caso, tem somente o efeito da permutação.

O resultado do cálculo da distância de Hamming é justamente o valor da diferença binária, logo, se calculado para duas entradas da cifra de permutação, é o Δx, e se calculado para as respectivas saídas, é o Δy, necessários para mensurar a força contra a análise diferencial, capítulo 3.

Os experimentos utilizando a distância de Hamming foram realizados para diversos tamanhos de blocos para a permutação, $Tam_{bloco} = \{1, 4, 8, 16, 32, 64\}$, nesse conjunto, o primeiro elemento representando uma permutação binária e o último, o próprio tamanho do *buffer* aleatório, representando nenhuma permutação, esse último foi só mantido como referência.

A operação do experimento, possui as seguintes etapas:

1. Geração de um *buffer* de 8 Bytes (64 *bits*) com valores aleatórios. Para tal, foi utilizada a classe java *Random*, uma vez que retorna inteiros, foi limitada para gerar valores de 0 a 255, os *bits* não utilizados, já que um inteiro tem 4 bytes e só utilizamos 1, são descartadas com uma operação de *casting* [8]. Esse *buffer* gerado representa o x';

2. Geração de uma sequência aleatória em função do tamanho de bloco da permutação. Por exemplo, se o tamanho do bloco for 1 (binário), a sequência aleatória terá 64 valores distintos. Se o tamanho for 4, terá 16 valores, e assim por diante;

3. Um *looping* é estabelecido para varrer todas as combinações dos 4 primeiros *bits* do *buffer* x', conforme explicado no capítulo 5;

4. Clonagem do *buffer* aleatório gerado e alteração dos quatro primeiros *bits* seguindo a combinação corrente (muda a cada passo do laço). O *buffer* clonado com os *bits* alterados representa o x'';

5. Permutação do *buffer* de dados x' com a sequência aleatória, obtendo-se assim o y';

6. Permutação do *buffer* de dados clonado e com *bits* alterados x'' com a sequência aleatória, obtendo-se assim o y'';

7. Realização do cálculo da distância de Hamming para x' e x'', obtendo-se assim o Δx;

8. Realização do cálculo da distância de Hamming para y' e y'', obtendo-se assim o Δy;

9. Contabilização dos Δx e Δy, conforme o agrupamento da combinação corrente.

Ao realizar esse experimento nas primeiras vezes, encontramos um resultado não previsto inicialmente. A operação de permutação não provoca um número maior de *bits* trocados na saída em relação a entrada, ou seja, não protege estatisticamente o texto elegível, pois um *bit* alterado na entrada corresponde diretamente em função da chave um *bit* na saída, em outras palavras, em qualquer combinação das entradas x' e x'', se a chave k for a mesma, o Δx será sempre igual ao Δy. E essa característica é válida inclusive para qualquer tamanho de bloco utilizado na permutação.

Na figura 6.7 representamos a operação de permutação em duas entradas x' e x'', repare que embora y' e y'' sejam diferentes, o número de *bits* alterados continua o mesmo.

Essa característica da permutação a invalida totalmente em relação a criptanálise diferencial pois avaliando as diferenças (operação *xor*) das entradas e saídas é possível inferir diretamente a chave. Verificando novamente a figura 6.7 onde alterou-se o primeiro *bit* da entrada x' em x'', olhando a diferença, é possível ver claramente que essa mudança foi permutada para o último bloco em y'', logo o valor da sequência aleatória correspondente aquele bloco é 1.

[8] Herança da linguagem, é a determinação de tipo de uma determinada instância de valor ou objeto.

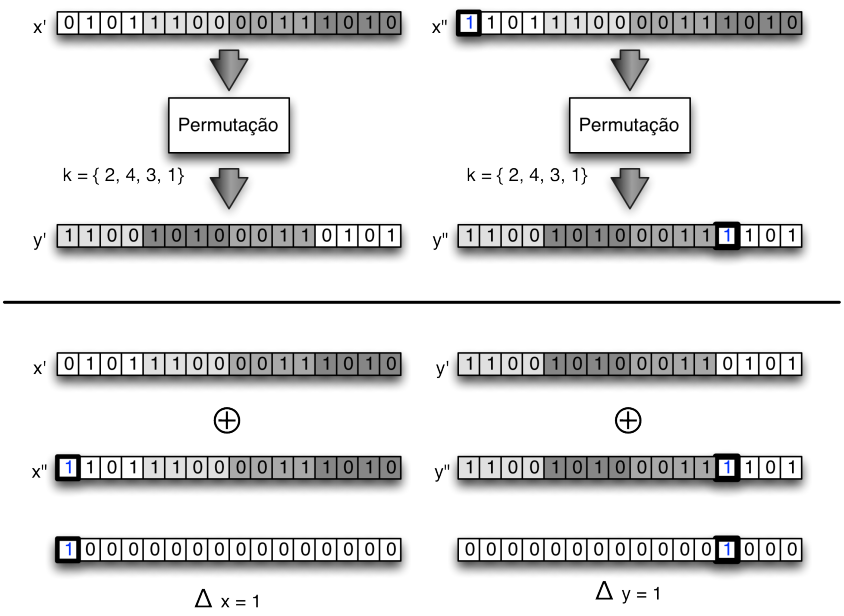

Figura 6.7: *Exemplo mostrando que a operação de permutação não causa difusão binária.*

Algumas maneiras foram pensadas para fortalecer a cifra multicanal em relação à criptanálise diferencial, uma delas era colocar um componente aleatório de *offset*, provocando um deslocamento dos *bits*, isso iria descaracterizar a cifra com um aumento de uma outra chave. Preferiu-se então, utilizar o mesmo mecanismo do AES para realizar o aumento de difusão dos *bits* de saída, que naturalmente realiza o trabalho de maneira segura, testada ao longo de vários anos, sem alterar o tamanho da chave.

A função principal do AES responsável pela difusão binária é a chamada de *MixColumns*, onde a entrada é transformada em uma matriz de *bytes* e multiplicada com uma outra matriz com valores dentro do $GF(2^8)$ (*Galois Field* - Corpo de Galois). E é ao misturar as linhas e colunas (na multiplicação matricial) que encontramos o efeito de difusão desejado, onde diversos *bits* são alterados em função de uma alteração na entrada.

O número de *bits* alterados em função da alteração de cada *bit* da entrada é no caso desejável a metade do tamanho do bloco da cifra, para a máxima entropia, onde a probabilidade de qualquer valor (0 ou 1) em qualquer *bit* do bloco é de 50%. No nosso caso o tamanho do bloco é 64, e o valor desejável é 32, portanto.

Experimentalmente em conjunto com a função *S-Box* (*Substitution Box* - Caixa de Substituição) do AES, onde valores são trocados por outros, também dentro do $GF(2^8)$; e com a função de deslocamento (*ShiftBytes* - Deslocamento de *bytes*), os números de *bits* alterados na saída a ficaram muito próximo do desejável.

A função *SubBytes*, consiste em dois passos, o primeiro, a inversão de cada um dos bytes por uma função $f(x)$. Sejam $t, u \in GF(2^8)$, $f : u \to t = u^{-1} \bmod m(x)$ Terada (2008). E a segunda etapa, uma multiplicação matricial da seguinte forma, sendo b os *bits* de entrada e y *bits* de saída:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

De igual maneira a função inversa da função *SubBytes*, possui duas etapas, a aplicação da inversa dentro do $GF(2^8)$. Sendo $t, u \in GF(2^8)$, temos $f^{-1} : t \to u = t^{-1} \bmod m(x)$.

