UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

SIMULADOR DE CODIFICAÇÃO DE CANAL PARA UM SISTEMA GSM COM CRIPTOFONIA

Autora:
Ana Fernanda Quaresma Batista Santos

Banca Examinadora:

Orientador:
Prof. Marcello Luiz Rodrigues de Campos, Ph.D.

Co-Orientador:
Fabiano Tondello Castoldi, M.Sc.

Examinador:
Prof. Mariane Rembold Petraglia, Ph.D.

Examinador:
Prof. Wallace Alves Martins, M.Sc.

DEL
Agosto de 2010

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

A Deus.

À minha família.

Agradecimentos

"Caminante no hay camino, se hace camino al andar." Antonio Machado Ruiz

Agradeço a Deus por tudo o que Ele faz em minha vida e por mais esta vitória. Agradeço pela minha família, amigos e outros anjos que pôs ao longo do meu caminho. Agradeço por todos os momentos alegres que tive durante esses anos de faculdade e também pelos momentos ruins e difíceis, pois são neles onde descobrimos nossa força e aprendemos mais.

Agradeço pela minha família, por ser meu porto seguro. Agradeço aos meus pais Ana Lúcia e Fernando pela educação que me deram e por todo carinho e amor que dedicam a mim desde sempre. Obrigada por incentivarem meus estudos e nunca me negarem nada quando se diz respeito aos estudos e ao aprendizado. Agradeço também ao meu irmão Guilherme, meu melhor amigo e meu exemplo, por todo carinho e cuidado. Agradeço a minha avó Emília por todo seu amor, por cuidar de mim, por me levar ao colégio e aos mais diversos cursos quando eu era criança e por me ter carregado no colo nos dias de chuva.

Agradeço as minhas amigas Januse Machado, Karla Alessandra, Karla Luraschy e Letícia Costa pela amizade e por compreenderem com paciência todas as vezes que tive que abrir mão de sair com elas para me dedicar aos trabalhos da faculdade.

Agradeço a Marcello Campos, meu orientador, por ter me dado a chance de fazer a iniciação científica e por me orientar neste trabalho. Agradeço a Fabiano Castoldi, meu co-orientador e amigo, por ter aceitado o desafio de me co-orientar, ajudar e por acreditar em mim. Agradeço à Mariane Rembold e Wallace Martins por terem aceitado a participar da minha banca. Agradeço também a todos os professores que tive na faculdade, principalmente os do Departamento de Engenharia Eletrônica e de Computação, por todo conhecimento, não apenas técnico, mas também de vida, que transmitiram com dedicação. Agradeço especialmente ao coordenador do curso professor Casé, não apenas pelos ensinamentos técnicos, mas também por me ajudar com os problemas que tive durante a graduação.

Logo que comecei a faculdade, conheci pessoas cuja amizade pretendo levar para vida toda, agradeço a elas por tornar os momentos que passei na UFRJ mais agradáveis. Agradeço às meninas super-poderosas Adriana Shulz e Camila Gussen, ao meu amigo desde a época do Ibeu Marcelo Tahiro, a Fernando Ferreira, Rafael Ribeiro, André Nunes e demais amigos da turma 04/01.

Apesar de não ser a turma com a qual comecei meus estudos na engenharia eletrônica, essa turma me acolheu e hoje me referencio a ela como "minha turma" por me sentir tão bem com os amigos que fiz aí. Agradeço a turma 04/02 por todos os momentos que passamos juntos, as horas que passamos estudando e queimando os neurônios fazendo os trabalhos, as aulas que assistimos juntos, os churrascos e conversas nos bares. Agradeço em especial a Alberto "Jorginho", Alberto Wagner, Amanda Alves, Isabel Santanna, Hugo Haas, Júlia Simões, Letícia Lemos (também conhecida como minha irmã gêmea), Pedro Coelho, Priscilla Luise, Rennan Roig e Ricardo Flach. Agradeço também a alguns amigos aleatórios que fiz ao longo da faculdade, em especial agradeço a Daniel Pinto, Luiz Monteso, Alan Dieguez, Priscilla Dinau, Andrei Battistel e Thiago Madureira.

Nesses últimos três anos de faculdade, tive o prazer de conviver com pessoas que passei admirar muito e me considero com sorte de ter amigos como os que eu fiz no LPS. Agradeço por serem sempre tão solícitos em me ajudarem a solucionar minhas dúvidas das mais diversas áreas e também pelos momentos de descontração nos almoços e lanches no Burguesão. Agradeço especialmente a Tadeu Ferreira, que foi quem eu mais perturbei durante esses anos e sempre me ajudou com a calma de sempre, um amigo que admiro muito. Agradeço também a Marcos Magalhães, vizinho e amigão, Fábio Freeland, Alessandro Dutra, Alexandre Leizor, Alexandre Ciancio, Rafael Paiva, Makus Lima, Leonardo Nunes, Thiago Prego, Flávio, Alan Tygel, Amaro Lima, Rafael de Jesus, Paulo César Filho, Rafael Amaro, Diego Felix, Dhiana Deva, Andressa, Felipe Grael, Rodrigo Torres, Felipe Diniz, Michel Pompeu, João Dias, Gabriel Matos, Rafael Amado, Waldir Sabino, Rodrigo Prates, Breno Espindola, Lucas Lago, Anderson Oliveira, Alex, Andreas Ellmauthaler, Marcus Fábio, Rodrigo Peres, Frederico Pinagé, Felipe Ribeiro, José Fernando e todos os amigos que fiz no LPS. Agradeço também aos funcionários do laboratório, Amanda, Thalia, Michele, Dinalva, Cláudio, Tiago, Waldir e todos os outros que sempre trabalharam pra manter o bom funcionamento do LPS e sempre nos ajudam quando precisamos.

No último ano de faculdade fiz amigos que com uma convivência curta e intensa se tornaram muito especiais em minha vida, pois nos seis meses que estive em Sevilla eles foram minha família e aprendemos muito juntos. Agradeço a família Eubranex, em especial Braulio Oliveira, Camila Lucio, Carine Barboza, Clarissa Gussen, Leonardo Bromberg, Letícia Goldani, Niala Pessuto, Thiago Cupertino e Rogério Miranda. Agradeço também a Ilana Gorayeb e Stephan Hollensteiner, coordenadores do projeto Eubranex, por me proporcionarem essa experiência única e enriquecedora.

Agradeço a todos os funcionários da Universidade e também as meninas do Burguesão e a tia Graça e seu Augusto, que sempre serviram meus lanchinhos com atenção e carinho. Agradeço também ao povo brasileiro, que através dos impostos proporcionou os estudos de qualidade que tive na UFRJ.

Agradeço também aos funcionários que mantêm os sites do Google e do Wikipédia, pois estes me ajudaram muito a tirar várias dúvidas de diversas áreas de conhecimento e enriquecerem meus trabalhos com mais informações.

Enfim, agradeço a todas as pessoas que de alguma maneira me ajudaram durante esses anos de faculdade.

Resumo

A telefonia móvel atualmente está presente na vida da grande maioria das pessoas e boa parte das redes implementadas no mundo utilizam a tecnologia GSM. Para realizar as conversas através desses sistemas temos a transmissão via ondas de rádio utilizando o ar como canal. Esta transmissão sofre muitas interferências que causam a destruição de parte da informação carregada pelo sinal. Para realizar uma boa recepção, faz-se o uso de codificadores de canal que estimam o sinal original recuperando o máximo possível da informação que eles carregam. Neste trabalho é implementado um simulador de codificação de canal para um tipo de transmissão segundo as especificações sugeridas pela 3GPP como padrão de rede, sendo assim possível ver os benefícios que esse tipo de codificação oferece. Um problema que pode ocorrer com os usuários do sistema GSM é a espionagem das conversas trocadas através da rede. Para solucionar isto, podem ser empregados sistemas de criptofonia impedindo que terceiros escutem as informações trocadas através da rede de telefonia celular. Existem algumas técnicas de criptofonia, como os cifradores analógicos, os quais são adequados para serem empregados em sistemas de telefonia móvel, já que não são necessárias modificações nos hardwares dos dispositivos para fazer uso de tal técnica de criptofonia. Para verificar a viabilidade e os efeitos da utilização dos cifradores analógicos em sistemas GSM, este trabalho também incluiu cifradores baseados em transformadas ortogonais e seu desempenho foi analisado sob diferentes níveis de SNR.

Palavras-Chave

GSM

BER

CRC

Código Convolucional

Viterbi

Criptofonia

DCT

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AMPS	*Advanced Mobile Phone Services*
AMR	*Adaptive Multi-Rate*
AuC	*Authentication Center*
AWGN	*Additive White Gaussian Noise*
BER	*Bit Error Rate*
BSC	*Base Station Controller*
BSS	*Base Station Subsystem*
BTS	*Base Transceiver Station*
CDMA	*Code Division Multiplex Access*
CELP	*Code Excited Linear Prediction*
CEPT	*Conférence Européenne des Postes Et Télécommunications*
CPFSK	*Continuous-Phase Frequency-Shift Keying*
CRC	*Cyclic Redundancy Check*
CSI	*Criptofonia por Segmentação da Informação*
CSI-F	*Criptofonia por Segmentação da Informação no domínio da Frequência*
CSI-T	*Criptofonia por Segmentação da Informação no domínio do Tempo*
D-AMPS	*Digital Advanced Mobile Phone Services*
DCT	*Discrete Cosine Transform*
DFT	*Discrete Fourier Transform*
EIR	*Equipament Identity Register*
ETSI	*European Telecommunications Standardisation Institute*
FDMA	*Frequency Division Multiplex Access*
FIR	*Finite Impulse Response*
IIR	*Infinite Impulse Response*

ISDN	*Integrated Services Digital Network*
IWF	*Interworking Functions*
GMSC	*Gateway Mobile Services Switching Center*
GMSK	*Minimum-Shift Keying*
GSM	*Global System for Mobile communications*
HLR	*Home Location Register*
LPC	*Linear Predictive Coding*
ME	*Mobile Equipment*
MS	*Mobile Station*
MSC	*Mobile Services Switching Center*
MSK	*Minimum-Shift Keying*
NMT	*Nordic Mobile Telephone*
NSS	*Network and Switching Subsystem*
OMC	*Operation and Maintenance Centers*
OSI	*Open System Interface*
OSS	*Operation Support Subsystem*
PESQ	*Perceptual Evaluation of Speech Quality*
PCM	*Pulse-Code Modulation*
SIM	*Subscriber Identity Module*
SNR	*Signal to Noise Rate*
SS7	*Signalling System 7*
TDMA	*Time Division Multiplex Access*
TRAU	*Transcoder Rate Adaptation Unit*
VLR	*Visitant Location Register*

Capítulo 1

Introdução

1.1 Histórico da Telefonia Celular

O homem está sempre tentando superar seus limites e atender suas necessidades, sendo uma destas a comunicação à distância. Para a comunicação entre pessoas em locais distantes foram utilizadas desde cartas, nos períodos medievais, até a telefonia, nos dias atuais.

A descoberta da transmissão de informação a longa distância via ondas de rádio foi um marco importante que modificou o cenário das telecomunicações. Atualmente grande parte da troca de informação por este meio utiliza televisores, telefones celulares, computadores, entre outros dispositivos.

As primeiras implementações de telecomunicação usando ondas de rádio foram realizadas no final do século XIX com a radiotelegrafia, que desde então tem sido largamente usada. A primeira aplicação sem fins militares foi com a radiodifusão por ser mais simples que a telefonia, uma vez que a informação é transmitida em apenas uma direção. Os sistemas de comunicação por rádio para o público civil só começaram a surgir após a segunda guerra mundial, quando já se conhecia a modulação em freqüência e já existiam alguns componentes eletrônicos como a válvula. O primeiro sistema de telefonia móvel nasceu oficialmente em Saint Louis (Missouri, USA) em 1946 [1].

Os primeiros sistemas de telefonia celular criados foram os sistemas analógicos, assim chamados porque fazem a transmissão da voz de forma analógica. Temos como exemplo destes o **AMPS** (*Advanced Mobile Phone Services*), sistema implantado nos Estados Unidos no final da década de 70, e também o **NMT**(*Nordic Mobile Telephone*), que começou a atuar na Suécia, Noruega, Dinamarca e Finlândia no início da década de 80 [2].

Com o passar dos anos a demanda por serviços de comunicação móvel crescia cada vez mais e os sistemas analógicos já não possuíam mais capacidade para atender a necessidade do mercado. Além disso a Europa sofria de um outro problema: os diversos sistemas em operação no continente eram incompatíveis entre si. Assim, o celular comprado em um país não funcionava nos demais.

Em 1982, na **CEPT** (*Conférence Européenne des Postes et Télécommunications*), foi criado um grupo chamado **GSM**, *Groupe Spécial Mobile*, com o objetivo de desenvolver especificações técnicas para uma nova rede de telefonia móvel comum à maioria dos países europeus. Esta nova rede deveria satisfazer os seguintes requisitos: permitir *roaming* pan-europeu, oferecendo compatibilidade de rede entre os diversos países do continente, interação com a rede de serviços digitais (**ISDN** - *Integrated Services Digital Network*), boa qualidade de áudio, uso eficiente das freqüências de rádio, alta capacidade, necessária para atender muitos usuários, e um bom nível de segurança para os assinantes.

Em 1988, foi criada a **ETSI**, a *European Telecommunications Standards Institute*, boa parte do trabalho realizado pela CEPT foi transferido para o novo grupo, incluindo as responsabilidades sobre o GSM. No ano de 1991, foi inaugurada na Europa a primeira rede GSM (cujo significado passou a ser *Global System for Mobile Communication*), usando as tecnologias de multiplexação **TDMA** e **FDMA**, e desde então o sistema tem crescido consideravelmente, não somente dentro do continente, mas também por todo o mundo.

Nos Estados Unidos também foram desenvolvidos sistemas digitais para substituir os sistemas analógicos: o **D-AMPS** (*Digital AMPS*), uma versão digital do AMPS mencionado anteriormente, e o **IS-95**, que utilizava como tecnologia de multiplexação o **CDMA**.

O número de assinantes de serviços móveis oferecidos pelo sistema GSM cresceu consideravelmente com o passar dos anos e, apesar do grande desenvolvimento da rede, existem formas de quebrar o sigilo da mesma possibilitando escutas de conversas telefônicas com certa facilidade. Cada usuário da rede pode ser alvo de espionagem e, dependendo do grau de confidencialidade da conversa, é recomendável a utilização de equipamentos de segurança para manter o sigilo das informações trocadas por meio de conversas telefônicas, equipamentos estes que até então estavam apenas à disposição de sistemas de comunicações militares e governamentais.

Essas medidas de segurança ganham importância à medida que casos de escuta não autorizada ganham relevância internacional e tornam-se realidade. Um caso muito famoso

ocorreu durante as Olimpíadas de Atenas, em 2004, quando centenas de celulares foram "grampeados" e dentre estes, linhas do Primeiro Ministro e sua esposa, Ministro da Defesa, Ministro da Justiça, empregados da Embaixada Americana e tantos outros mais. Este acontecimento ficou conhecido como *The Athens Affair* [3].

1.2 Descrição da Rede GSM

Como mostra a Figura 1.1, o sistema GSM pode ser dividido em: Estação Móvel, Subsistema da Estação-Base, Subsistema de Rede e Comutação e Subsistema de Operação e Manutenção.

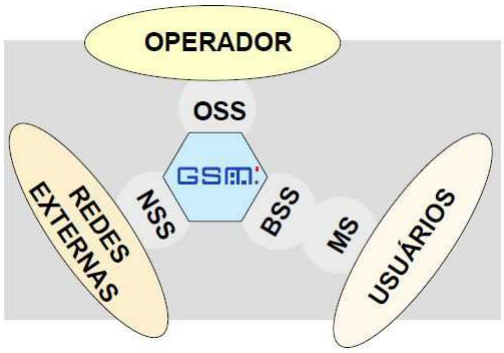

Figura 1.1: Subsistemas do GSM

1.2.1 Estação Móvel

A Estação Móvel (**MS** - *Mobile Station*) é o equipamento que funciona como interface entre a rede e o assinante. Ela é composta pelo equipamento móvel (**ME** - *Mobile Equipment*), que não está relacionado ao assinante, e pelo Módulo de Identidade do Assinante (**SIM** - *Subscriber Identity Module*), que é responsável por armazenar informações sobre o cliente, relacionadas à assinatura e informações de segurança, e é de uso individual. Quando o assinante insere o SIM em um ME, o equipamento então passa a pertencer ao cliente, recebendo as chamadas direcionadas a ele.