E a segunda a etapa, uma multiplicação matricial da seguinte forma:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Na operação *ShiftRows*, o bloco a ser codificado é representado em uma matriz M de 4 linhas e N_b colunas. A operação ocorre da seguinte maneira:

- A linha 0 da matriz M não é alterada;
- A linha 1 da matriz M é deslocada circularmente para a esquerda em C_1 *bytes*;
- A linha 2 da matriz M é deslocada circularmente para a esquerda em C_2 *bytes*;
- A linha 3 da matriz M é deslocada circularmente para a esquerda em C_3 *bytes*;

Os valores constantes de C_1, C_2 e C_3 dependem do valor N_b, conforme a tabela 6.7.

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	2	3

Tabela 6.7: *Tabela de constantes de deslocamento em função do tamanho do bloco de entrada do AES.*

A operação inversa da função *ShiftRows* é realizada no deslocamento ao contrário, do seguinte modo:

- A linha 0 da matriz M não é alterada;
- A linha 1 da matriz M é deslocada circularmente para a esquerda em $N_b - C_1$ *bytes*;
- A linha 2 da matriz M é deslocada circularmente para a esquerda em $N_b - C_2$ *bytes*;
- A linha 3 da matriz M é deslocada circularmente para a esquerda em $N_b - C_3$ *bytes*;

A seguir, detalhamos a função *MixColumns*, uma multiplicação matricial da seguinte forma:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Detalhe da operação inversa da função *MixColumns*:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

A cifra multicanal (uma permutação) associada ao mecanismo de difusão do AES, foi batizada de SMP (*S-Box*/*ShiftBytes*, a *MixColumns*, acrônimos das funções que a compõem e a permutação aleatória).

As figuras 6.8 e 6.9, representam as operações da cifra SMP. Repare que as operações importadas do AES não sofrem interferência da chave (sequência aleatória i), portanto, juntas formam apenas uma função de codificação.

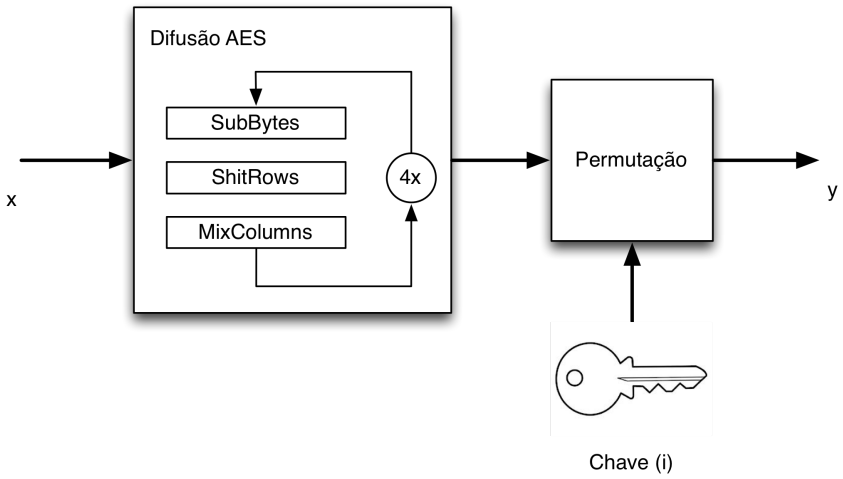

Figura 6.8: *Representação da cifra SMP para a operação criptográfica.*

6.2.1 Resultados dos experimentos em relação à criptanálise diferencial

Os dados contabilizados por Δx e Δy são apresentados a seguir para a SMP, estão representados graficamente e organizados por valores diferentes do tamanho de bloco de permutação, começando com a permutação binária (*bit* a *bit*), bloco de 4 *bits*, bloco de 8 *bits*, bloco de 16 *bits*, bloco de 32 *bits* e, por último, o bloco de 64 que coincide com o tamanho do bloco de entrada, nesse caso não há permutação.

Quatro gráficos são apresentados para cada tamanho de bloco, o primeiro, o total de *bits* alterados na saída (Δy) em função do número de combinações de um determinado agrupamento. O segundo, uma estatística do número médio de *bits* alterados, o desvio padrão e os valores de máximos e mínimos encontrados. E o último, o coeficiente de variação, dado pelo desvio padrão dividido pela média.

Os resultados obtidos do SMP não apresentam grande variação em qualquer tamanho de bloco, isso mostra claramente que a cifra de permutação não colabora com a difusão. Isso pode ser confir-

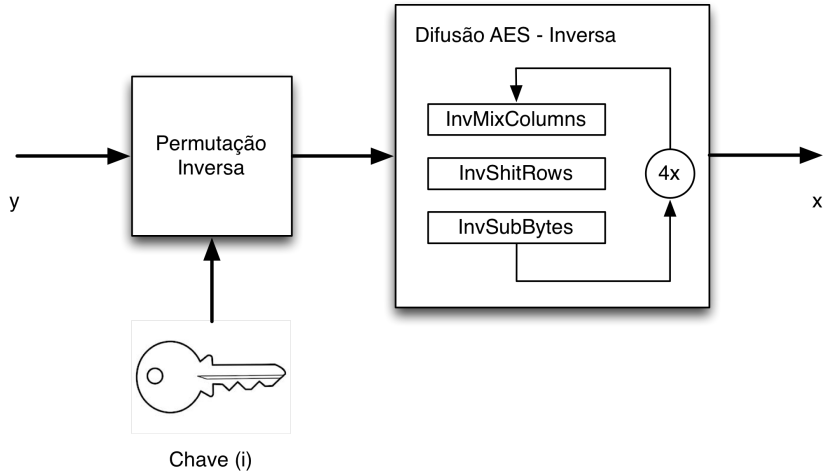

Figura 6.9: *Representação da operação inversa da cifra SMP.*

mado no último gráfico, o de controle, com o bloco de 64 *bits*, onde não há permutação, há ainda grande difusão dos *bits* na saída.

Total de *bits* alterados pela operação SMP

A seguir mostramos os gráficos de números de *bits* alterados pela cifra SMP tanto na entrada como na saída.