1.2.2 Subsistema de Estação-Base

O Subsistema de Estação-Base (**BSS** - *Base Station Subsystem*) é composto pela infraestrutura específica da parte de rádio celular do GSM e tem a função de conectar a MS ao NSS. Um BSS contém um Controlador de Base (**BSC** - *Base Station Controller*) e várias Estações Rádio Base (**BTS** - *Base Transceiver Station*), podemos ver na Figura 1.2 o esquema de um BSS simplificado com apenas uma BTS. As BTSs são responsáveis pela transmissão e recepção de dados entre as MSs e o BSS através de ondas de rádio e por isso incluem as antenas e todo processamento de sinal específico da interface de rádio.

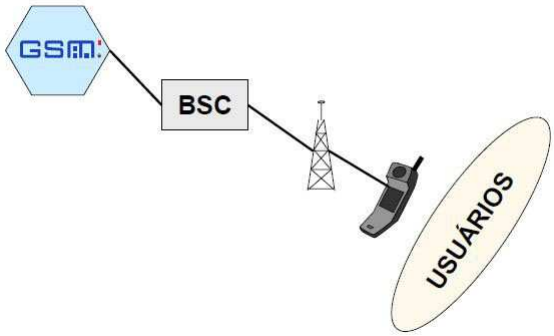

Figura 1.2: Subsistema de Estação-Base.

Um equipamento importante da BTS é a Unidade de Transcodificação e Adaptação (**TRAU** - *Transcoder Rate Adaptation Unit*). Ela é um equipamento específico do GSM, que faz a codificação e decodificação de voz e a adaptação da taxa para a transmissão de dados. Apesar da TRAU ser considerada uma subparte da BTS, ela também pode ser alocada longe da BTS, geralmente na Central de Comutação de Serviços Móveis (**MSC** - *Mobile Services Switching Center*). Nessa posição há a vantagem da transmissão entre a BTS e a TRAU ser mais compacta, com uma taxa menor, como podemos ver melhor na Figura 1.3.

O BSC é o componente do BSS encarregado pela administração dos canais da interface de rádio e dos *handovers*[1]. O BSC é conectado a diversas BTSs por um lado e ao NSS por outro, mais exatamente à MSC (*Mobile Services Switching Center*).

[1] *Handover*, também conhecido por handoff também, é o procedimento que permite fazer a transição de MSs entre células de forma transparente para o usuário.

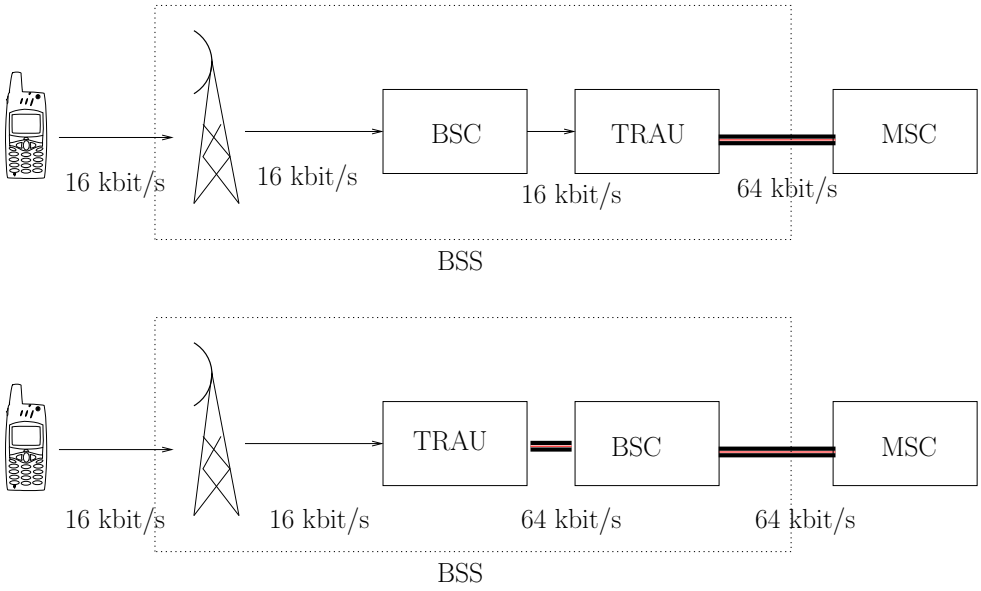

Figura 1.3: Possíveis posições para a TRAU.

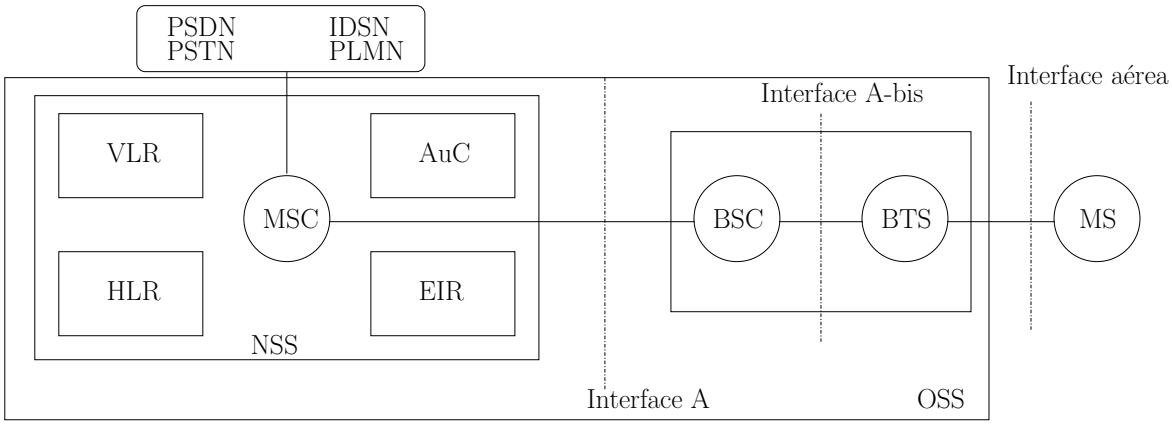

Figura 1.4: Componentes do sistema GSM e suas interfaces

1.2.3 Subsistema de Rede e Comutação

O Subsistema de Rede e Comutação (**NSS** - *Network Switching Subsystem*) possui as principais funções de comutação da rede GSM, além de armazenar dados dos assinantes para o controle de mobilidade. De uma maneira geral, pode-se dizer que a mais importante tarefa do NSS é conectar a rede GSM com as demais redes de telecomunicação existentes. Para isso, o NSS é dividido em algumas subpartes com funções específicas descritas a seguir e que são apresentadas na Figura 1.4.

Todas as ligações para os assinantes GSM e originadas por eles são coordenadas pela Central de Comutação de Serviços Móveis (**MSC** - *Mobile Services Switching Center*). Para

administrar essas ligações, a MSC controla em uma ponta alguns BSSs e na outra ponta se conecta a outras redes externas. O NSS necessita de um suporte de sinalização para realizar essa comunicação através da interface que liga o sistema às redes externas, para tal faz uso dos protocolos do Sistema de Sinalização 7 (**SS7** - *Signalling System 7*).

Além do MSC, o NSS também possui bancos de dados. Informações relevantes dos assinantes para o aprovisionamento dos serviços de telecomunicação, tais como a localização temporária do cliente, ficam armazenadas no Registro de Localização de Origem (**HLR** - *Home Location Register*). Este registro ainda é subdividido em duas partes, o Centro de Autenticação (**AuC** - *Authentication Center*), que tem o papel de administrar as informações sigilosas dos assinantes necessárias para fazer autenticação do usuário no sistema, e o Registro de Identidade de Equipamento (**EIR** - *Equipament Identity Register*), que guarda a informação sobre o ME.

O NSS possui, além desses equipamentos, o *gateway* MSC (**GMSC** - *Gateway Mobile Services Switching Center*) que é o responsável por fazer a conexão entre a MSC onde o usuário está registrado com redes externas.

1.2.4 Subsistema de Operação e Manutenção

O Subsistema de Operação e Manutenção (**OSS** - *Operations Support Subsystem*) interage com quase todos os equipamentos de infraestrutura da rede GSM para realizar suas funções, que podem ser divididas em três tipos: operação e manutenção da rede, gerenciamento de assinantes e gerenciamento de equipamentos móveis.

O serviço de operação e manutenção da rede que o OSS executa atua como uma interface entre os operadores e todas as máquinas do sistema. Por um lado a operação de rede está conectada às máquinas de telecomunicação (MSCs, BSCs, HLRs entre outras, exceto as BTSs que são acessadas através de suas BSCs) e por outro lado conecta-se às estações de trabalho que funcionam como interfaces homem-máquina. Estas são máquinas dedicadas e, quando encontradas em arquiteturas de simples manutenção, autônomas, são chamadas de Centros de Operação e Manutenção (**OMC** - *Operation and Maintenance Centers*).

As funções de gerenciamento de assinantes são independentes daquelas de operação e manutenção da rede e são executadas por máquinas diferentes. De um modo geral, estas funções são divididas na gestão dos dados do assinante e na cobrança de chamadas.

Por fim, o gerenciamento de equipamentos móveis faz um controle dos MEs através dos

números de identificação que ficam armazenados no EIR, sua tarefa está concentrada nesta máquina. Como os telefones celulares podem servir a vários usuários facilmente, apenas trocando o SIM no ME, esse controle dos dispositivos visa dificultar que aparelhos celulares roubados sejam usados.

1.3 Objetivos

O objetivo desse projeto é fazer um simulador do sistema GSM que sirva para investigar o funcionamento da codificação de canal utilizado pelo mesmo. O simulador deve conter características relevantes para seu funcionamento e são especificadas pela 3GPP para o padrão da rede, assim as simulações realizadas poderão retratar um contexto próximo de um sistema real, gerando resultados confiáveis.

Como vimos na Seção 1.1, a segurança das conversas telefônicas é uma questão importante, apesar de não ter apresentado tanto desenvolvimento como algumas outras áreas do sistema GSM. Este projeto também visa implementar técnicas de criptofonia abordadas em *Criptofonia Aplicada a Sistemas Modernos de Comunicações* [3] para que se possa observar os efeitos do canal GSM sobre esses sistemas. As técnicas propostas têm por fim aumentar a segurança das conversas utilizando a telefonia celular sem que sejam necessárias modificações no hardware do ME.

1.4 Organização do Trabalho

Este projeto está dividido em seis capítulos que abordam o sistema GSM, a codificação de canal e a criptofonia conforme a descrição a seguir:

Capítulo 1 Este capítulo visa introduzir o leitor ao contexto que envolve o assunto abordado ao longo deste trabalho, para isto apresenta um breve histórico da telefonia celular e uma descrição da rede GSM. Também são apresentados os objetivos do projeto e a estrutura do mesmo.

Capítulo 2 Neste capítulo são apresentados os conceitos teóricos relacionados à codificação de canal, como é realizada a codificação de canal: o desenvolvimento matemático das técnicas utilizadas no projeto e como é feita a decodificação. Uma explicação sobre a modulação de canal utilizada pelo sistema GSM também se encontra neste capítulo.

Capítulo 3 Este capítulo é dedicado à explicação do tratamento que o sinal de voz recebe. Primeiramente é introduzido o conceito de codificação de voz e são apresentados os tipos de codificadores existentes. Em seguida o leitor será apresentado às técnicas de criptofonia de uma maneira geral, recebendo explicações mais detalhadas sobre as técnicas que atendem aos objetivos deste trabalho.

Capítulo 4 Este capítulo é dedicado à descrição do simulador implementado neste trabalho de maneira que o leitor possa conhecer melhor sua estrutura, seu funcionamento e o fluxo de informação trocado entre os módulos do simulador.

Capítulo 5 Neste capítulo são apresentadas as simulações realizadas com o simulador implementado, os parâmetros usados nas mesmas, os resultados obtidos e uma breve discussão sobre estes.

Capítulo 6 Este capítulo contém as conclusões obtidas ao longo da realização deste projeto, bem como sugestões de trabalhos futuros.

Capítulo 2

Codificação de Canal em um Sistema GSM

2.1 Processo de Comunicação Sem Fio

Todo processo de comunicação pode ser representado por seus três principais elementos: a fonte, o destinatário e o sistema de comunicação, sendo este último composto pelo transmissor, pelo canal e pelo receptor. O canal é o meio por onde a mensagem produzida pela fonte é transmitida ao destinatário. Dependendo do sistema de comunicação podemos ter diversos tipos de canal, como o par trançado, a fibra ótica, ou a atmosfera. No caso de comunicação sem fio, o meio utilizado para a transmissão da mensagem é a atmosfera.

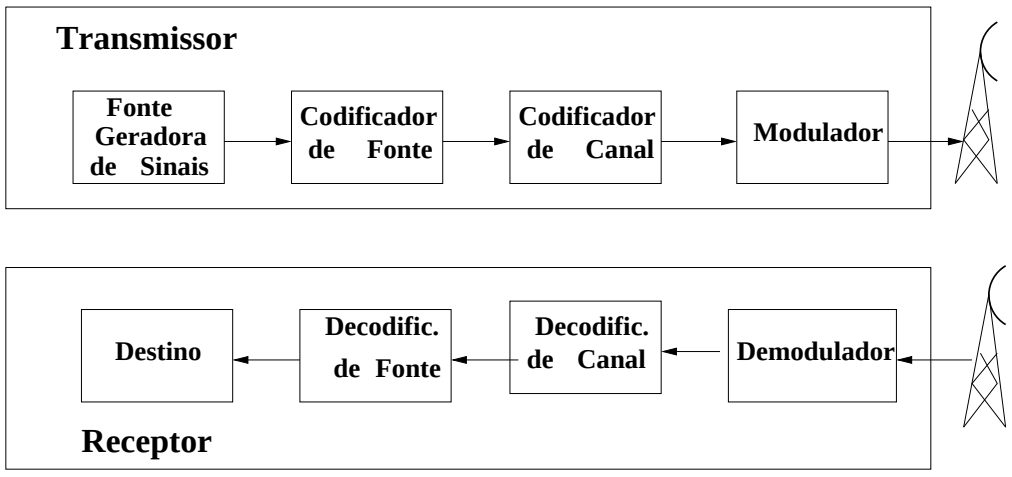

Figura 2.1: Processo de comunicação sem fio.

Como podemos ver na Figura 2.1, o transmissor pode ser subdividido em codificador de

fonte, codificador de canal e modulador. A codificação de fonte retira informação redundante da mensagem tornando-a o mais compacta possível para passar pelo canal e transforma os dados num alfabeto finito de símbolos que os represente e que permita a recuperação dos mesmos pelo decodificador. A codificação de canal introduz informação redundante à mensagem de forma a torná-la robusta aos erros introduzidos pelo canal, sendo assim podemos recuperar a mensagem original, ou pelo menos boa parte dela, no receptor. O modulador prepara o sinal para ser transmitido pelo canal; ele mapeia os dados em uma constelação que é conhecida tanto pelo transmissor quanto pelo receptor.

O receptor também é subdividido em três partes: o demodulador, o decodificador de canal e o decodificador de fonte. Estes elementos são responsáveis pela recuperação da estimativa da mensagem original transmitida e, para que esta recuperação seja feita, executam um processo inverso ao realizado pelo transmissor. O demodulador recebe o sinal entregue pelo canal e estima quais símbolos foram enviados pelo transmissor. O decodificador de canal recebe os símbolos estimados pelo demodulador, retira os bits de redundância inseridos pelo codificador de canal e com eles detecta se o sinal recebido possui erros, corrigindo-os se possível. Por fim, o decodificador de fonte recebe o sinal do codificador de canal e o interpreta segundo o alfabeto finito de símbolos utilizado pelo codificador de fonte.

Um dos objetivos do trabalho proposto é simular a codificação de canal feita no sistema GSM. Segundo as especificações, são utilizados dois tipos de codificação de canal: código de verificação de redundância cíclica (**CRC** - *Cyclic Redundancy Check*) e códigos convolucionais. A decodificação não é padronizada, portanto esta parte do sistema é aberta e cada fabricante pode escolher o algoritmo que desejar para recuperar os dados codificados.

2.2 Codificação Cíclica

Um dos códigos utilizados para fazer a codificação de canal no sistema GSM é o código de verificação de redundância cíclica, ou apenas CRC, que é um tipo de código cíclico. Os códigos cíclicos formam uma importante subclasse dos códigos lineares por blocos e a seguir vamos entender como é feita a codificação usando este últimos.

Consideremos que os símbolos binários que desejamos codificar estão separados em blocos de tamanho fixo k. Iremos denominar cada "bloco de mensagem" como $\mathbf{m}$. Cada mensagem $\mathbf{m}$ será transformada em uma n-upla binária $\mathbf{c}$, sendo $n > k$. A n-upla $\mathbf{c}$ é

chamada de palavra-código, ou ainda de vetor-código, de $\mathbf{m}$. Temos 2^k possíveis mensagens e, portanto, 2^k possíveis palavras-código correspondentes que formam um conjunto chamado de bloco de código. Para esse código ser útil, as 2^k palavras-código precisam ser distintas. Assim, a função de transformação que leva as 2^k possíveis mensagens ao bloco de código deve ser bijetora.