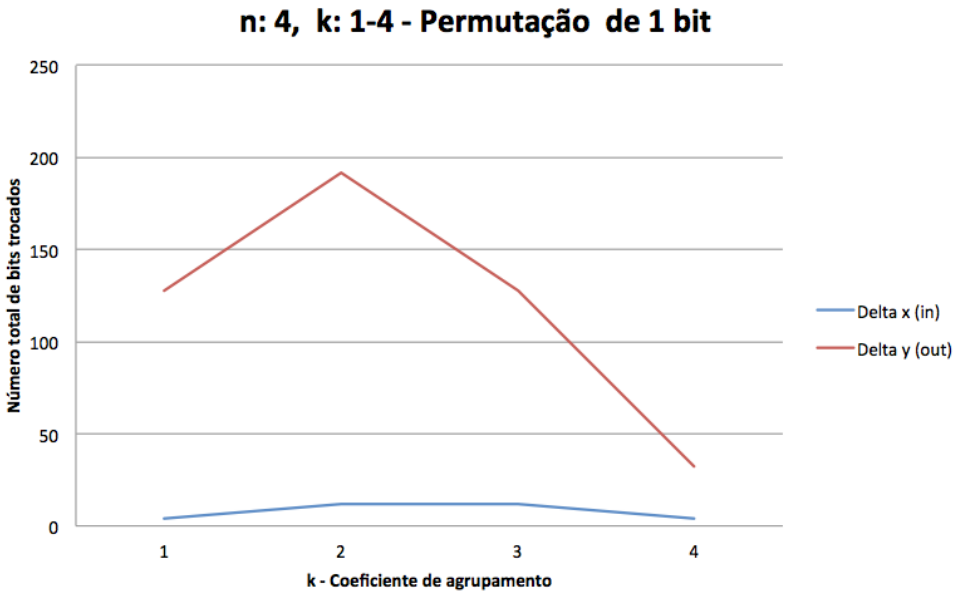

Figura 6.10: *Total de* bits *alterados para a permutação binária.*

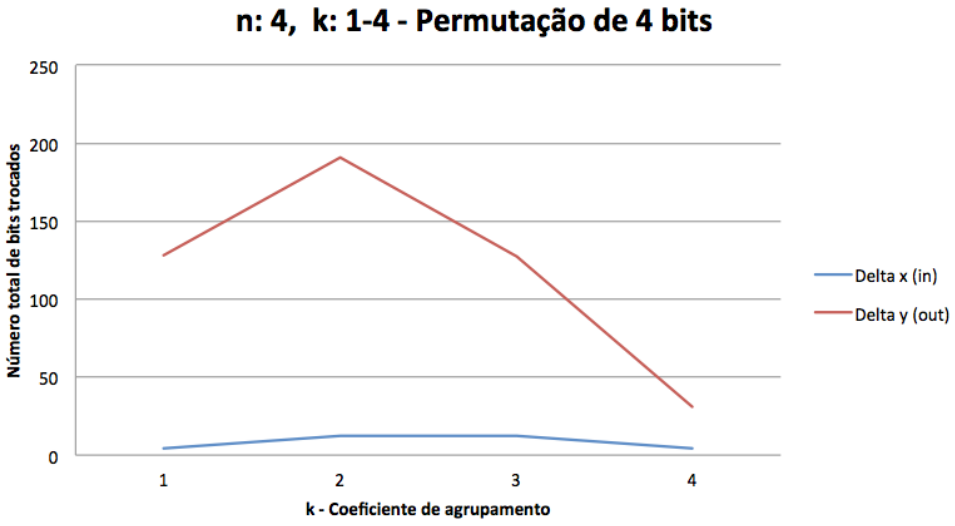

Figura 6.11: *Total de* bits *alterados para a permutação de blocos de 4* bits.

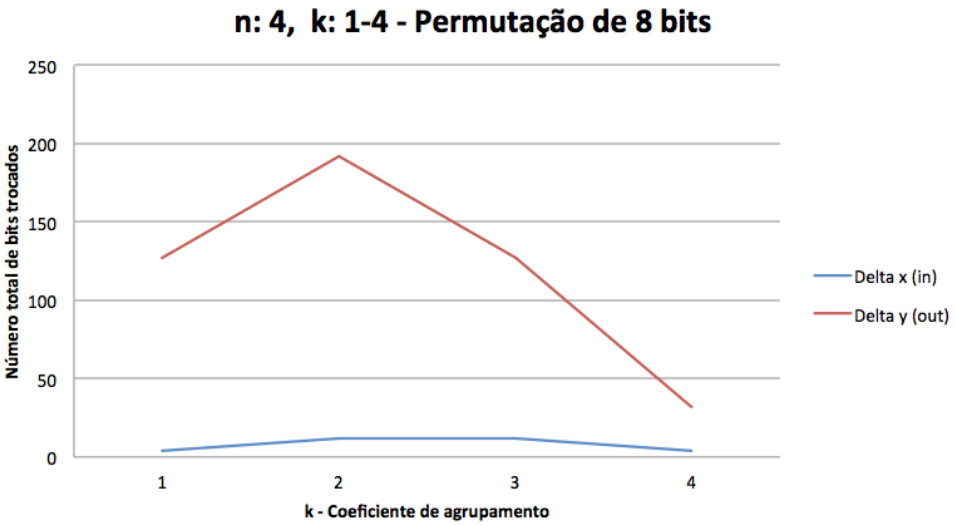

Figura 6.12: *Total de* bits *alterados para a permutação de blocos de 8* bits.

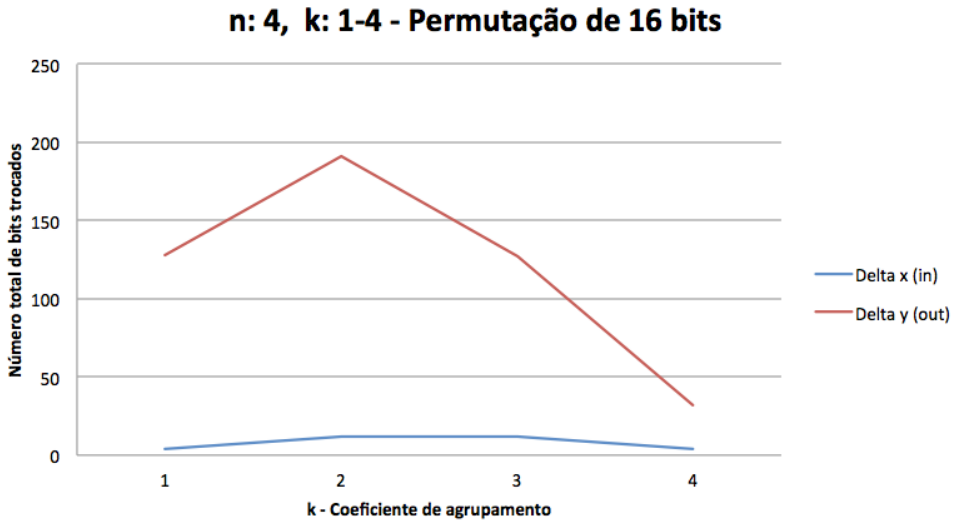

Figura 6.13: *Total de* bits *alterados para a permutação de blocos de 16* bits.

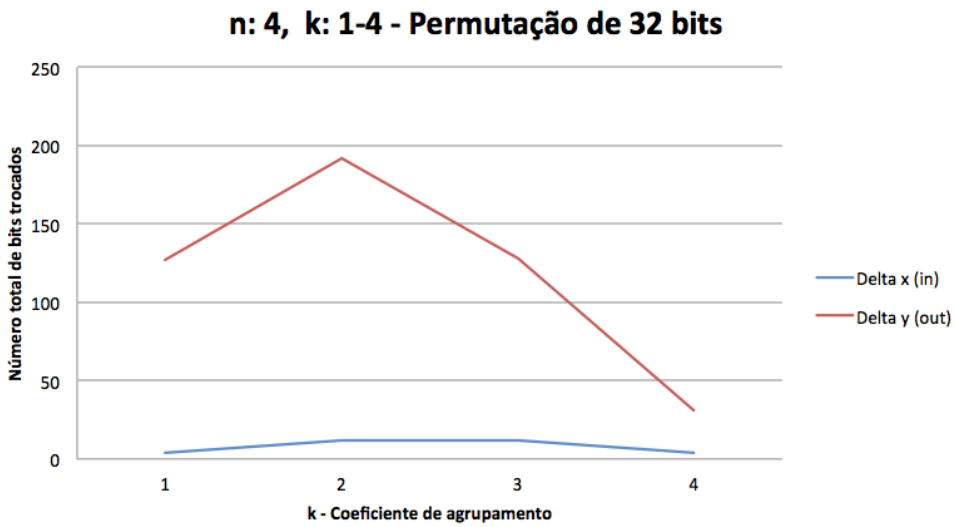

Figura 6.14: *Total de* bits *alterados para a permutação de blocos de 32* bits.

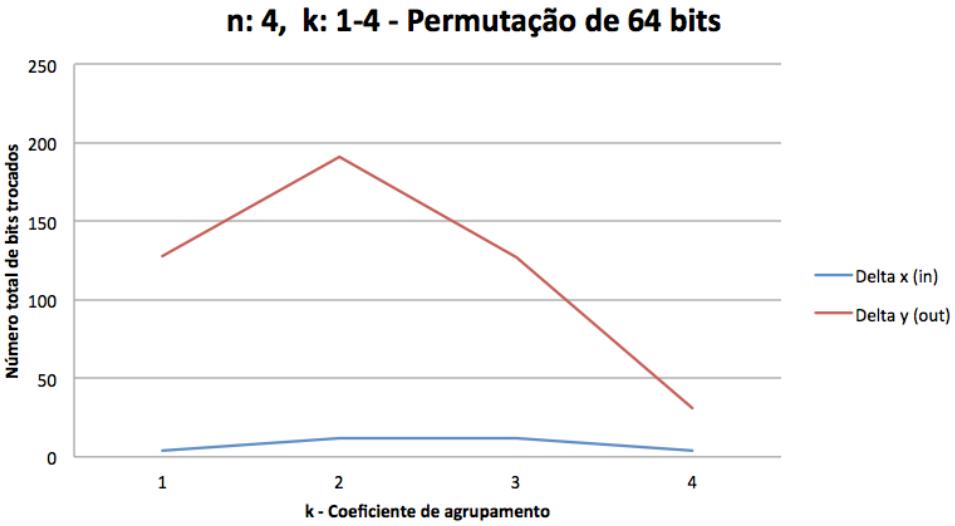

Figura 6.15: *Total de* bits *alterados para a permutação de blocos de 64* bits.

Estatística dos *bits* alterados na operação SMP

Apresentamos aqui as médias, mínimos, máximos e os desvios padrão dos *bits* alterados pela cifra SMP.

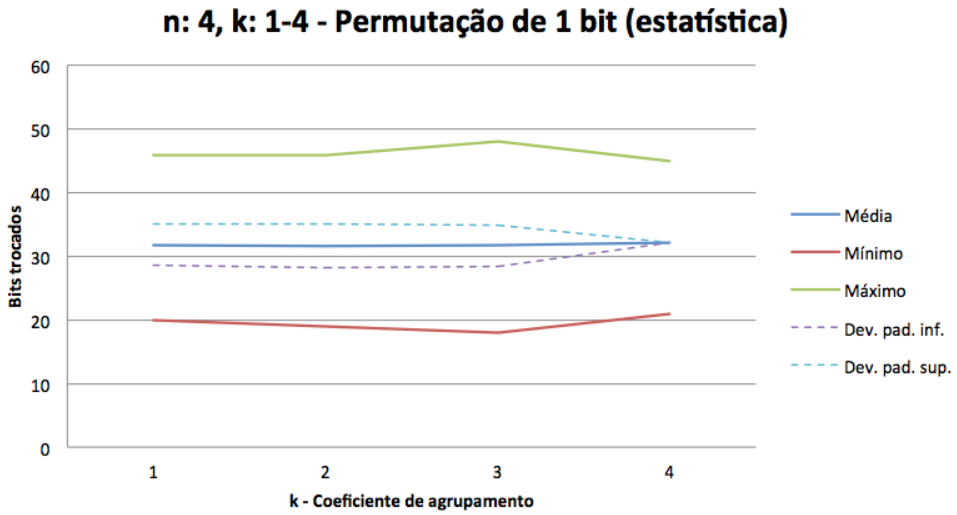

Figura 6.16: *Estatística dos* bits *alterados para a permutação binária.*

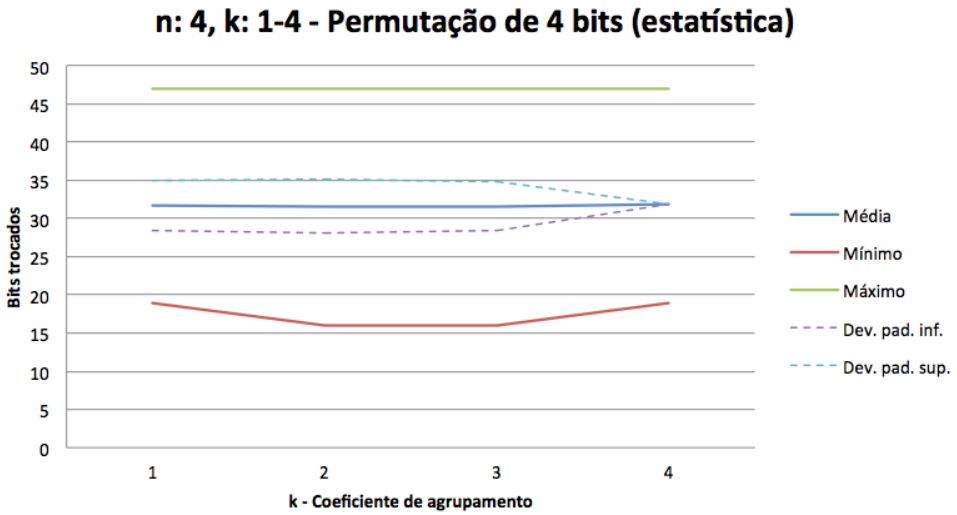

Figura 6.17: *Estatística dos* bits *alterados para a permutação de blocos de 4* bits.

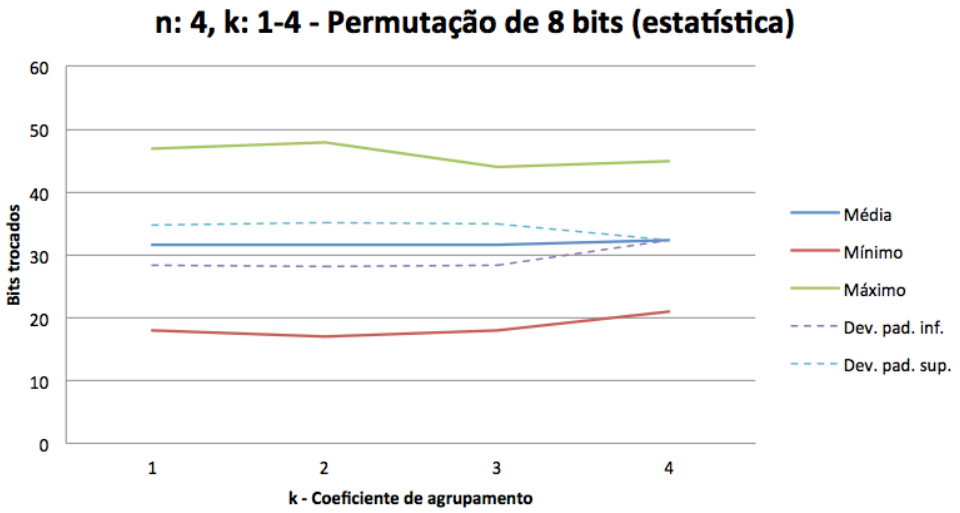

Figura 6.18: *Estatística dos* bits *alterados para a permutação de blocos de 8* bits.

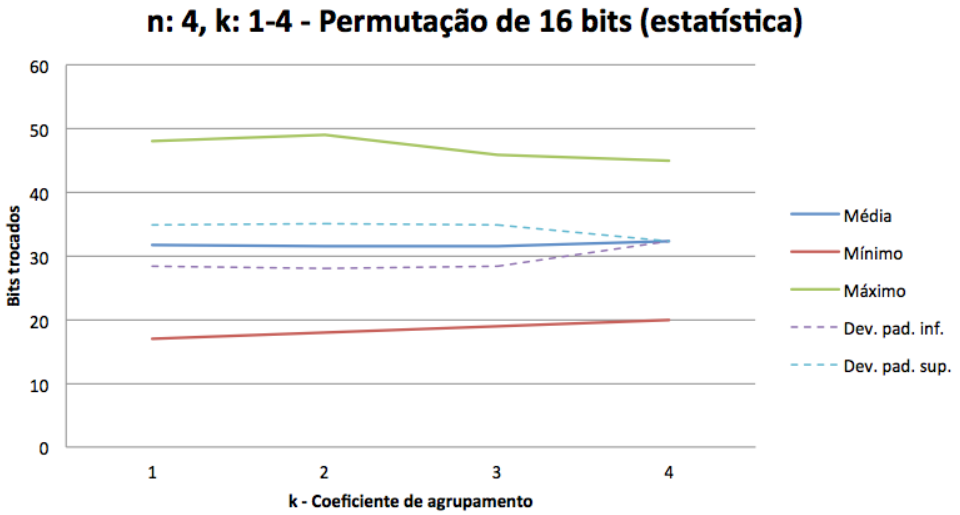

Figura 6.19: *Estatística dos* bits *alterados para a permutação de blocos de 16* bits.