Definição: Um bloco de código de tamanho n com 2^k palavras-código é chamado de código linear (n,k) se e somente se suas 2^k palavras-código formam um subespaço vetorial de dimensão k com todas suas n-uplas pertencentes ao corpo GF(2)[1] [4].

Um código binário só é linear se a soma módulo dois de duas palavras-código também é uma palavra-código. Então o código linear C (n,k) forma um subespaço vetorial k-dimensional com todas as combinações lineares de suas palavras-código:

$$\mathbf{c} = m_0\mathbf{g_0} + m_1\mathbf{g_1} + ... + m_{k-1}\mathbf{g_{k-1}}$$

onde $m_i = 0$ ou 1 para todo $0 \leq i < k-1$ e $\mathbf{g}_i$ é um vetor linha pertencente a matriz $\mathbf{G}$ como mostra a Equação (2.3).

Podemos escrever estas combinações também na forma matricial:

$$\begin{aligned} \mathbf{c} &= \mathbf{m} \cdot \mathbf{G} \\ &= \begin{bmatrix} m_0 & m_1 & \dots & m_{k-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} \\ &= m_0\mathbf{g_0} + m_1\mathbf{g_1} + ... + m_{k-1}\mathbf{g_{k-1}} \end{aligned} \tag{2.1}$$

sendo

$$\mathbf{m} = \begin{bmatrix} m_0 & m_1 & \dots & m_{k-1} \end{bmatrix} \tag{2.2}$$

[1] GF(n) significa Corpo de Galois, em inglês *Galois Field*, também conhecido por Corpo Finito. Este corpo contém n elementos. Se n=2, por exemplo, então o corpo é formado pelos elementos {0,1}.

e

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix} \tag{2.3}$$

As linhas da matriz $\mathbf{G}$ são compostas por k vetores linearmente independentes que geram o subespaço k-dimensional que contém todas as palavras-código, por essa razão a matriz $\mathbf{G}$ é chamada de matriz geradora. Através da multiplicação da mensagem $\mathbf{m}$, que queremos codificar, pela matriz $\mathbf{G}$ obtemos a palavra-código que será transmitida, como mostra a equação (2.1). Essa matriz ainda pode ser construída de duas maneiras de forma a criar um código *sistemático* ou *não sistemático*. Em um código sistemático, a geração da palavra-código consiste em adicionar $n-k$ bits de paridade no final, ou no início, do bloco de mensagem $\mathbf{m}$, por essa razão é interessante trabalhar com este tipo de código uma vez que ele facilita a decodificação.

$$\mathbf{c} = \begin{bmatrix} b_0 & b_1 & \ldots & b_{n-k-1} & m_0 & m_1 & \ldots & m_{k-1} \end{bmatrix} \tag{2.4}$$

onde os bits b_i, com $i = 1, 2, ..., n-k-1$, são os bits de paridade que são gerados por

$$b_i = p_{0i}m_0 \oplus p_{1i}m_1 \oplus p_{2i}m_2 \oplus \ldots \oplus p_{k-1,i}m_{k-1}$$

Escrevendo a matriz geradora na forma sistemática, temos:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \mathbf{g_2} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \ldots & 0 \\ p_{10} & p_{11} & p_{12} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \ldots & 0 \\ p_{20} & p_{21} & p_{22} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & & & & & \\ p_{k-1,0} & p_{k-1,1} & p_{k-1,2} & \vdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{2.5}$$

Podemos observar que a matriz geradora é composta por uma matriz de paridade $\mathbf{P}$ de tamanho $k \times n-k$ e uma matriz identidade $\mathbf{I_k}$, $\mathbf{G} = [\mathbf{P},\ \mathbf{I_k}]$.

Outra matriz importante associada ao código linear C é a matriz de cheque de paridade, a qual será representada por $\mathbf{H}$. Assim como a matriz $\mathbf{G}$ tem k linhas linearmente independentes, a matriz $\mathbf{H}$ tem $n-k$ linhas linearmente independentes e ortogonais aos vetores linha de $\mathbf{G}$. Portanto podemos dizer que o espaço gerado por $\mathbf{G}$ é ortogonal ao espaço

gerado por $\mathbf{H}$, ou seja, todas as palavras-código $\mathbf{c}$ geradas por $\mathbf{G}$ pertencerão ao espaço nulo de $\mathbf{H}$. A matriz de cheque de paridade será composta por $\mathbf{P^T}$, que é a transposta da matriz $\mathbf{P}$, e por $\mathbf{I_{n-k}}$, que é a matriz identidade de ordem $n-k$.

$$\mathbf{H} = [\mathbf{I}_{n-k},\, \mathbf{P}^T] \quad , \qquad (n-k) \times k$$

Então temos que

$$\begin{aligned} \mathbf{G} \cdot \mathbf{H}^T &= \begin{bmatrix} \mathbf{P} & \mathbf{I}_k \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_{n-k} \\ \mathbf{P} \end{bmatrix} \\ &= \mathbf{P}_{k,n-k} \cdot \mathbf{I}_{k,k} \oplus \mathbf{I}_{n-k,n-k} \cdot \mathbf{P}_{k,n-k} \\ &= \mathbf{P} \oplus \mathbf{P} \;=\; \text{matriz nula} \;=\; \mathbf{0}_{k,n-k} \end{aligned}$$

e como estamos trabalhando no corpo GF(2), portando usando somas módulo 2, ao somar a matriz $\mathbf{P}$ com ela própria, nos índices onde tínhamos 1s agora teremos $1 \oplus 1 = 0$. Por essa razão vamos obter uma matriz nula.

Então se $\mathbf{c}$ é uma palavra-código gerada por $\mathbf{G}$

$$\begin{aligned} \mathbf{c} \cdot \mathbf{H}^T &= \mathbf{0} \\ \mathbf{m} \cdot \mathbf{G} \cdot \mathbf{H}^T &= \mathbf{0} \\ \mathbf{m} \cdot \mathbf{0} &= \mathbf{0} \end{aligned} \tag{2.6}$$

Se a matriz geradora do código linear por blocos C (n,k) está na forma sistemática como na equação (2.5), a matriz de cheque de paridade associada a ela estará na seguinte forma:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h_0} \\ \mathbf{h_1} \\ \mathbf{h_2} \\ \vdots \\ \mathbf{h_{n-k-1}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & p_{00} & p_{10} & p_{20} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \dots & 0 & p_{01} & p_{11} & p_{21} & \cdots & p_{k-1,1} \\ 0 & 0 & 1 & \dots & 0 & p_{02} & p_{12} & p_{22} & \cdots & p_{k-1,2} \\ & & & & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & p_{2,n-k-1} & \vdots & p_{k-1,n-k-1} \end{bmatrix} \tag{2.7}$$

Exemplo 1 A Tabela 2.1 descreve um código linear (7,4). Para este código temos a matriz geradora $\mathbf{G}$ e a matriz de cheque de paridade $\mathbf{H}$ apresentadas a seguir, ambas na forma sistemática.

Mensagens	**Palavras-código**
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

Tabela 2.1: Código Linear $k = 4$ e $n = 7$

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Considere que queremos codificar a mensagem [1 1 0 0], para isso multiplicamos esse vetor pela matriz $\mathbf{G}$ como apresentado pela equação (2.1) e teremos a palavra-código que encontramos na quarta linha da tabela:

$$\mathbf{c} = [1\ 1\ 0\ 0] \cdot \mathbf{G} = [1\ 0\ 1\ 1\ 1\ 0\ 0]$$

Se agora multiplicarmos a palavra-código pela matriz $\mathbf{H}^T$, teremos um vetor de zeros, como visto na equação (2.6):

$$\mathbf{s} = [1\ 0\ 1\ 1\ 1\ 0\ 0] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = [0\ 0\ 0]$$

E como esperado o vetor código $\mathbf{c}$ pertence ao espaço nulo de $\mathbf{H}$.

Agora vamos considerar que ao enviar a palavra-código $\mathbf{c}$ por um canal ruidoso esta foi corrompida e, em vez de receber o vetor $\mathbf{c}$, recebemos $\mathbf{r} = \mathbf{e} + \mathbf{c}$, sendo $\mathbf{e}$ uma n-upla chamada *vetor de erro*. O decodificador, ao receber o vetor $\mathbf{r}$, o multiplicará por $\mathbf{H}^T$ e tentará encontrar a mensagem $\mathbf{m}$ correspondente a ele, ou seja,

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = \begin{bmatrix} s_0 & s_1 & \dots & s_n - k - 1 \end{bmatrix} \tag{2.8}$$

A $(n-k)$-upla $\mathbf{s}$ apresentada pela equação (2.8) é conhecida por *síndrome*. Ela indica se o vetor recebido foi corrompido ou não pelo canal, ou seja, se o sinal recebido for igual à palavra-código enviada, então $\mathbf{s} = \mathbf{0}$, como vimos no Exemplo 1, porém se $\mathbf{r}$ não for igual a $\mathbf{c}$, então $\mathbf{s} \neq \mathbf{0}$.

$$\mathbf{s} \ = \ \mathbf{r} \cdot \mathbf{H}^T \ = \ (\mathbf{c} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{c} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T$$

Como $\mathbf{c}$ pertence ao espaço nulo de $\mathbf{H}^T$, como foi mostrado pela equação (2.6), então

$$\mathbf{s} = \mathbf{e} \cdot \mathbf{H}^T$$

Portanto a síndrome depende apenas do vetor de erro, se este for nulo, assim também será a síndrome, mas se o vetor de erro não for nulo, $\mathbf{s} \neq \mathbf{0}$.

Exemplo 2 Vamos considerar que enviando a mensagem que codificamos no Exemplo 1, esta foi corrompida pelo erro [0 0 0 0 1 0 0]. O vetor que recebemos é

$$\begin{aligned}\mathbf{r} &= \mathbf{c}+\mathbf{e}\\ &= [1\ 0\ 1\ 1\ 1\ 0\ 0]\oplus[0\ 0\ 0\ 0\ 1\ 0\ 0]\\ &= [1\ 0\ 1\ 1\ 0\ 0\ 0]\end{aligned}$$

Calculando a síndrome, temos

$$\begin{aligned}\mathbf{s} &= [1\ 0\ 1\ 1\ 0\ 0\ 0]\cdot\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&0\\0&1&1\\1&1&1\\1&0&1\end{bmatrix}\\ \mathbf{s} &= [0\ 1\ 1]\\ \mathbf{s} &\neq \mathbf{0}\end{aligned}$$

Dentre os códigos lineares por blocos, existe um grupo de códigos que, além de ter todas as propriedades vistas até este momento, possui uma característica especial: qualquer deslocamento cíclico provocado em uma palavra-código gera também uma palavra-código. Por essa razão este códigos são chamados de cíclicos. Vamos considerar que a palavra-código $\mathbf{c} = [c_0\ c_1\ c_2\ \ldots\ c_{n-1}]$ sofra deslocamentos para a direita gerando as seguintes palavras-código

$$\begin{aligned}\mathbf{c^{(1)}} &= \begin{bmatrix}c_{n-1} & c_0 & c_1 & \ldots & c_{n-2}\end{bmatrix}\\ \mathbf{c^{(2)}} &= \begin{bmatrix}c_{n-2} & c_{n-1} & c_0 & \ldots & c_{n-3}\end{bmatrix}\\ \vdots &\qquad\qquad \vdots\\ \mathbf{c^{(i)}} &= \begin{bmatrix}c_{n-i} & c_{n-i+1} & c_{n-i+2} & \ldots & c_{n-i-1}\end{bmatrix}\end{aligned}$$

Podemos tratar esses códigos por polinômios módulo $D^n - 1$[2], onde cada coeficiente representa um bit e D representa um deslocamento. Então podemos escrever as palavras-código da seguinte maneira

$$c(D) = c_0 + c_1 D + c_2 D^2 + \ldots + c_{n-1} D^{n-1}$$

Se $c(D)$ tem grau $n-1$, então podemos representar seus deslocamentos pela multiplicação por D, como podemos ver a seguir.

$$\begin{aligned} c(D) &= c_0 + c_1 D + c_2 D^2 + \ldots + c_{n-1} D^{n-1} \\ Dc(D) &= c_0 D + c_1 D^2 + c_2 D^3 + \ldots + c_{n-1} D^n \\ &= c_{n-1} + c_0 D + c_1 D^2 + \ldots + c_n D^{n-1} \end{aligned}$$

Todo polinômio palavra-código de grau $n-1$ de um código linear $C(n,k)$ possui um polinômio de grau mínimo $(n-k)$ como um de seus fatores. A esse polinômio damos o nome de *polinômio gerador* e ele é simbolizado por $g(D)$. Este polinômio é equivalente à matriz geradora $\mathbf{G}$.

Para codificar a seqüência de símbolos $m(D)$ de grau k, podemos multiplicá-la pelo polinômio gerador $g(D)$ de grau $n-k$, gerando um polinômio palavra-código $c(D) = m(D)g(D)$ de grau n. Porém esta é a maneira não sistemática de fazer a codificação. Para encontrarmos uma palavra-código na maneira sistemática precisamos fazer mais que uma multiplicação de $m(D)$ por $g(D)$. Primeiro precisamos multiplicar os símbolos que desejamos codificar por D^{n-k} e depois dividimos o resultado por $g(D)$ obtendo

$$\frac{D^{n-k} m(D)}{g(D)} = a(D) + \frac{b(D)}{g(D)} \tag{2.9}$$

onde $a(D)$ é o quociente e $b(D)$ é o resto da divisão. Por essa razão, $b(D)$ poderá ter no máximo grau $n-k-1$ e será usado como a seqüência de bits de paridade. Reescrevendo a equação (2.9) temos[3]

[2]Como os códigos são tratados como polinômios módulo $(D^n - 1)$, então observar que, sendo $a(D)$ um polinômio qualquer, $a(D)D^{n+r} = a(D)D^r(D^n - 1) + a(D)D^r$. Portanto $a(D)D^{n+r} = a(D)D^r \ mod(D^n - 1)$ [5]

[3]Lembrar que estamos trabalhando no corpo GF(2), portanto a soma é equivalente à subtração.

$$b(D) + D^{n-k}m(D) = a(D)g(D)$$

logo $c(D) = b(D) + D^{n-k}m(D)$, que é uma palavra-código válida.

Também podemos fazer este cálculo através de operações matriciais, como vimos anteriormente. Para os códigos cíclicos a matriz geradora $\mathbf{G}$ possui uma forma característica como podemos ver na equação (2.10).

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & & & & \mathbf{0} \\ & g_0 & g_1 & \cdots & g_{n-k} & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & g_0 & g_1 & \cdots & g_{n-k} & \\ \mathbf{0} & & & & g_0 & g_1 & \cdots & g_{n-k} \end{bmatrix} \tag{2.10}$$

Porém, nesta forma, teremos um código não-sistemático, ao contrário do que vimos na equação (2.5). Mas, se quisermos, podemos construir um código sistemático com $\mathbf{G} = [\mathbf{P},\ \mathbf{I_k}]$.

Da mesma forma que o código cíclico tem um polinômio gerador equivalente à matriz $\mathbf{G}$, ele tem também um polinômio de cheque de paridade $h(D)$ que é equivalente à matriz $\mathbf{H}$.

$$h(D)g(D)\ (mod\ D^n - 1) = 0$$

Assim como a matriz geradora tem uma forma peculiar, a matriz de cheque paridade apresenta características parecidas

$$\mathbf{H} = \begin{bmatrix} h_k & \ldots & h_1 & h_0 & & & & \mathbf{0} \\ & h_k & \ldots & h_1 & h_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & h_k & \ldots & h_1 & h_0 & \\ \mathbf{0} & & & & h_k & \ldots & h_1 & h_0 \end{bmatrix} \tag{2.11}$$

Exemplo 3 Vamos considerar que temos um código cíclico C cujo polinômio gerador é $g(D) = 1 + D^2 + D^3$. Podemos representar este polinômio em forma de um vetor, então teríamos

$$\mathbf{g} = [1\ 0\ 1\ 1]$$

Construindo a matriz geradora na forma mostrada pela equação (2.10), temos:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Mas também podemos construir $\mathbf{G}$ na forma sistemática. Para isso temos que fazer uma divisão entre cada uma das linhas de $\mathbf{I}_k$ pelo polinômio gerador, gerando os bits que formam a matriz $\mathbf{P}$. Então concatenamos essas matrizes na forma $[\mathbf{P}, \mathbf{I}_k]$ obtendo

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Seguindo a equação (2.11), temos que a matriz de cheque de paridade associada ao código C é

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Da mesma forma como fizemos para matriz geradora, também podemos construir $\mathbf{H}$ na forma sistemática, lembrando de substituir o polinômio gerador pelo polinômio de cheque de paridade e assim obter

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Códigos cíclicos são excelentes para detectar erros e existem duas razões pelas quais estes códigos costumam ser usados para este fim: eles podem ser projetados para detectar

muitas combinações de erros prováveis e a implementação tanto de circuitos de codificação como de detecção de erros é prática, segundo Haykin em [6]. Ainda dentro do grupo de códigos cíclicos há diversos subgrupos de códigos, cada um com suas propriedades particulares. Um destes subgrupos é formado por códigos que sofreram uma modificação chamada *encurtamento* (do inglês *shortening*) que consiste em retirar uma das coordenadas da mensagem, ou seja, um código (n,k) passa a ser um código $(n-1,k-1)$. Através dessa modificação obtemos os códigos CRC, que são códigos cíclicos muito usados para detecção de erros, principalmente erros em rajada.