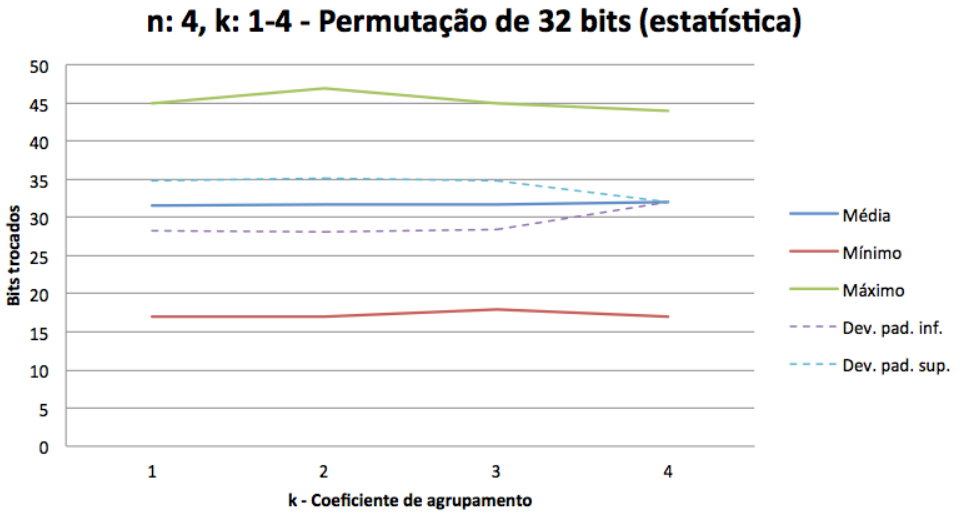

Figura 6.20: *Estatística dos* bits *alterados para a permutação de blocos de 32* bits.

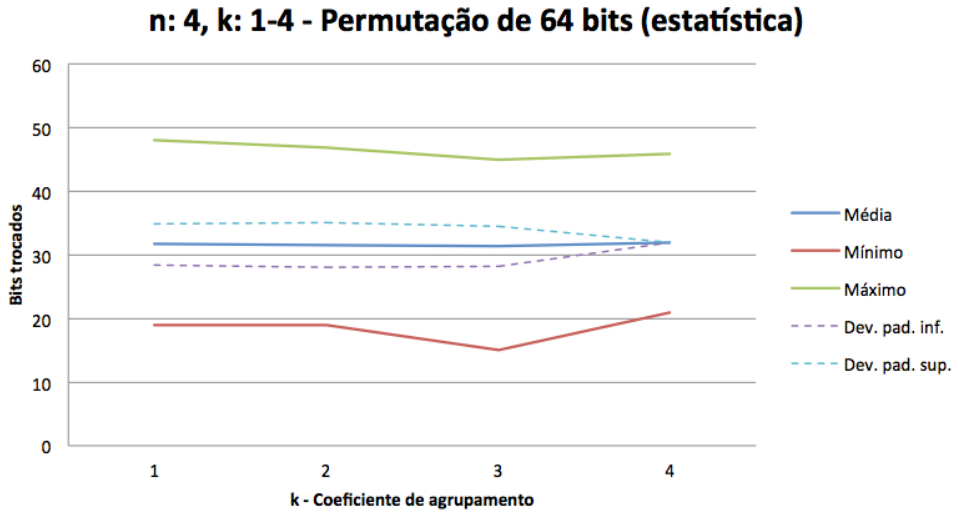

Figura 6.21: *Estatística dos* bits *alterados para a permutação de blocos de 64* bits.

Coeficientes de variação

Mostramos agora o coeficiente de variação (devio padrão pela média) de cada uma das estatísticas realizadas.

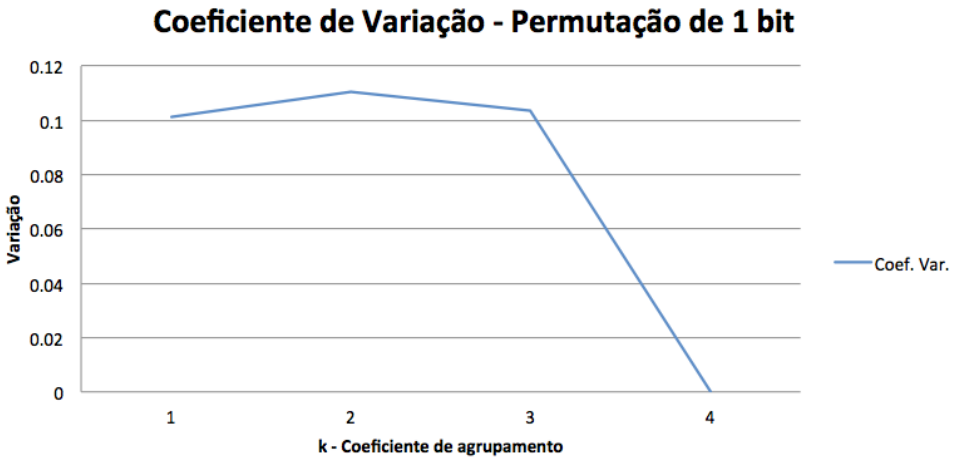

Figura 6.22: *Coeficiente de variação (desvio padrão pela média) para a permutação binária.*

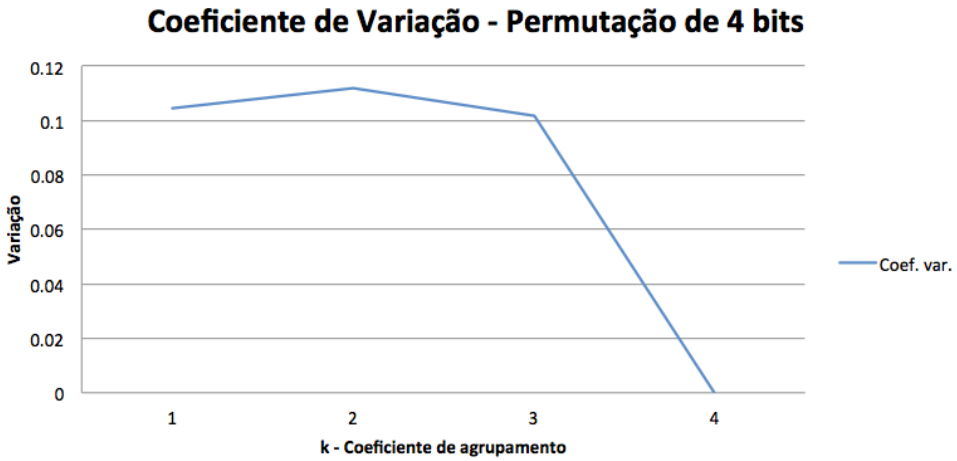

Figura 6.23: *Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 4* bits.

Observe que o último ponto em todos os gráficos vai a zero, isso ocorre porque só um único valor, dada combinação binomial de 4 a 4. Portanto, não há média, nem devio padrão.

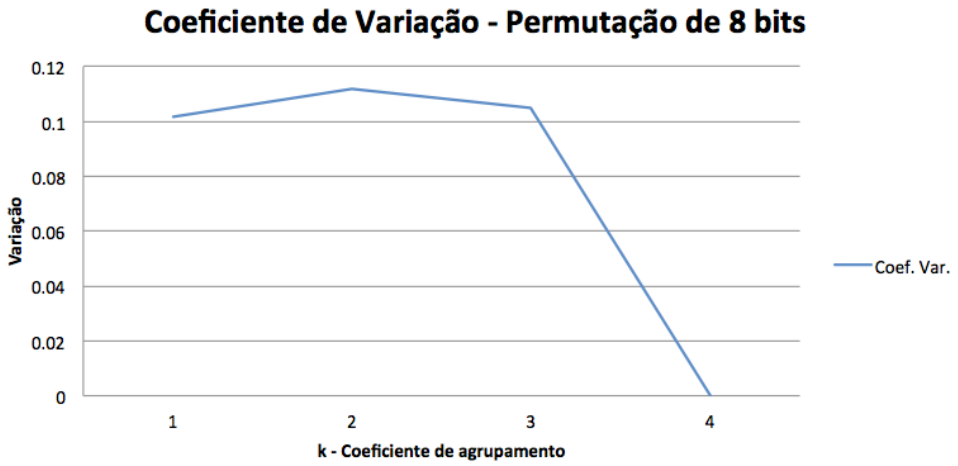

Figura 6.24: *Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 8* bits.

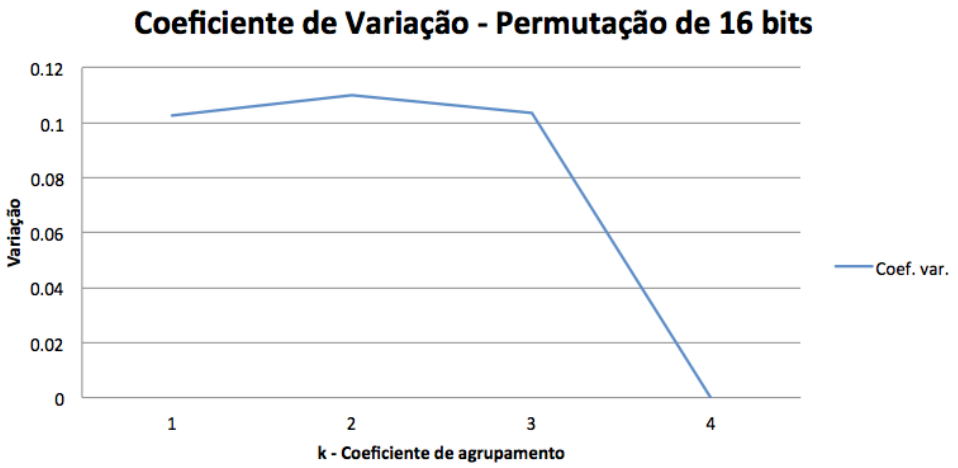

Figura 6.25: *Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 16* bits.

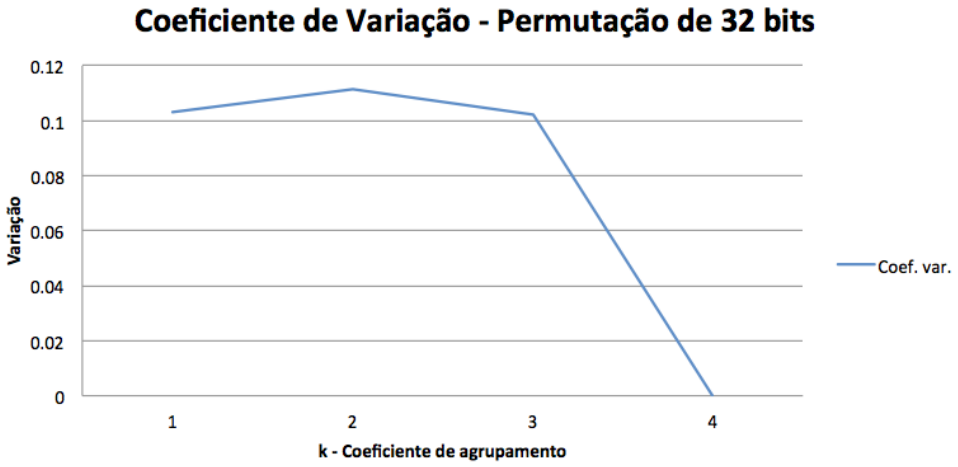

Figura 6.26: *Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 32* bits.

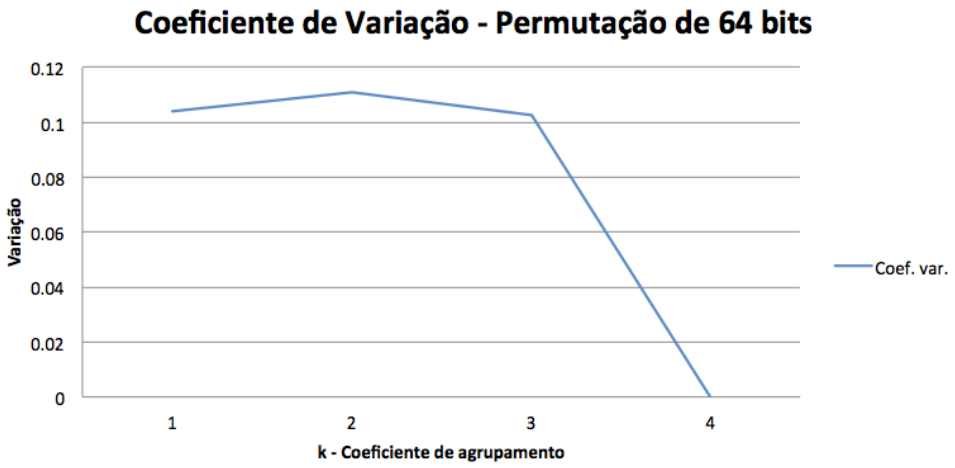

Figura 6.27: *Coeficiente de variação (desvio padrão pela média) para a permutação de blocos de 64* bits.

Média de bits trocados na operação SMP

Apresentamos aqui a média de bits trocados na operação SMP.