2.3 Codificação Convolucional

Outro tipo de codificação de canal utilizada no sistema GSM é a codificação convolucional. Este nome se deve ao fato da palavra-código ser gerada a partir de convoluções feitas no corpo GF(2). Os códigos convolucionais possuem uma vantagem em relação aos códigos vistos na Seção 2.2: na codificação cíclica precisamos esperar a recepção de um bloco de k símbolos para então fazer a codificação, enquanto que na codificação convolucional esse processo é feito serialmente; podemos iniciar a codificação a partir da recepção do primeiro símbolo.

Podemos ver o codificador convolucional como uma máquina de estados finitos que é composta por M memórias (registrador de deslocamento de M elementos), n somadores módulo 2 e um multiplexador que serializa as saídas de cada somador. Consideremos que a seqüência de símbolos binários que desejamos codificar tenha tamanho L, então a seqüência codificada terá $n(L+M)$ bits. A *taxa de código*, que é a razão entre os bits que entram no codificador e os bits da mensagem codificada, é dada por

$$r = \frac{L}{n(L+M)} \qquad \text{bits / símbolo}$$

Como geralmente $L \gg M$, a taxa de código pode ser simplificada para

$$r \simeq \frac{1}{n} \qquad \text{bits / símbolo} \tag{2.12}$$

Outro parâmetro importante dos codificadores convolucionais é o *comprimento de restrição*, que define o número máximo de bits na saída que um bit na entrada do codificador pode influenciar. Para calcular o comprimento de restrição, representado por K, devemos

somar 1 ao comprimento do maior número de registradores de deslocamento, então $K = 1 + M$. A seguir temos alguns exemplos de codificadores convolucionais.

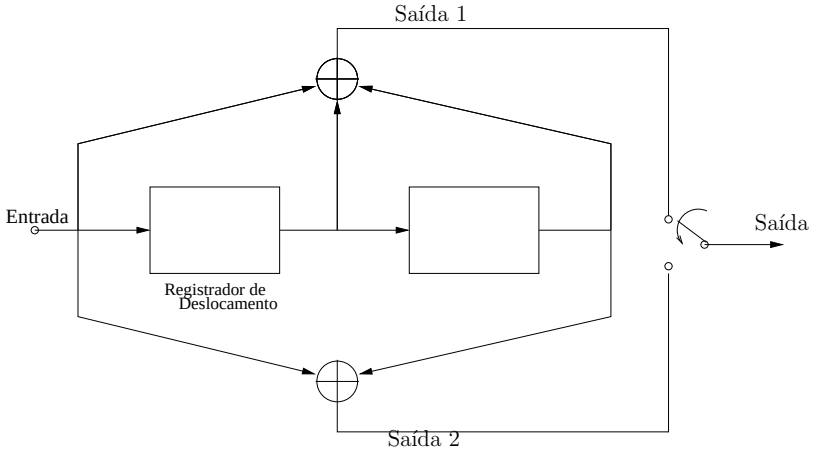

Figura 2.2: Codificador convolucional com $r = 1/2$ e $K = 3$.

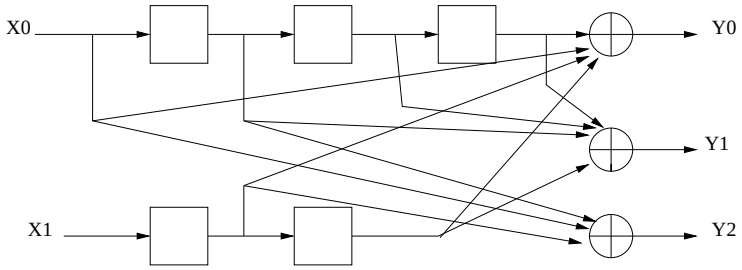

Figura 2.3: Codificador convolucional com $r = 2/3$ e $K = 4$.

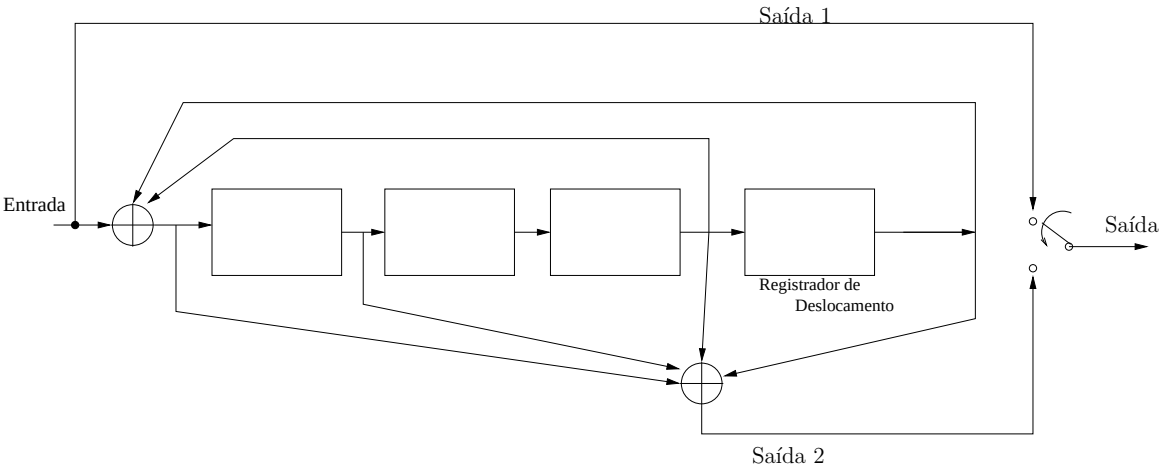

Figura 2.4: Codificador convolucional com $r = 1/2$ e $K = 5$.

Podemos observar que cada percurso do codificador convolucional que interliga a entrada à saída pode ser representado por seu comportamento no domínio do tempo em termos de suas respostas ao impulso. As Figuras 2.2 e 2.3 mostram codificadores compostos por filtros FIR e na Figura 2.4 temos um codificador composto por um filtro IIR. Usando o codificador da Figura 2.2 podemos representar seus filtros pelos seus coeficientes:

$$\mathbf{g}^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{g}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Os vetores formados pelos coeficientes dos filtros são chamados de *polinômios geradores*. Temos que os vetores $\mathbf{y}^{(0)}$ e $\mathbf{y}^{(1)}$ formam $\mathbf{y}$ que é a saída do codificador convolucional são obtidos através da convolução entre a mensagem original $\mathbf{m}$ e os polinômios geradores. Matematicamente, temos:

$$\begin{aligned} y_i^{(0)} &= \sum_{l=0}^{M} g_l^{(0)} m_{i-l} = \mathbf{g}^{(0)} * \mathbf{m} \\ y_i^{(1)} &= \sum_{l=0}^{M} g_l^{(1)} m_{i-l} = \mathbf{g}^{(1)} * \mathbf{m} \end{aligned}$$

onde o símbolo $*$ representa uma convolução enquanto que os elementos $g_l^{(0)}$ e $g_l^{(1)}$, com $l = 1, 2, \ldots, M$, são os coeficientes dos polinômios geradores.

Assim como no caso dos codificadores lineares por blocos, também podemos representar a codificação convolucional por meio de operações matriciais. Então $\mathbf{y} = \mathbf{mG}$, onde $\mathbf{G}$ terá o seguinte formato:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0^0 & \mathbf{g}_0^1 & \mathbf{g}_1^0 & \mathbf{g}_1^1 & \cdots & \cdots & \mathbf{g}_m^0 & \mathbf{g}_m^1 & & & \mathbf{0} \\ & & \mathbf{g}_0^0 & \mathbf{g}_0^1 & \mathbf{g}_1^0 & \mathbf{g}_1^1 & \cdots & \cdots & \mathbf{g}_m^0 & \mathbf{g}_m^1 & \\ & & & & \mathbf{g}_0^0 & \mathbf{g}_0^1 & \mathbf{g}_1^0 & \mathbf{g}_1^1 & \cdots & \cdots & \\ \mathbf{0} & & & & \ddots & & \ddots & & \ddots & \ldots & \ddots \end{bmatrix} \tag{2.13}$$

Exemplo 4 Utilizando a Figura 2.2 e considerando que desejamos codificar uma mensagem $\mathbf{m}$ composta por 4 bits, a matriz G terá o seguinte formato

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Considerando que $\mathbf{m} = [1\ 0\ 1\ 0]$ então a mensagem codificada será

$$\mathbf{y} = [1\ 0\ 1\ 0] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = [1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0]$$

A convolução da mensagem na entrada do codificador pelos filtros que o representam, estes lineares e invariantes ao deslocamento, equivale a uma multiplicação se trabalharmos no domínio da freqüência. Neste caso, os polinômios geradores do código convolucional são definidos como:

$$g^{(0)}(D) = g_0^{(0)} + g_1^{(0)} D + g_2^{(0)} D^2 + \ldots + g_M^{(0)} D^M \tag{2.14}$$

$$g^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + g_2^{(1)} D^2 + \ldots + g_M^{(1)} D^M \tag{2.15}$$

A variável D^l, onde $l = 1, 2, \ldots, M$, pode ser interpretada como o atraso z^{-l} utilizado na representação de filtros digitais no domínio de freqüência. Definindo a seqüência de entrada como

$$m(D) = m_0 + m_1 D + m_2 D^2 + \ldots + m_L D^L \tag{2.16}$$

e utilizando as equações (2.14) e (2.16), podemos reescrever as saídas do codificador convolucional como

$$y^{(0)}(D) = g^{(0)}(D) \cdot m(D) \tag{2.17}$$

$$y^{(1)}(D) = g^{(1)}(D) \cdot m(D) \tag{2.18}$$

Exemplo 5 Usando ainda o codificador da Figura 2.2, podemos refazer o exemplo 4 na forma polinomial. Os polinômios geradores são

$$\begin{aligned} g^{(0)}(D) &= 1 + D + D^2 \\ g^{(1)}(D) &= 1 + D^2 \end{aligned}$$

Usando a mesma seqüência de entrada

$$m(D) = 1 + D^2$$

e fazendo as operações polinomiais descritas pelas equações (2.18) e (2.18), temos:

$$\begin{aligned} y^{(0)} &= (1 + D + D^2) \cdot (1 + D^2) \\ &= 1 + D + D^3 + D^4 \\ &= [1\ 1\ 0\ 1\ 1\ 0] \\ y^{(1)} &= (1 + D^2) \cdot (1 + D^2) \\ &= 1 + D^4 \\ &= [1\ 0\ 0\ 0\ 1\ 0] \end{aligned}$$

Combinando as saídas para formar y, temos o mesmo resultado encontrado no Exemplo 4.

Para apresentar as propriedades estruturais de um codificador convolucional na forma gráfica, costuma-se usar um desses três tipos de diagrama: árvore de código, treliça ou digrama de estados. Para construir esses diagramas utilizamos os estados do codificador, que são os $(K - 1)$ bits de mensagem armazenados no registrador de deslocamento.

A árvore de código considera que o codificador é iniciado com estado zero, ou seja, com todos as suas memórias contendo zero. Cada ramo da árvore representa um símbolo de entrada com o correspondente símbolo de saída escrito no final do ramo. Para identificar qual valor do bit de entrada, utilizamos o ramo superior para entrada 0 e o ramo inferior

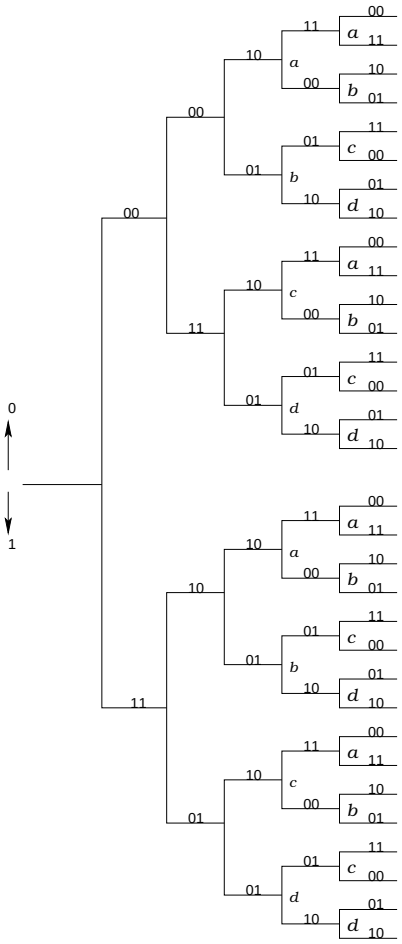

Figura 2.5: Árvore de código para o codificador convolucional da Figura 2.2.

para a entrada 1. Para saber a mensagem que sairá do codificador convolucional, seguimos os ramos de acordo com os bits de entrada fazendo um percurso da esquerda para a direita.

Observando a árvore de código da Figura 2.5 percebemos que a partir de uma certa etapa ela se torna repetitiva. Outro porém é o crescimento do número de ramificações da árvore, a cada entrada de um novo símbolo da mensagem temos 2 vezes o número de ramificações que tínhamos na etapa anterior. Portanto se tivermos uma mensagem longa para decodificar, a árvore de código não será uma opção prática para visualizar o codificador convolucional. A representação gráfica do código por meio de *treliça* não apresenta este problema, por meio dela podemos enxergar facilmente que o codificador convolucional está associado a uma máquina de estados **finitos**.

Na Figura 2.6 podemos observar uma treliça. O diagrama recebe este nome porque a treliça tem uma estrutura em forma de árvore com ramos entrelaçados. Por convenção, utilizamos uma linha cheia para representar o bit de entrada 0 e uma linha tracejada para

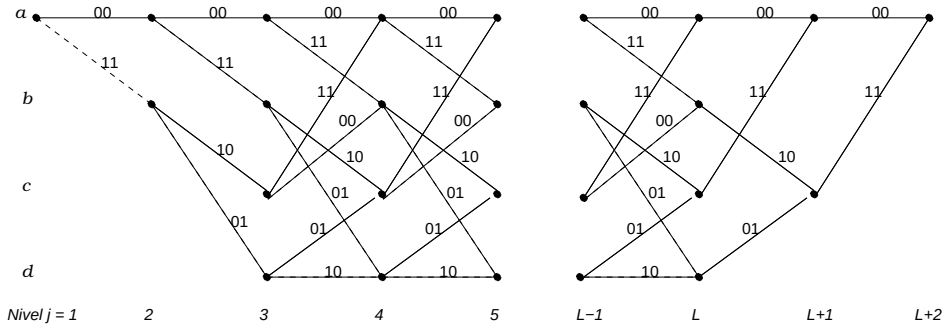

Figura 2.6: Treliça para o codificador convolucional da Figura 2.2.

o bit de entrada 1 e, escrito em cima de cada ramo, temos o símbolo que o bit provoca na saída. Assim como na árvore, cada seqüência de entrada corresponde a um percurso ao longo da treliça. Na direita da Figura 2.6 temos os possíveis estados do codificador dispostos na vertical e os nós pertencentes a mesma linha correspondem ao mesmo estado. Na parte inferior está representado o tempo j, também conhecido por nível ou profundidade, onde m_j é o bit de entrada. A treliça contém $(L + K)$ níveis, onde L é o tamanho da mensagem que queremos codificar e K é comprimento de restrição do código.

Estudando a treliça mais detalhadamente, podemos perceber que até o tempo $j = K$ o codificador não pode assumir qualquer um dos seus possíveis estados, apenas no momento seguinte ele poderá assumir qualquer um destes. A partir desse nível as ramificações são todas iguais, então vamos considerar uma parte da treliça correspondente às profundidades j e $j + 1$. Os nós à esquerda representam os estados atuais do codificador enquanto que os nós à direita representam os estados futuros. Podemos juntar estes nós, como mostra a Figura 2.7, gerando assim o *diagrama de estados* que também descreve totalmente a relação entre a entrada e a saída do codificador convolucional.

2.4 Algoritmo de Viterbi

Existem algumas maneiras de decodificar uma mensagem protegida por um código convolucional. Uma delas é o algoritmo de Viterbi que tem por princípio encontrar um percurso na árvore de códigos cuja seqüência codificada possua o menor número possível de símbolos diferentes da sequência recebida. Como vimos anteriormente, na árvore de códigos o número de nós aumenta a cada bit recebido da sequência a ser codificada enquanto que na treliça o número de nós permanece constante em 2^{K-1}, onde K é o comprimento de restrição.

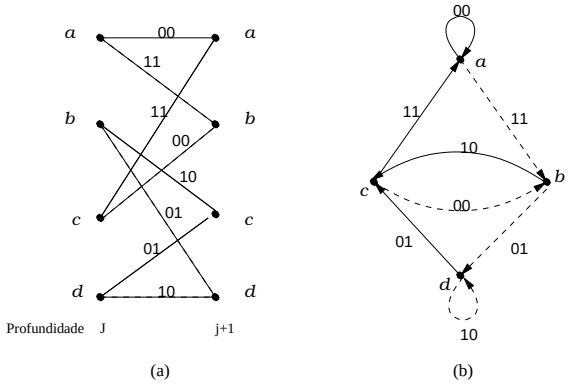

Figura 2.7: Diagrama de estados para o codificador convolucional da Figura 2.2.

Por essa razão a treliça é escolhida para descrever o codificador convolucional.

Usando a treliça da Figura 2.6 como base para o entendimento desse algoritmo de decodificação, percebemos que a partir do tempo $j = K - 1$, todos os níveis subseqüentes terão dois ramos entrando em cada nó (estado) e a partir desse ponto os dois percursos farão os mesmos movimentos até o final. Podemos decidir qual dos dois percursos queremos manter sem perder desempenho e é isso que faz o algoritmo de Viterbi. O algoritmo calcula a métrica ou a discrepância referente a todos os percursos na treliça até cada nó e utiliza esse valor para decidir quais seqüências devem ser guardadas. Para os dois percursos que entram no nó, o algoritmo calcula a métrica destes que é definida como a distância de Hamming entre a seqüência codificada representada por esse percurso e a seqüência recebida. O percurso que tiver menor métrica é mantido, este é chamado de percurso sobrevivente ou ativo, o outro, o de maior métrica, é descartado. Se chegar em um nó dois percursos com a mesma métrica, então o percurso sobrevivente será escolhido ao acaso.

Para realizar a decodificação, o algoritmo de Viterbi segue os seguintes passos:

- **Inicialização**

  1. Devemos começar pelo estado zero que está mais a esquerda da treliça. A partir deste estado devemos desenhar os percursos correspondentes a cada possível bit da seqüência de entrada.
  2. Comparamos a saída do decodificador para o bit de entrada correspondente ao percurso com a mensagem recebida e calculamos sua métrica, que é o número de bits diferentes entre as duas seqüências. Guardamos este valor, que aparece perto de cada nó.

3. Desenhamos os percursos correspondentes aos estados obtidos até o momento. Calculamos a métrica para os novos percursos desenhados e somamos a métrica obtida pelo percurso até o nível anterior.
4. Repetimos o passo 3 até a profundidade $j = K - 1$.

- **Passos Intermediários**

1. Desenhamos os percursos correspondentes a cada estado.
2. Calculamos a métrica para cada percurso e somamos a este valor a métrica acumulada pelos percursos sobreviventes.
3. Como em cada nó chegam dois percursos, comparamos as métricas deles e guardamos aquele que possuir menor métrica, este será o percurso sobrevivente. Se ambos tiverem o mesmo valor de métrica, escolhemos um ao acaso.
4. Repetimos os três passos acima até o algoritmo atingir o nó de finalização, ou seja, quando chegarmos no último símbolo da seqüência recebida.

- **Finalização**

1. A partir do nó de finalização, traçamos um caminho ao longo da treliça, percorrendo-a da direita para a esquerda, seguindo os percursos que possuem as menores métricas.
2. O percurso traçado no item acima nos dá a seqüência mais parecida com a mensagem codificada originalmente.

Exemplo 6 Vamos considerar que fizemos a codificação convolucional da mensagem $\mathbf{m}$, como demonstrado no Exemplo 4, e ao transmitir a seqüência $\mathbf{y}$ tivemos um erro no quarto bit, então recebemos o sinal $\mathbf{r} = [1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0]$. Executamos os passos do algoritmo de Viterbi como descritos anteriormente. As Figuras de 2.8 a 2.12 mostram o desenvolvimento.
Quando $j = 6$, chegamos ao último símbolo da seqüência recebida $\mathbf{r}$. Nesse momento vemos qual nó neste nível possui a menor métrica e o percurso que leva a ele é o mais parecido com a mensagem codificada, neste caso $\hat{\mathbf{x}} = [1\ 0\ 1\ 0]$. Os M últimos bits, neste caso $M = 2$, são descartados, pois eles são causados pelo codificador convolucional. É importante também notar algo interessante, o valor

da menor métrica possui o mesmo valor de bits corrompidos pelo codificador, que no caso deste exemplo é igual a 1.

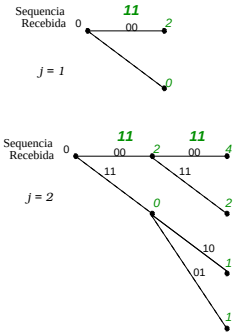

Figura 2.8: Algoritmo de Viterbi passos 1 e 2.

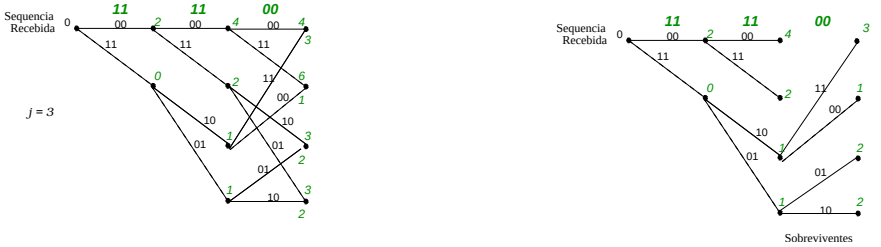

Figura 2.9: Algoritmo de Viterbi passo 3.

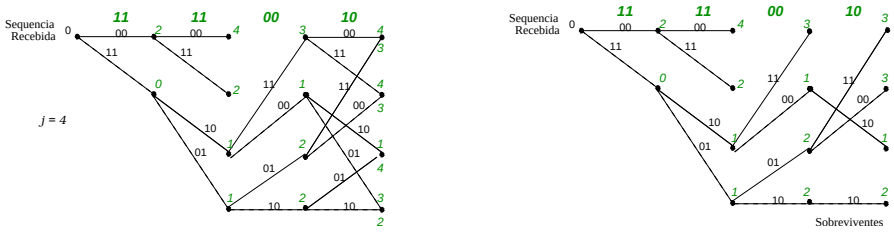

Figura 2.10: Algoritmo de Viterbi passo 4.

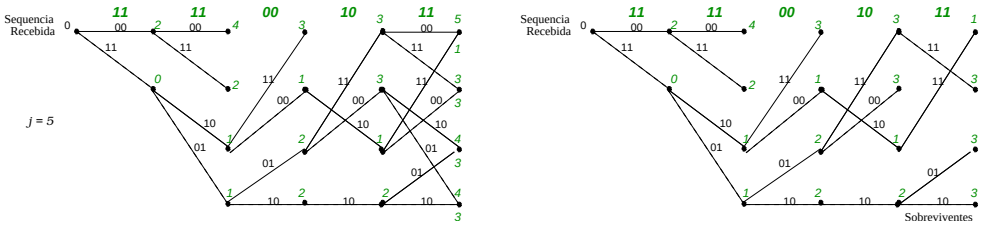

Figura 2.11: Algoritmo de Viterbi passo 5.

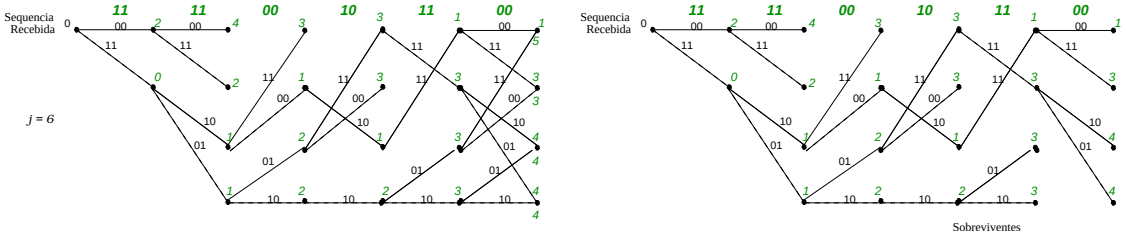

Figura 2.12: Algoritmo de Viterbi passo 6.

2.5 Modulação

A modulação utilizada pelo sistema GSM para a transmissão é o GMSK, que é um sinal de chaveamento de frequência mínima (**MSK** - *Minimum-Shift Keying*) filtrado por um filtro gaussiano. Ao utilizar tal filtro, obtemos algumas vantagens que não serão necessárias neste trabalho, uma vez que utilizaremos banda-base e, portanto, o sinal MSK.

O sinal MSK é um caso particular do sinal de chaveamento de frequência de fase contínua (**CPFSK** - *Continuous-Phase Frequency-Shift Keying*) e pode ser definido através da expressão [7]

$$s(t) = \sqrt{\frac{2E_b}{T_b}} \cos[2\pi f_c t + \Theta(t,\mathbf{I})] \tag{2.19}$$

onde E_b é a energia do sinal por bit transmitido, T_b a duração de bit, f_c a frequência da portadora, $\Theta(t,\mathbf{I})$ é a função que carrega a informação de fase do sinal MSK e $\mathbf{I}$ a sequência de dados a serem modulados pela CPFSK. Os dados da seqüência $\mathbf{I}$ estão mapeados numa representação NRZ e portanto cada um de seus elementos assume valores $+1$ ou -1.

Como o sinal está em banda-base, podemos utilizar apenas a envoltória complexa do sinal modulado $\overline{s}(t) = \exp\left(j\Theta(t,\mathbf{I})\right)$, onde

$$s(t) = Re\left\{\sqrt{\frac{2E_b}{T_b}}\overline{s}(t)\exp j2\pi f_c\right\}$$

Isto é suficiente para expressar o sinal original descrito por (2.19).

Representando o mesmo sinal na forma discreta temos

$$\overline{s}[n] = \exp j\Theta[n,\mathbf{I}] \tag{2.20}$$

onde [8]

$$\Theta[n,\mathbf{I}] = \frac{\pi}{2}\sum_{k=0}^{n-1} I_k \tag{2.21}$$

através dessas duas equações chegamos à seguinte expressão

$$\overline{s}[n] = j\overline{s}[n-1]I[n] \tag{2.22}$$

Essa equação mostra que se o sinal é real no instante n, no próximo instante ele será imaginário puro, mas se ele for um imaginário puro, no instante seguinte ele será real.

Capítulo 3

Codificador de Voz

3.1 Codificação de Voz

A transmissão em sistemas de telefonia em geral é bastante custosa, então é interessante e necessário fazer a compressão dos dados a serem enviados através do canal de forma que a qualidade do sinal recebido seja satisfatória. Esse trabalho é feito por codificadores de voz que, através de pesquisas ao longo dos anos, são aperfeiçoados a fim de conseguirem uma boa relação entre taxa de transmissão e qualidade.

Os codificadores de voz podem ser classificados em três tipos dependendo de como a informação extraída dos sinais de voz é enviada: codificadores por forma de onda, os quais transmitem o sinal de voz ou seus variantes; paramétricos, que enviam parâmetros extraídos da manipulação do sinal de voz; e híbridos, que misturam os dois tipos anteriores e tentam combinar o melhor de cada um.

Os codificadores por forma de onda fazem uso de propriedades de caráter temporal e/ou espectral do sinal de entrada. O objetivo destes é reconstruir a forma de onda do sinal original e o fazem com baixo custo computacional. Estes codificadores apresentam uma boa qualidade em detrimento de uma taxa de bits elevada. Um dos codificadores por forma de onda mais conhecido é o PCM (*Pulse-Code Modulation*) utilizado em praticamente todos os sistemas de telefonia fixa do mundo.

Os codificadores paramétricos utilizam características da fonte geradora do sinal de voz a partir da modelagem do trato vocal. Essa modelagem é feita de forma matemática, considerando o aparelho fonológico humano como um filtro digital, onde o sinal de saída é a voz, e cujos parâmetros são transmitidos pelo codificador. Um dos codificadores paramétri-

cos mais conhecidos é o LPC (*Linear Predictive Coding*), que apresenta uma pequena taxa de transmissão. Porém essa taxa reduzida compromete a qualidade do sinal decodificado, degradando o sinal de áudio. Para intervalos de voz de 20 ms e uma taxa de amostragem de 8 kHz, os codificadores LPC transmitem 13 valores, enquanto que codificadores PCM transmitem 160 valores pra uma mesma amostra.

Os codificadores híbridos usam o melhor dos dois codificadores apresentados acima, eles extraem os parâmetros do sinal de voz assim como os codificadores paramétricos e, ao mesmo tempo, utilizam características temporais e espectrais dos sinais como os codificadores por forma de onda. A técnica mais usada para esse tipo de codificação é a CELP (*Code Excited Linear Prediction*), a qual faz uso de dicionários para determinar as excitações, conseguindo utilizar um número de excitações maior em relação ao LPC e assim melhorar significativamente a qualidade do sinal reconstruído. Este tipo de codificador consegue uma taxa de codificação reduzida, comparada a dos codificadores por forma de onda, com qualidade superior a dos codificadores paramétricos. Na Tabela 3.1 podemos ver os três tipos de codificadores, as respectivas taxas de codificação com as quais trabalham e a qualidade do sinal codificado que produzem.

Tipo de Codificador	**Taxa de Codificação**	**Qualidade**
Forma de Onda	32 a 64 Kbits/s	Boa a Excelente
Paramétrico	1 a 4 Kbits/s	Regular a Ruim
Híbrido	3 a 12 Kbits/s	Boa a Excelente

Tabela 3.1: Comportamento dos codificadores com relação à taxa de codificação e qualidade.

O codificador de voz padronizado pelo 3GPP para utilização em redes de telefonia móvel é o AMR (*Adaptive Multi-Rate*), um codificador da família CELP. Este codificador pode operar em diversas taxas de transmissão, que variam entre 4,75 e 12,2 Kbits/s, dependendo das condições do canal de rádio por onde o sinal será transmitido. Mais detalhes deste codificador serão apresentados na Seção 4.3.

3.2 Técnicas de Criptofonia

A criptofonia é o estudo de princípios e técnicas pelas quais um sinal de áudio, mais especificamente um sinal de voz, é modificado segundo uma regra de forma que não seja

possível compreender o que foi dito se escutarmos o sinal cifrado. Um importante elemento dessa técnica é a chave de criptofonia e apenas aquelas pessoas que possuírem a chave poderão desfazer o processo de criptofonia e compreender a mensagem que o sinal original possui.

Os sistemas de criptofonia, de um modo geral, podem ser classificados em dois grandes grupos: Cifradores Analógicos e Cifradores Digitais. Os cifradores analógicos, também conhecidos por misturadores ou *scramblers*, realizam um processo de criptofonia no sinal de áudio resultando em um sinal cifrado analógico, ainda que todo o processamento seja digital. Esses cifradores apresentam níveis de segurança que variam de casual a tático, numa escala onde casual é o nível de segurança mínimo, tático é o nível intermediário e estratégico é o nível de segurança máxima.

Os cifradores digitais enviam parâmetros adquiridos através da análise do sinal realizado durante o processo de cifragem. Este tipo de criptofonia apresenta um nível de segurança que varia entre o tático e o estratégico. Eles podem ser divididos em dois subgrupos:

Categoria I: Informação codificada (digital) e transmissão não codificada (analógica);

Categoria II: Informação e transmissão codificadas (digital).

Segundo Andrade em [3], independente do tipo de sistema de criptofonia usado, é necessário que alguns requisitos sejam atendidos:

- Largura de banda do sinal cifrado compatível com o canal de transmissão utilizado;
- O sinal cifrado (voz) deve ser ininteligível ao ouvido humano, ou seja, ter baixa inteligibilidade residual[1];
- A voz decifrada deve apresentar boa inteligibilidade e preservar as características (timbre e altura) da voz do locutor;
- Pouco atraso nos processos de cifragem e decifragem do sinal;
- Resistência à criptoanálise adequada ao nível de segurança alcançado;
- Custo de implementação aceitável e compatível com o nível de segurança pretendido.

[1]A inteligibilidade residual expressa o quanto o sinal cifrado é parecido com o sinal original. Ela possui natureza subjetiva, entretanto há alguns métodos objetivos que podem ser usados para sua medida indireta, como os apresentados na Seção 3.3

Apesar da criptofonia digital apresentar um grau de segurança superior, os *scramblers* são os mais indicados a serem utilizados na telefonia móvel, pois para implementá-los não é necessário mudanças no hardware, que geralmente são complicadas e caras, e preservam a banda do sinal original.