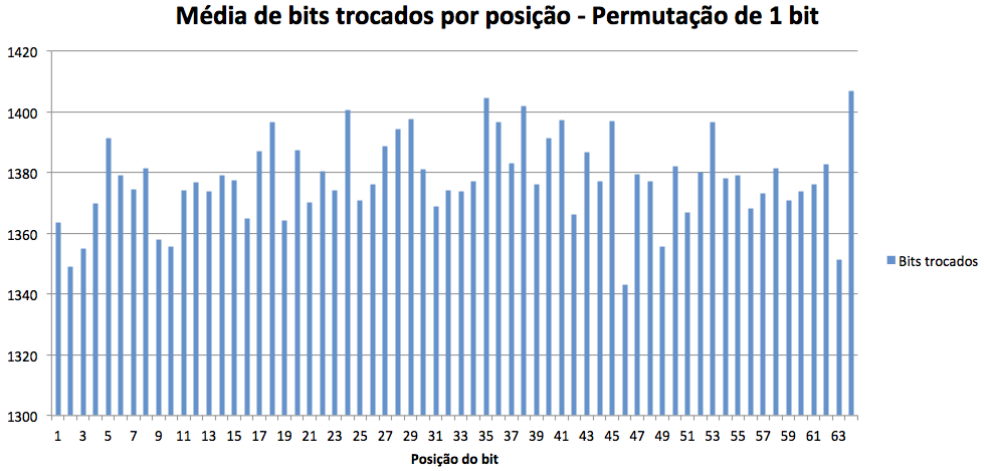

Figura 6.28: *Média de* bits *trocados por posição em permutação binária.*

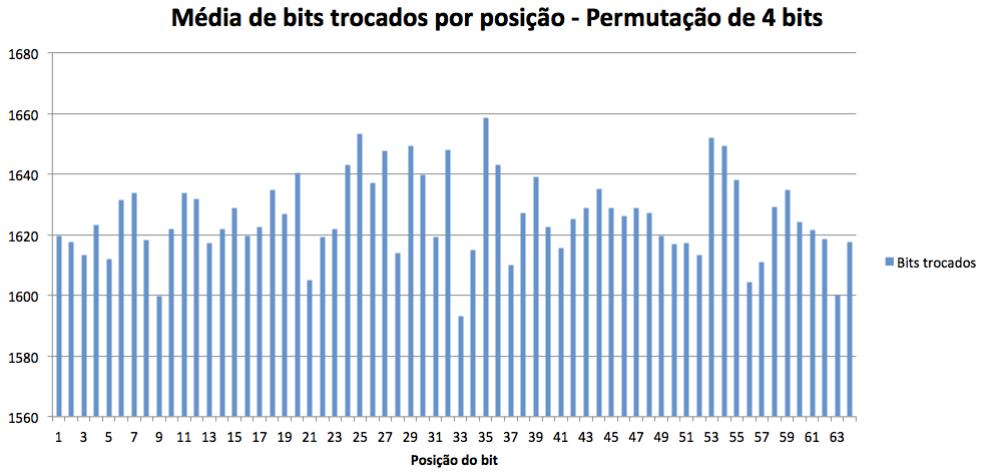

Figura 6.29: *Média de* bits *trocados por posição em permutação de blocos de 4* bits.

Neste capítulo, após o detalhamento das duas implementações e apresentação dos dados, podemos verificar que a cifra multicanal é viável na prática, onde o crescimento do número de canais não representa necessariamente um crescimento explosivo do tempo de processamento necessário para sua execução, mas que um projetista tem que ficar atento com o *overhead* da infraestrutura e considerar algumas limitações tecnológicas.

Em relação a criptanálise diferencial, vimos que a função de permutação é totalmente ineficiente, um atacante pode facilmente recuperar a chave secreta utilizada. Mas que a cifra modificada, SMP, tem as propriedades requeridas contra a criptanálise diferencial e de fato fortalece um sistema onde seja utilizada.

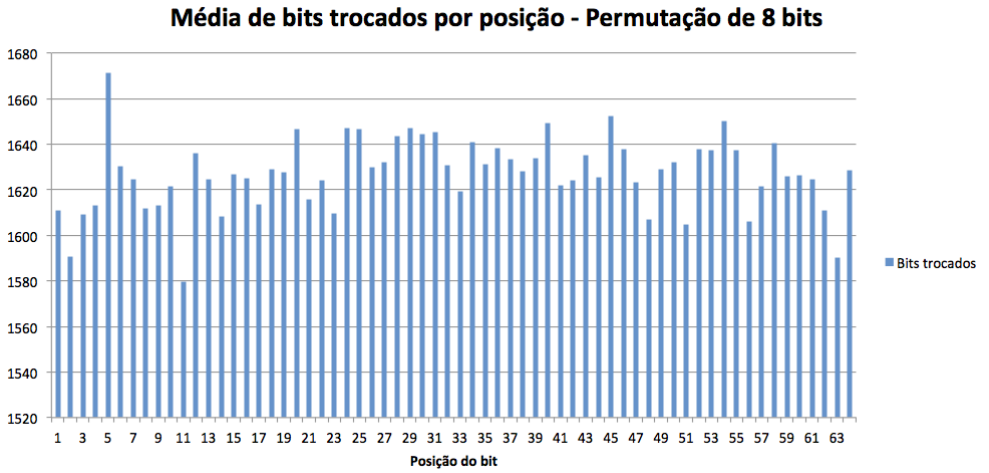

Figura 6.30: *Média de* bits *trocados por posição em permutação de blocos de 8* bits.

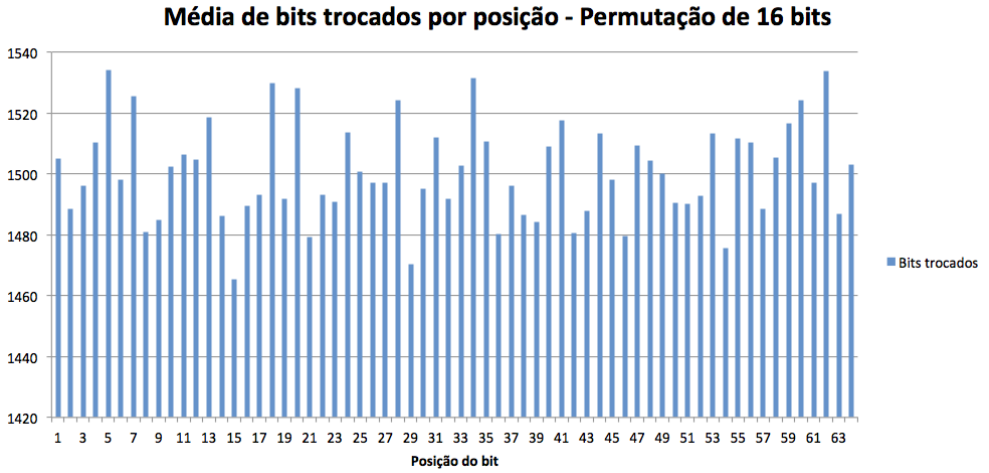

Figura 6.31: *Média de* bits *trocados por posição em permutação de blocos de 16* bits.

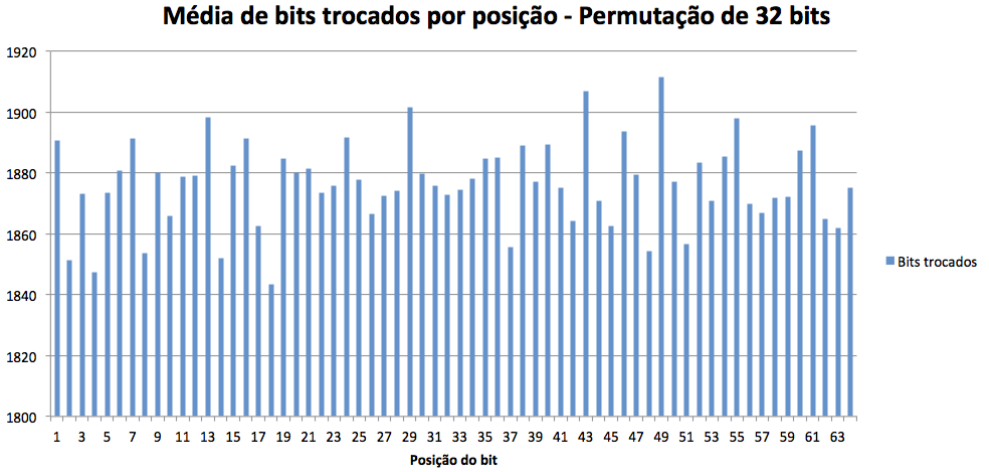

Figura 6.32: *Média de* bits *trocados por posição em permutação de blocos de 32* bits.