Os cifradores analógicos apresentados neste trabalho usam a técnica de Criptofonia por Segmentação da Informação (CSI). Sistemas que empregam essa técnica manipulam elementos do sinal de áudio, como amplitude, freqüência, espectro, entre outros, tornando o sinal codificado inteligível ao ouvinte. Esta técnica pode ser empregada tanto no domínio do tempo como no domínio da freqüência.

A técnica CSI no domínio do tempo (CSI-T) realiza trocas nas posições de segmentos de amostras temporais que compõem o sinal. Em geral, sua implementação consiste em dividir o sinal em N segmentos de duração igual a 20 ms e permutar estes segmentos alterando a ordem original do áudio.

A CSI-T possui algumas desvantagens que a torna inadequada para um sistema de telefonia, como atrasos inevitáveis introduzidos por essa técnica. O cifrador necessita de N segmentos do sinal para fazer a permutação atrasando o sistema em pelo menos $N \times 20$ ms.

As primeiras técnicas de Criptofonia por Segmentação da Informação no domínio da frequencia (CSI-F) utilizavam inversões do sinal original neste domínio, e consequentemente no espectro, a fim de tornar o sinal ininteligível a qualquer usuário que não possuísse um receptor capaz de fazer o processo contrário. Devido a facilidade para reconstruir o sinal usando a CSI-F, esses tipos de cifradores deixaram de ser utilizados. Com o avanço da eletrônica e o surgimento de circuitos capazes de realizar funções complexas, o projeto de sistemas de criptofonia CSI-F usando bancos de filtros e transformadas ortogonais se tornou possível.

3.2.1 CSI-F Baseada em Bancos de Filtros

A Figura 3.1 mostra um banco de filtros com N subfaixas que permite abranger todo o sinal de voz na entrada do banco de filtros. O sinal filtrado pelo banco de filtros de análise $H_k(z)$ é decimado por um fator N e as subbandas são então permutadas utilizando-se uma matriz de permutação $\mathbf{P}$.

Será considerado o banco de filtro usando DFT (*Discrete Fourier Transform*) uniforme e que o sinal de voz está dividido em blocos, sendo representado pelo vetor $\mathbf{x}_i$ o i-ésimo bloco.

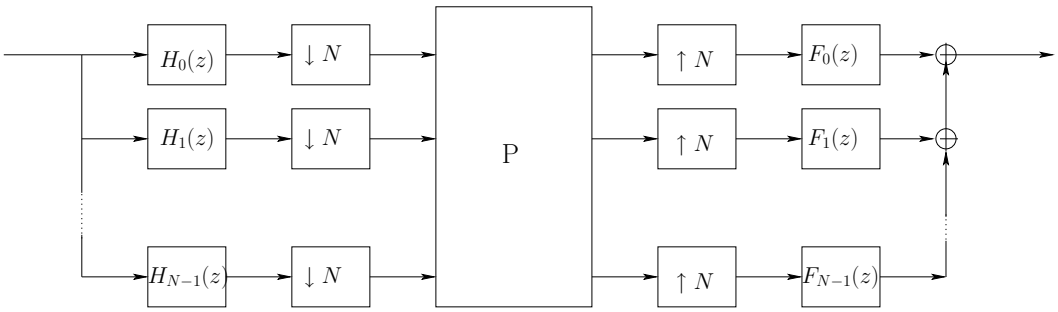

Figura 3.1: CSI-F baseado em banco de filtros.

A amostra $\mathbf{V}_k^i(z)$, que é o sinal após a passagem pelo banco de filtros de análise, representa a k-ésima subfaixa do sinal de entrada no domínio z pertencente ao i-ésimo bloco e pode ser calculada como:

$$\mathbf{V}_k^i(z) = \mathbf{H}_k(z)\mathbf{X}_k^i(z) \qquad \forall \quad k = 0,1,\ldots,N-1 \tag{3.1}$$

onde os vetores $\mathbf{V}_k^i$ são arrumados na forma de uma matriz de tamanho $N \times M$, onde $M-1$ é a ordem do filtro de análise $\mathbf{H}_k(z)$. Então temos esta matriz com as amostras da seguinte maneira:

$$\mathbf{V}^i = \left[\mathbf{V}_0^i \ \mathbf{V}_1^i \ \ldots \ \mathbf{V}_{N-1}^i\right]^T \tag{3.2}$$

Fazendo a multiplicação da matriz $\mathbf{V}^i$ pela matriz de permutação $\mathbf{P}$, obtemos a matrix $\mathbf{U}^i$ cujas linhas correspondem a cada uma das subbandas do i-ésimo bloco permutadas pela matrix $\mathbf{P}$. Aplicando o banco de filtro de síntese nos vetores linha da matriz $\mathbf{U}^i$ resultando o i-ésimo bloco cifrado $\mathbf{Y}^i$, ou seja,

$$\mathbf{U}^i = \mathbf{P}\mathbf{V}^i \tag{3.3}$$

$$\mathbf{U}^i = \left[\mathbf{U}_0^i \ \mathbf{U}_1^i \ \ldots \ \mathbf{U}_{N-1}^i\right]^T \tag{3.4}$$

$$\mathbf{Y}_k^i(z) = \mathbf{F}_k(z)\mathbf{U}_k^i \quad , \quad k = 0,1,\ldots,N-1 \tag{3.5}$$

$$\mathbf{Y}^i = \left[\mathbf{Y}_0^i \ \mathbf{Y}_1^i \ \ldots \ \mathbf{Y}_{N-1}^i\right]^T \tag{3.6}$$

Aplicando a inversa da tranformada $\mathcal{Z}$ a cada linha do bloco $\mathbf{Y}_i$ após a interpolação e fazendo o somatório bit a bit de cada linha, obtemos o sinal cifrado como mostrado a seguir:

$$\mathbf{y}_i = \sum_{k=0}^{N-1} \mathcal{Z}^{-1}\{Y_k^i(z)\} \tag{3.7}$$

Para decifrar o sinal criptofonado, utilizamos o mesmo processo de cifragem, porém substituindo a matriz de permutação $\mathbf{P}$ por sua inversa $\mathbf{P}^{-1}$.

3.2.2 CSI-F Baseadas em Transformadas Ortogonais

Cifradores baseados em transformadas ortogonais, também denominados de misturadores no domínio da transformada, realizam o processo de transformação do sinal através da multiplicação de cada bloco do sinal pela matriz de transformação. Considerando o vetor $\mathbf{x}_i$, com NM elementos, o representante do i-ésimo bloco do sinal e $\mathbf{T}$ a matriz de transformação ortogonal. A multiplicação destes gera o vetor $\mathbf{v}_i$ no domínio da transformada, ou seja, no domínio da frequência:

$$\mathbf{v}_i = \mathbf{T}\mathbf{x}_i \tag{3.8}$$

Dividimos o vetor $\mathbf{v}_i$ em N segmentos com os quais podemos construir uma matriz $N \times M$, como na equação (3.2), onde cada segmento é um vetor coluna de $\mathbf{V}^i$ no domínio da transformada. Como mostra a equação (3.3, multiplicamos a matriz $\mathbf{V}_i$ por $\mathbf{P}$ para fazer a permutação dos N segmentos.

As linhas da matriz $\mathbf{U}_i$ são concatenadas originando o vetor coluna $\mathbf{u}_i$, o qual aplicamos a transformada inversa $\mathbf{T}^{-1}$ obtendo o sinal cifrado $\mathbf{y}_i$

$$\mathbf{y}_i = \mathbf{T}^{-1}\mathbf{u}_i \tag{3.9}$$

Para recuperar o sinal original, devemos fazer o mesmo processo de cifragem, porém substituindo a matriz $\mathbf{T}$ por sua inversa $\mathbf{T}^{-1}$. Matematicamente temos:

$$\mathbf{x}_i = \mathbf{\Phi}^{-1}\mathbf{y}_i \tag{3.10}$$

$$\mathbf{\Phi}^{-1} = \mathbf{T}^{-1}\mathbf{P}^{-1}\mathbf{T} \tag{3.11}$$

Além da baixa complexidade computacional destas transformadas ortogonais e unitárias, durante o processo de recuperação do sinal original elas não amplificam a energia

do ruído adicionado pelo canal. Por essa razão essas transformadas são escolhidas para o processo de cifragem do sinal.

Considere que o sinal $\mathbf{y}_i$ foi enviado por um canal que adicionou um ruído branco η a ele. O sinal no receptor será

$$\widetilde{\mathbf{y}}_i = \mathbf{y}_i + \eta(t) \tag{3.12}$$

E o sinal decifrado pode ser expresso como

$$\widetilde{\mathbf{x}}_i = \phi^{-1}\{\mathbf{y}_i + \eta(t)\} \tag{3.13}$$

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_i + \phi^{-1}\eta(t) \tag{3.14}$$

A transformação ϕ^{-1} é ortogonal e possui norma unitária, portanto o ruído no receptor não é afetado pelo processo de criptofonia. Então podemos concluir que a matriz de transformação ortogonal não afeta a energia do ruído durante o processo de recuperação do sinal, como mostra a equação (3.15)

$$\|\phi^{-1}\eta(t)\| = \|\eta(t)\| \tag{3.15}$$

3.3 Chaves para Criptofonia

A escolha da matriz de permutação, também conhecida por chave para criptofonia, é essencial para um bom funcionamento do processo de criptoanálise do sinal de voz. Esta chave é a responsável pela inteligibilidade residual do sinal cifrado e pelo nível de segurança que o sinal transmitido terá.

Existem $N_p = N!$ possíveis chaves para criptofonia, onde N é o números de subbandas as quais o sinal será dividido. Dentre as N_p chaves, apenas um pequeno número possui as características necessárias para produzir sinais com inteligibilidade residual e resistência a criptoanálise. Devemos fazer a escolha das chaves obedecendo dois critérios:

Critério I: Todas as chaves $\mathbf{P}_i$ pertencentes ao subconjunto $\mathbf{S}$, que possui todas as chaves para criptofonia, devem produzir baixa inteligibilidade residual; e

Critério II: Para cada chave $\mathbf{P}_i \in \mathbf{S}$, deverá existir somente uma chave $\mathbf{P}_i^{-1} \in \mathbf{U}$, subconjunto que possui todas as chaves no receptor, capaz de recuperar o sinal cifrado.

Se outra chave for empregada no processo de decifragem, o sinal produzido deverá ser ininteligível.

A inteligibilidade residual é uma grandeza subjetiva e, por isso, difícil de medir, mas podemos definir uma "distância" $D(\mathbf{P}_i, \mathbf{I})$ como medida indireta para inteligibilidade residual, onde $\mathbf{I}$ é a matriz identidade da mesma ordem de $\mathbf{P}_i$. O Critério I pode ser escrito como

$$D(\mathbf{P}_i,\mathbf{I}) > \mathcal{L}_I \quad , \quad \forall \ \mathbf{P}_i \ \in \ \mathbf{U} \tag{3.16}$$

onde o limiar $\mathcal{L}_I$ deve ser estimado de maneira a garantir uma baixa inteligibilidade residual.

O critério II também pode ser escrito matematicamente da seguinte forma

$$D(\mathbf{P}_i, \mathbf{P}_j^{-1}) > \mathcal{L}_{II} \quad , \quad \forall \ \mathbf{P}_i, \mathbf{P}_j \ \in \ \mathbf{S} \quad \text{e} \quad i \neq j \tag{3.17}$$

Considerando que $\mathbf{I}$ representa a posição dos segmentos no sinal original, utilizando a *Distância de Hamming* para o cálculo de $D(\mathbf{P}_i,\mathbf{I})$, obtemos como resultado o número de elementos que foram deslocados da sua posição original pela permutação.

Uma nova abordagem, proposta em [3], considera a permutação como sendo uma rotação de eixos dos espaços vetorias $\mathbb{R}^N \rightarrow \mathbb{R}^N$, onde N é o tamanho da chave. Podemos então usar o ângulo entre $\mathbf{P}_i$ e $\mathbf{I}$ para calcular a "distância" $D(\mathbf{P}_i,\mathbf{I})$.

Seja $\mathbf{V}_N = [1 \ 2 \ \ldots \ N]_{1\times N}$, o ângulo entre $\mathbf{P}_i$ e $\mathbf{I}$ é definido como

$$D(\mathbf{P}_i,\mathbf{I}) = \Phi_I = \arccos\left\{\frac{(\mathbf{V}_N\mathbf{P}_i)\cdot(\mathbf{V}_N\mathbf{I})^T}{\|\mathbf{V}_N\|^2}\right\} \quad , \quad \forall \ \mathbf{P}_i \ \in \ \mathbf{U} \tag{3.18}$$

$$D(\mathbf{P}_i, \mathbf{P}_j^{-1}) = \Phi_{II} = \arccos\left\{\frac{(\mathbf{V}_N\mathbf{P}_i)\cdot(\mathbf{V}_N\mathbf{P}_j^{-1})^T}{\|\mathbf{V}_N\|^2}\right\} \quad , \quad \forall \ \mathbf{P}_i, \mathbf{P}_j \ \in \ \mathbf{S} \text{ e } i \neq j \tag{3.19}$$

Através da equação (3.18), calculamos indiretamente o valor de rotação provocada pela matriz $\mathbf{P}_i$. Obtemos então o valor do ângulo Φ_I entre o vetor $\mathbf{V}_N$ e o vetor permutado $(\mathbf{V}_N\mathbf{P}_i)$.

Para cada tamanho de chave N, existe um valor máximo Φ_I^{max} que pode ser obtido usando a matriz $\mathbf{P}^{90^\circ}$, que é uma matriz com 1s na sua diagonal secundária e zero nas demais entradas. Substituindo $\mathbf{P}_i$ por $\mathbf{P}^{90^\circ}$ e resolvendo a equação (3.18), temos

$$\Phi_I^{max}(N) = \arccos\left\{\frac{N+2}{2N+1}\right\} \tag{3.20}$$

Escolhendo um limiar $\mathcal{L}_I$ suficientemente grande, garantimos que as chaves farão com que a maior parte os segmentos seja alocada na metade oposta do sinal considerando como referência sua posição original. Isso gera um sinal com baixa inteligibilidade residual. Segundo os resultados apresentados em [3], $\mathcal{L}_I = 0{,}85\Phi_I^{max}(N)$ representa um valor adequado.

A Tabela 3.2 mostra a quantidade de chaves que atendem ao critério I para o limite $\mathcal{L}_I = 0{,}85\Phi_I^{max}(N)$. Podemos observar que para uma quantidade de subfaixas N menor que 8 temos poucas possibilidades de chaves que produzem baixa inteligibilidade residual e portanto não deve ser utilizados em sistemas de criptofonia. Também temos que levar em consideração um limite superior para N devido a utilização do CODEC AMR. Se escolhermos um valor muito grande para N, a permutação desse elevado número de subfaixas pode comprometer a qualidade do sinal se levarmos em conta os processos de codificação e decodificação realizados pelo CODEC AMR.

N	**N° Total de Chaves**	$\mathbf{\Phi_I^{max}}$	$\mathcal{L}_\mathbf{I}$	**N° de Chaves para** $\Phi_I \geqslant \mathcal{L}_I$
4	24	48,19°	40,96°	5 (20,83%)
5	120	50,49°	42,91°	27 (22,50%)
6	720	52,02°	44,22°	128 (17,78%)
7	5.044	53,13°	45,16°	672 (13,33%)
8	40.320	53,97°	45,87°	4.900 (12,15%)
9	362.880	54,62°	46,43°	35.163 (9,69%)
10	3.628.800	55,15°	46,89°	301.704 (8,31%)

Tabela 3.2: Número de chaves que atendem ao Critério I ($4 \leqslant N \leqslant 10$).

Uma solução para elevar o nível de segurança do sistema, sem aumentar exageradamente o valor de N, é o uso de trocas periódicas das chaves. Para realizar este tipo de criptofonia, precisamos adotar um mecanismo de sincronismo preciso, que auxilie a troca das chaves correspondentes tanto no transmissor quanto no receptor.

Capítulo 4

Simulador

4.1 Descrição Geral

O simulador é composto por: fonte, que é o arquivo de áudio utilizado para simulação; cifrador, que pode ser ativado ou não pelo usuário; CODEC AMR; codificador de canal; bloco de transmissão, que engloba a modulação, o canal e a demodulação; decodificador de canal e decifrador; como podemos observar na Figura 4.1. Os blocos de criptofonia e codificação de canal foram desenvolvidos em MATLAB© e o CODEC AMR está implementado em C, seu código pode ser adquirido no site do 3GPP. Uma descrição de cada módulo do simulador implementado neste projeto será detalhado a seguir.

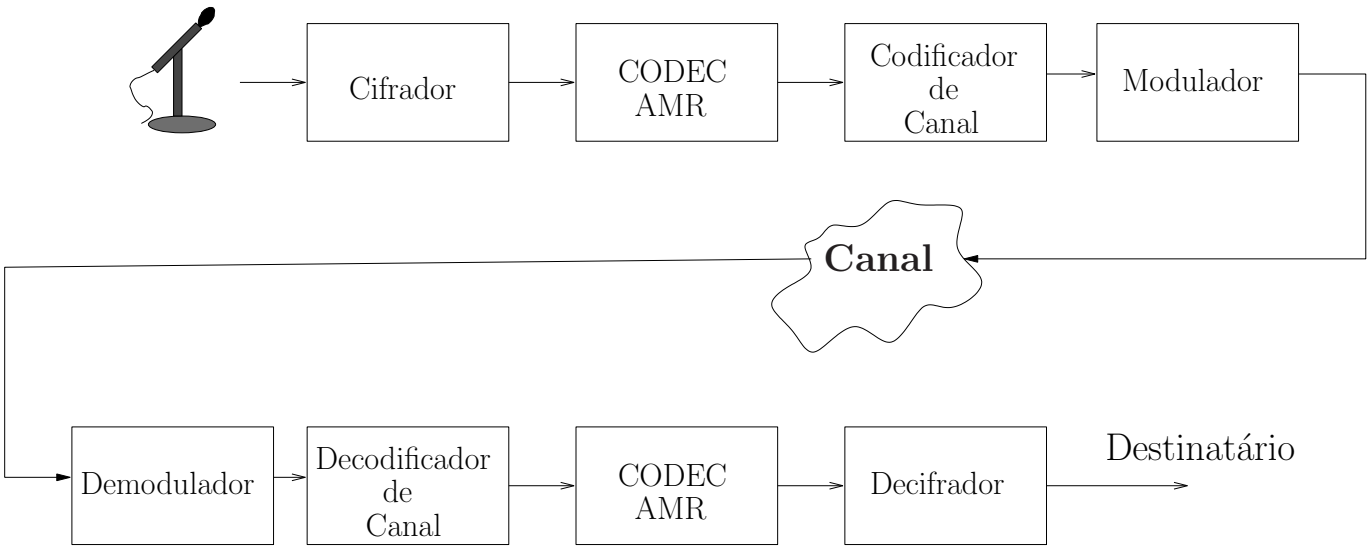

Figura 4.1: Arquitetura do simulador de codificação de canal com criptofonia.

4.2 Fonte e Bloco de Criptofonia

A fonte será um arquivo de áudio com a extensão *WAVE* amostrado em 8 kbits. O usuário tem a liberdade de escolha do arquivo que deseja passar pelo simulador.

O método de criptofonia utilizado no simulador foi o CSI-F baseado em transformadas ortogonais e para sua implementação foi usada a *Discrete Cosine Transform* (DCT).

Como saída do codificador teremos dois arquivos de audio, um com extensão *WAVE* e outro com extensão *RAW*, ambos conterão o arquivo de áudio criptofonado, sendo o último utilizado como entrada do CODEC AMR.

4.3 CODEC AMR

Utilizamos para a codificação de fonte o CODEC AMR, cujo código fonte pode ser encontrado no site da 3GPP em [11]. Sua execução é feita através de linha de comando e os arquivos de entrada e saída possuem extensão *RAW*.

O CODEC AMR recebe um arquivo de áudio e divide o sinal contido neste arquivo em blocos com 20 ms de duração e 160 amostras[1]. Cada bloco é processado pelo codificador de voz e que gera um *frame*, que será entregue ao bloco de codificação de canal, e cujo formato pode ser observado na Figura 4.2.

FRAME_TYPE	$B1$	$B2$	$\cdots$	$B244$	MODE_INFO	unused 1	$\cdots$	unused 4

Figura 4.2: Parâmetros de um *frame* do sinal de saída do CODEC AMR

Os parâmetros do *frame* gerado pelo CODEC AMR são os seguintes:

- **FRAME_TYPE**: Especifica se o *frame* transmitido possui um sinal de voz, um sinal de controle ou é um *frame* que não carrega informação útil;
- **B1, B2, ... , B244**: Esses bits carregam o sinal codificado;
- **MODE_INFO**: Informa o modo de transmissão que foi utilizado pelo CODEC. Podemos escolher 1 dentre 8 modos de transmissão que o AMR oferece, mas, neste projeto, estamos utilizando apenas o modo de transmissão com taxa igual a 12,2 kbps;

[1]Como o CODEC AMR trabalha com sinais de áudio amostrados em 8 kHz, cada bloco de duração de 20 ms possui $8000 \times 20 \times 10^{-3} = 160$ amostras.

- **unused1, ... , unused4**: São bits não usados, são preenchidos com zeros.

4.4 Bloco de Codificação de Canal

O bloco de codificação de canal, como podemos observar na Figura 4.3, é composto por: ordenação subjetiva; codificação cíclica; codificação convolucional; embaralhamento e formatador de *burst*.

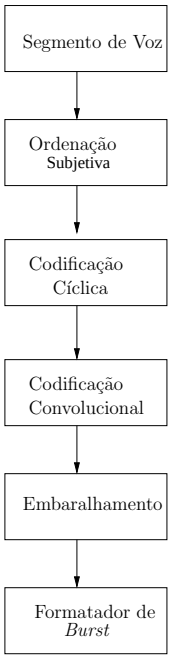

Figura 4.3: Bloco de codificação de canal

4.4.1 Ordenação Subjetiva

Esta função é responsável por ordenar os bits que carregam a informação do sinal de voz de acordo com sua importância subjetiva antes da codificação de canal. Cada modo de transmissão do CODEC AMR terá uma arrumação específica definida por tabelas que podem ser encontradas em [9]. Os bits são classificados em dois grupos dependendo do tratamento de codificação de canal que recebem. Os bits da classe 1b recebem apenas a codificação convolucional e os que pertencem a classe 1a recebem os dois tipos de codificação, como mostra a Figura 4.4.

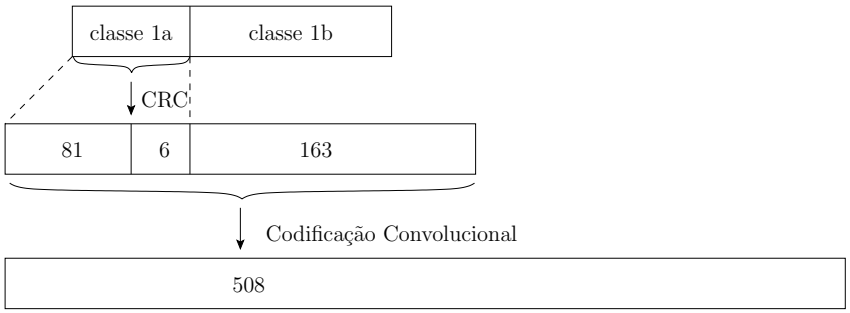

Figura 4.4: Classificação dos bits segundo a codificação de canal que recebem

4.4.2 Codificação Cíclica

Na codificação cíclica, usamos um código CRC de 6-bits para detecção de erros. O polinômio gerador usado pelo sistema GSM é $G(D) = D^6 + D^5 + D^4 + D^3 + D^2 + D + 1$. Ele gera os 6 bits de paridade que são alocados junto com os bits do sinal recebido pelo codificador cíclico obedecendo a seguinte regra:

$$\begin{aligned} u(k) &= d(k) && \text{para } k = 0,1,\ldots,\ 80 \\ u(k) &= p(k-81) && \text{para } k = 81,82,\ldots,\ 86 \\ u(k) &= d(k-6) && \text{para } k = 87,88,\ldots,\ 249 \end{aligned}$$

onde $u(k)$ são os bits na saída do codificador, $d(k)$ os bits na entrada e $p(k)$ os bits de paridade produzidos com a codificação cíclica.

4.4.3 Codificação Convolucional

O codificador convolucional usado pelo sistema GSM para taxa de transmissão de 12,2 kbps é representado por um filtro IIR como mostra a Figura 2.4 na Seção 2.3. Cada *frame* do sinal recebido pelo codificador possui 250 bits que serão transformados em 508 bits[2]. Ainda é aplicado sobre esse sinal uma perfuração, do inglês *puncturing*, que retira 60 bits de posições predeterminadas, resultando em um sinal na saída deste codificador contendo 448 bits.

[2]A taxa do codificador convolucional que estamos usando possui taxa $r = 1/2$, então teremos na saída $(250) + (M \times 2)$, onde M é a quantidade de elementos do maior registrador de deslocamento usado para construir o filtro. Consideramos além dos 500 bits obtidos através da convolução, os bits gerados pelo final do sinal que resta dentro do registrador.

$$G0/G0 \;=\; 1 \tag{4.1}$$

$$G1/G0 \;=\; 1 + D + D^3 + D^4 \;/\; 1 + D^3 + D^4 \tag{4.2}$$

A codificação convolucional e a perfuração foram implementadas usando a função `convenc` do MATLAB©. Esta função possibilita realizar ambos os processos de maneira eficiente do ponto de vista computacional.

4.4.4 Embaralhamento

O embaralhamento, do inglês *interleaving*, mistura os bits para prevenir possíveis erros em rajadas, ou seja, erros em bits consecutivos. Este procedimento permite com que, na recepção, o decodificador tenha um melhor desempenho. Este módulo recebe 456 bits (os 448 bits do codificador convolucional + 8 bits que servem para sinalização) e, além de embaralhar os bits, também os mapeia em 8 blocos com 57 bits cada um.

4.4.5 Formatador de *Burst*

O formatador de *burst* recebe os 456 bits organizados em 8 blocos de 57 bits cada e adiciona a cada bloco dois bits de *flags* (hl e hu) usados para o controle de sinalização de canal. A implementação dessa função é feita segundo a regra indicada pelas equações (4.3) e (4.4).

$$e(B,j) = i(B,j) \quad \text{e} \quad e(B,59+j) = i(B,57+j) \qquad \text{para } j = 0,1,\ldots,56 \tag{4.3}$$

$$e(B,57) = hl(B) \quad \text{e} \quad e(B,58) = hu(B) \tag{4.4}$$

onde e representa os bits do sinal de saída do formatador de *burst*, B representa o bloco, i os bits do sinal de saída do embaralhador e j o índice da posição de cada bit dentro do bloco.

4.5 Transmissão

Esta Seção descreve o canal e também a modulação do sinal que será transmitido através do canal. Para implementação deste bloco foram utilizadas funções do MATLAB©,

como será detalhado nas Seções 4.5.1 e 4.5.2.

4.5.1 Modulação

O sistema GSM usa a modulação GMSK, como foi citado na Seção 2.5, mas como no simulador trabalhamos com a banda-base, usamos a modulação MSK. Para implementá-la utilizamos a função `modulate`, que nos permite escolher alguns parâmetros como a quantidade de bits por amostras, optamos por fazer a modulação com 1 bit por símbolo.

4.5.2 Canal

Para as simulações escolhemos neste trabalho o canal com ruído branco gaussiano aditivo (AWGN - *Additive White Gaussian Noise*). Os códigos cíclico e convolucional são conhecidos na literatura por apresentar um desempenho melhor em canais sem memória, o que motivou a escolha do *awgn*. O MATLAB© possui a função `awgn` que introduz esse tipo de ruído ao sinal transmitido podendo-se escolher a razão sinal-ruído, também conhecida por SNR (*Signal-to-Noise-Rate*), para introduzir o ruído.

4.5.3 Demodulação

O demodulador recebe os símbolos transmitidos através do canal e decodifica segundo o mapeamento da modulação MSK. Para implementá-la usamos a função análoga ao `modulate`, o `demodulate` que usa os mesmos parâmetros escolhidos para a modulação.

4.6 Bloco de Decodificação de Canal

Nesta Seção será detalhado o bloco de decodificação de canal que realiza o processo inverso ao apresentado na Seção 4.4. Foram feitas funções que desfazem o processamento realizado pelos módulos de formatador de burst, embaralhamento e em seguida pelo algoritmo de Viterbi que faz a decodificação dos bits que receberam codificação convolucional. A implementação deste módulo de decodificação é feita utilizando a função `vitdec` do MATLAB©, o usuário deve dizer qual é o filtro utilizado para fazer a codificação convolucional; o tamanho da janela de decodificação, que deve ser no mínimo 5 vezes o tamanho de restrição do código; os bits perfurados e se a decodificação deve ser feita de maneira '*hard*' ou '*soft*', no

projeto estamos usando o modo '*hard*' que escolhe o valor do bit entre 0 e 1 sem levar em consideração modulação ou energia do bit.

Após este processamento, temos a decodificação do sinal que foi tratado com o código CRC, identificando se o bloco de sinal recebido possui ou não erros como foi mostrado na Seção 2.2, mas não faz a correção destes. Por fim, fazemos o processo inverso da Ordenação Subjetiva, colocando os bits do arquivo de fala na sua ordem original e preparando-os para passar pelo CODEC AMR novamente.

Capítulo 5

Simulações e Resultados

5.1 Simulação I

Esta simulação visa observar o efeito da codificação de canal no sistema de transmissão GSM e compará-lo à transmissão com o sistema que não faz o uso desta técnica. Para simular este contexto, geramos dados aleatórios para servir de fonte ao invés de usar arquivos de áudio pré-gravados. Passamos estes dados aleatórios pelo simulador e fazemos a comparação com os dados recebidos pelos dois métodos de transmissão.

Os dados utilizados nesta simulação foram:

- Número de *frames* por sinal: 500;
- Número de repetições do *loop* de Monte Carlo: 200.

5.1.1 Resultados

A Figura 5.1 apresenta as curvas de BER (*Bit Error Rate* - taxa de erro de bits) para a simulação sem e com codificação de canal. A transmissão que não faz uso dos códigos corretores de erro possui valores de BER maiores se comparado ao outro método de transmissão, isso indica que a codificação de canal está corrigindo os erros adicionados ao sinal pelo canal durante o processo de transmissão. O processo de codificação de canal ainda utiliza perfuração que remove 60 bits do sinal transmitido e substitui por zeros no receptor, mesmo assim o resultado encontrado utilizando a codificação de canal ainda é melhor do que quando não a utilizamos.

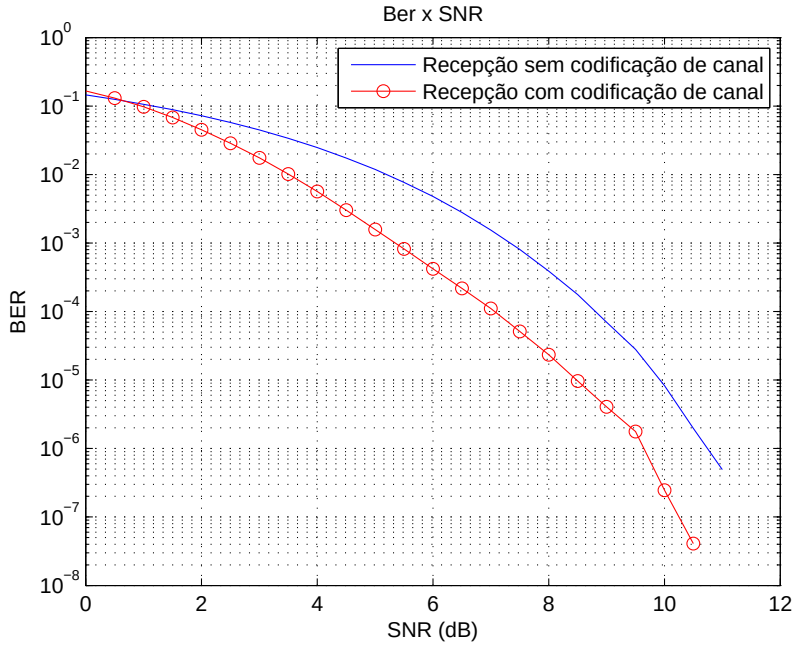

Figura 5.1: Curva de BER para transmissão do sistema GSM

Podemos observar que quando a SNR é igual a zero decibel, a curva de BER para uma transmissão sem codificação de canal possui valores menores que para uma transmissão com codificação de canal. Isto acontece pois a quantidade de erros inseridos pelo canal é significativamente grande que, ao tentar corrigi-los, os códigos corretores acabam inserindo mais erros. A codificação de canal utilizada neste simulador só começa a corrigir erros a partir de, aproximadamente, SNR = 0,7 dB.

5.2 Simulação II

Nesta simulação utilizamos arquivos de fala, contendo frases gravadas por diferentes locutores, como a fonte do sinal a ser transmitido. Passamos estes arquivos pelo CODEC AMR e transmitimos os dados de saída deste CODEC nos dois casos apresentados pela Simulação I, ou seja, fazendo as transmissões sem e com codificação de canal.

Com esta simulação desejamos investigar os efeitos do simulador sobre os dados transmitidos, no que diz respeito à qualidade do áudio. Para fazer a comparação entre os sinais de áudio original e recebido, utilizamos o PESQ (*Perceptual Evaluation of Speech Quality*). Ele é um algoritmo de avaliação subjetiva de qualidade que compara os sinais recebidos e dá um valor entre 0 e 5 dependendo do grau de semelhança entre esses sinais, se os sinais

forem muito parecidos teremos um valor próximo de 5, mas se tivermos sinais bem diferentes, obteremos um valor próximo de zero.