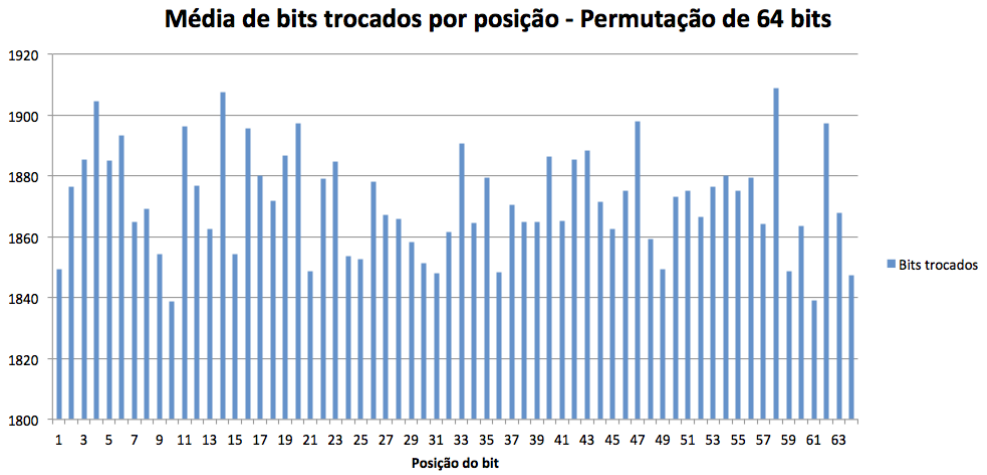

Figura 6.33: *Média de* bits *trocados por posição em permutação de blocos de 64* bits.

Capítulo 7

Conclusões

A cifra multicanal representa uma aplicação da nova instrução *RdRand* dos novos *chips* da *Intel*, uma operação simples que pode aumentar a segurança das futuras comunicações, modificada, a SMP, é resistente em relação a criptanálise diferencial, pode dar sobrevida a algumas cifras antigas fragilizadas por essa classe de ataques.

Removendo-se a dificuldade de obtenção dos blocos do texto-ilegível, por parte de um adversário, verificamos que ainda há uma certa complexidade sobre a combinação correta, o que explica porque a interceptação prática de dados das redes sem fio (telefonia celular, principalmente) são relativamente difíceis de serem obtidas e difundidas.

Em uma comunicação serializada (foi o caso, em nossos experimentos, com apenas uma interface de rede em ambos os agentes da comunicação), não representa um ganho de tempo, nem em transferência de dados, como se pudera pensar inicialmente, pois toda comunicação é em algum momento serializada pelo sistema operacional. Ainda sim, por outro lado não há uma explosão no tempo de processamento em função do número de canais, que nos experimentos foram dobrando em cada etapa, o que implica que a cifra pode realmente ser utilizada na prática, talvez com aplicações distribuídas apresente melhores tempos totais de comunicação.

Em relação à criptanálise diferencial, a permutação original da cifra multicanal mostrou-se totalmente ineficiente, e mais, na verdade um facilitador para obtenção da chave secreta. Mostra que esse tipo de ataque poderia ser empregado para analisar as redes que utilizam a técnica de FHSS, como meio de serialização dos dados.

Não obstante, a cifra multicanal modificada, a SMP, mostrou-se relativamente segura em relação a cifra multicanal, inclusive o mecanismo de difusão binária extraído do AES, poderia ser utilizado em outros sistemas como meio de fortalecê-los contra a criptanálise diferencial. Uma aplicação direta seria a própria rede GSM que poderia utilizá-lo depois da criptografia pelo A5 e antes da difusão por rádio frequência.

A cifra multicanal é uma proposta que requer mais estudos e experimentos, mas com os computadores cada vez mais interconectados e utilizando cada vez mais a computação em nuvem, podem surgir cenários onde se torne relevante e necessária.

7.1 Sugestões para Pesquisas Futuras

Aqui só avaliamos a cifra em relação a criptanálise diferencial, um trabalho semelhante poderia ser realizado em relação à criptanálise linear, uma outra técnica muito importante contra as cifras simétricas de bloco.

Um experimento de desempenho da SMP em relação ao AES poderia ser desenvolvido para a verificação se há ganho em eficiência de processamento na adoção da primeira.

Na cifra multicanal, os blocos são todos transmitidos para o cliente após a sua requisição. Um cenário de requisição e resposta poderia se explorado, de modo que o cliente devesse pedir cada um dos blocos ao servidor em uma sequência correta. Esse mecanismo teria a propriedade de identificar um intruso no sistema, pois o adversário ao não saber a sequência correta, poderia pedir um bloco

errado, o que o identificaria. Um precaução importante é separar esse mecanismo de identificação de um adversário do mecanismo de montagem dos dados, pois do contrário, bastaria ao adversário escutar os múltiplos canais e verificar a ordem em que estão sendo pedidos os dados, para obter a sequência correta de montagem dos mesmos.

Referências Bibliográficas

Barkan *et al.* **(2006)** Elad Barkan, Eli Biham e Nathan Keller. Instant ciphertext-only cryptanalysis of gsm encrypted communication (full version). Relatório técnico, Technion - Computer Science Departament. Citado na pág. 4

Biham e Shamir (1991) Eli Biham e Adi Shamir. Differential cryptanalysis of des-like cryptosystems. Em *CRYPTO'91*. Citado na pág. 1, 13, 14, 21

Cormen *et al.* **(2009)** Tomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd ed. ISBN 978-0-262-03384-8. Citado na pág. 17

Dunkelman *et al.* **(2010)** Orr Dunkelman, Nathan Keller e Adi Shamir. A practical-time attack on the a5/3 cryptosystem used in third generation gsm telephony. Cryptology ePrint Archive, Report 2010/013, 2010. http://eprint.iacr.org/. Citado na pág. 4

Eberspacher *et al.* **(2001)** Jorg Eberspacher, Hans-Joerg Vogel e Christian Bettstetter. *GSM Switching, Services and Protocols*. Wiley, 2nd. ed. Citado na pág. 4

Fazel e S.Kaizer (2008) K. Fazel e S.Kaizer. *Multi-Carrier and Spread Spectrum Systems*. Wiley, 2nd. ed. Citado na pág. 3

Hamming (1950) Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(147-160). Citado na pág. 26

Heys (2001) Howard Heys. A tutorial on linear and differential cryptanalysis. Relatório técnico. Citado na pág. 13

Hofemeier (2012) Gael Hofemeier. Intel digital random number generator (drng) - software implementation guide. http://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide, 2012. Acessado em 25/06/2013. Citado na pág. 21

Meel (1999) Jan Meel. Spread spectrum (ss) - introduction. Relatório técnico, Hoges Voor Wetenschap & Kunst de Nayer Instituut. Citado na pág. 3

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 ed. ISBN 0849385237. Citado na pág. 11

Michtchenko e Vilnaski (2007) Valentin Michtchenko e Yuri Vilnaski. *Harmed Texts and Multichannel cryptography*. Minsk - Enciclopedics, 1st. ed. Citado na pág. 2

Nohl (2009) Karsten Nohl. Gsm: Srsly? 26th Caos Communication Congress(26C3), 2009. Citado na pág. 4

Takács (2007) Péter Takács. The extension of cns-logic for multi-channel protocols. Em *Proceedings of the 7th International Conference on Applied Informatics*, volume 2, páginas 147–153. Eger. Citado na pág. 2

Terada (2008) Routo Terada. *Segurança de dados - Criptografia em rede de computador*. Blucher, 2a. ed. Citado na pág. 9, 15, 34, 37

Wegner (1960) Peter Wegner. A technique for counting ones in a binary computer. *Communications of the ACM*, 3(322). Citado na pág. 26

Wong e Stajano (2005) Ford-Long Wong e Frank Stajano. Multi-channel protocols. Em *In Proc. Security Protocols Workshop*. Springer. Citado na pág. 13