Os dados utilizados nesta simulação foram:

- Frequencia de amostragem: 8 kHz;
- Número de frases: 40;
- Número mínimo de frases por locutor: 5;
- Total de valores de SNR: 7 pontos igualmente espaçados entre 0 e 15 dB.

5.2.1 Resultados

Os gráficos representados nas Figuras de 5.2 a 5.15 mostram as distribuições dos arquivos de fala segundo o seu valor PESQ para uma dada SNR, utilizando transmissão sem e com codificação de canal.

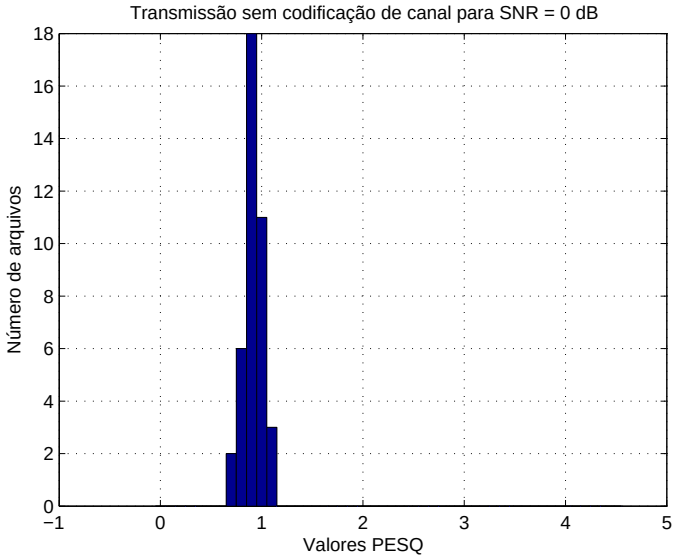

Figura 5.2: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 0 dB utilizando uma transmissão sem criptofonia.

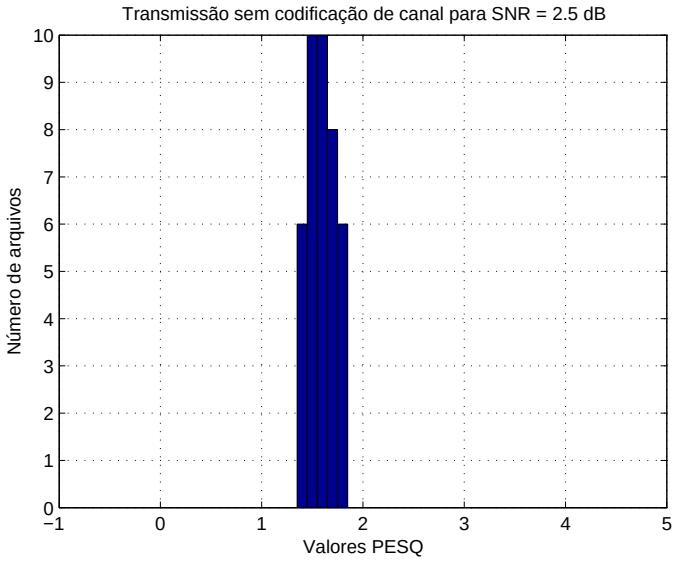

Figura 5.3: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 2.5 dB utilizando uma transmissão sem criptofonia.

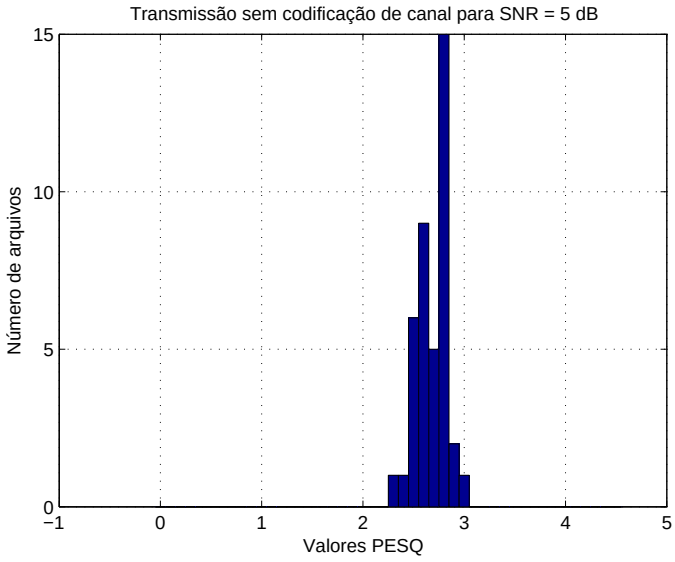

Figura 5.4: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 5 dB utilizando uma transmissão sem criptofonia.

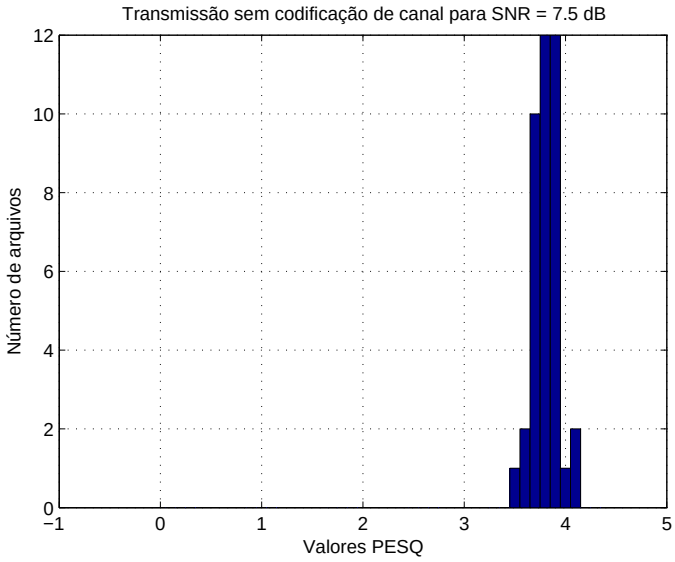

Figura 5.5: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 7.5 dB utilizando uma transmissão sem criptofonia.

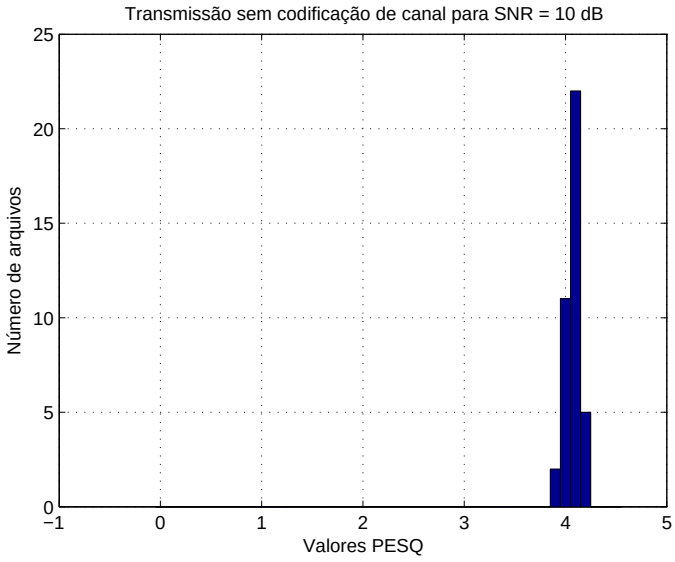

Figura 5.6: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 10 dB utilizando uma transmissão sem criptofonia.

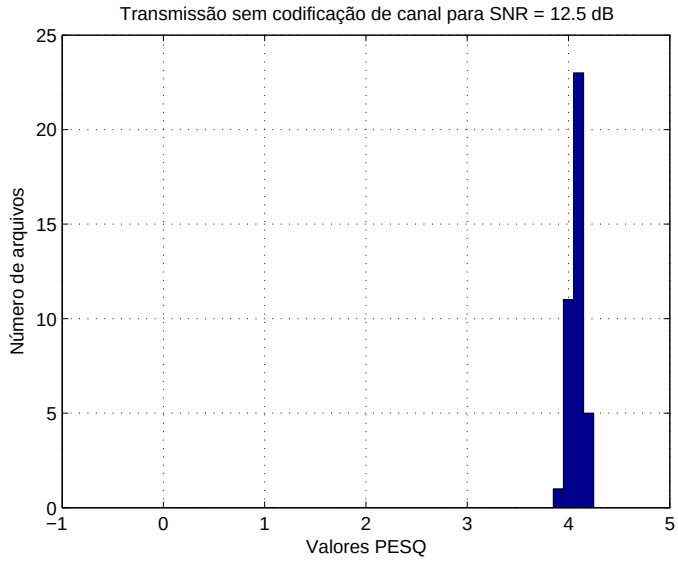

Figura 5.7: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 12.5 dB utilizando uma transmissão sem criptofonia.

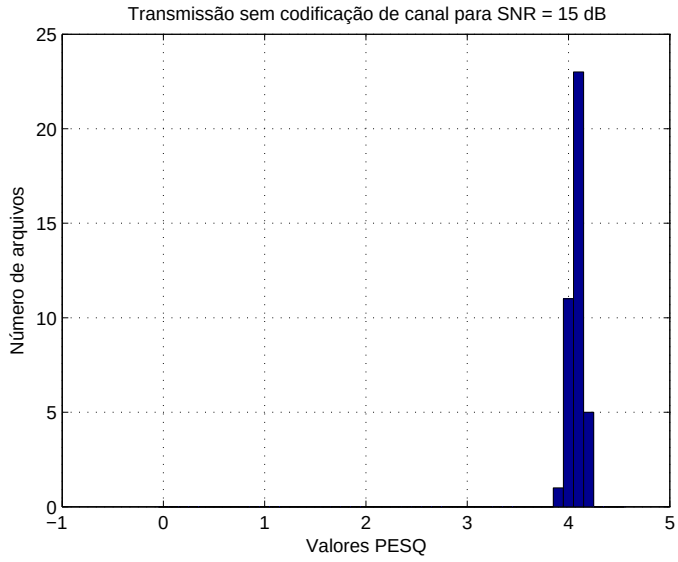

Figura 5.8: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 15 dB utilizando uma transmissão sem criptofonia.

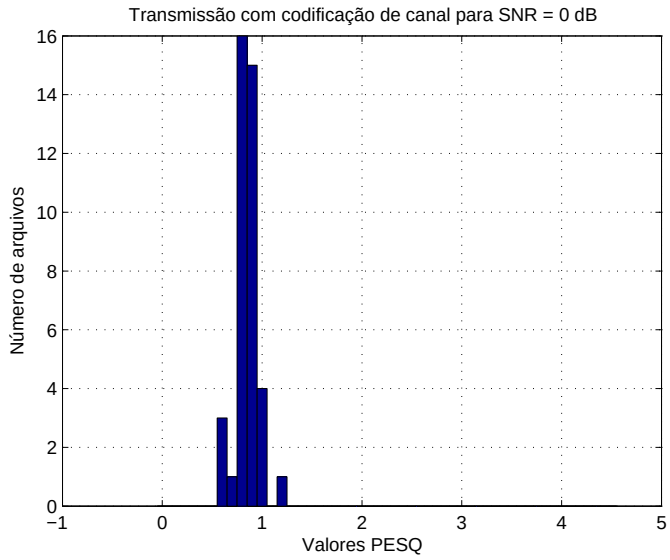

Figura 5.9: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 0 dB utilizando uma transmissão sem criptofonia.

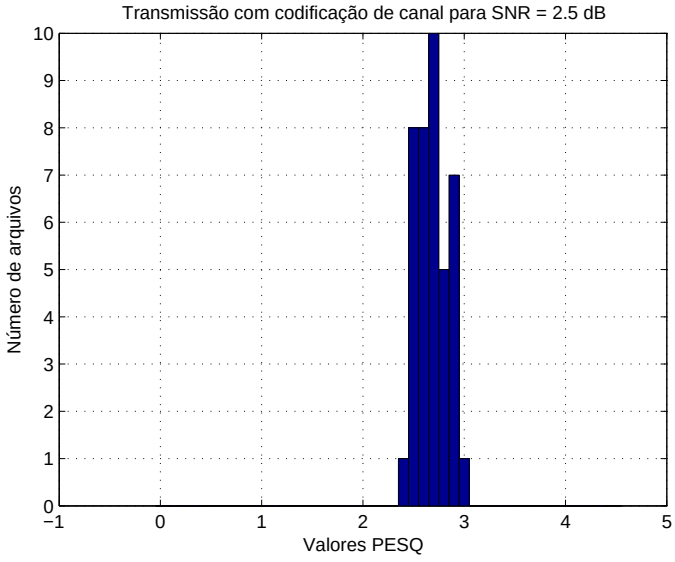

Figura 5.10: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 2.5 dB utilizando uma transmissão sem criptofonia.

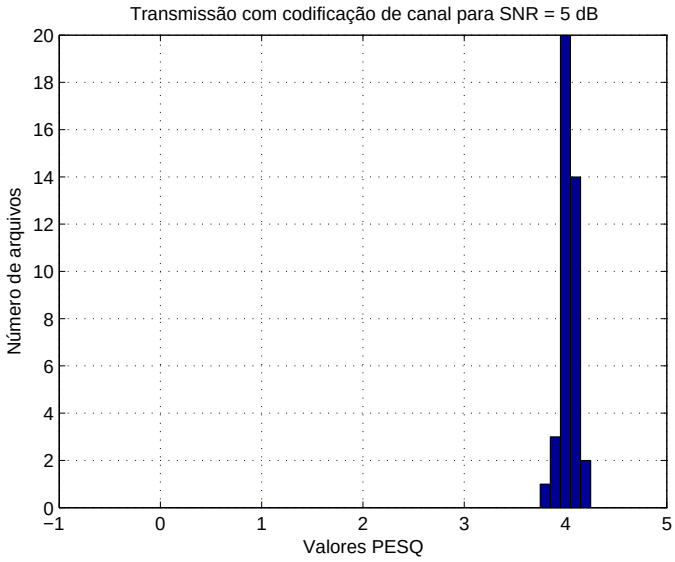

Figura 5.11: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 5 dB utilizando uma transmissão sem criptofonia.

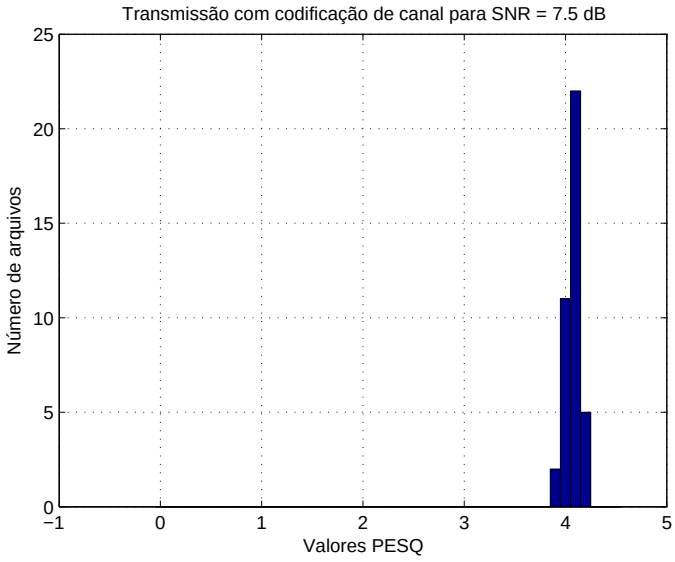

Figura 5.12: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 7.5 dB utilizando uma transmissão sem criptofonia.

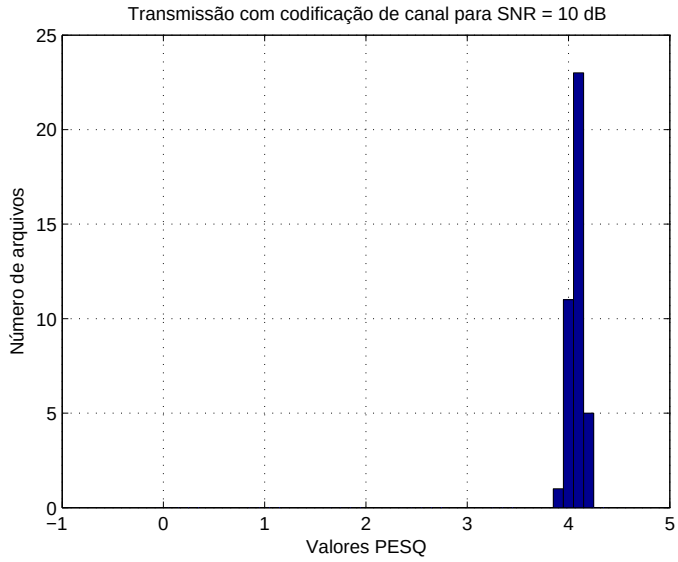

Figura 5.13: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 10 dB utilizando uma transmissão sem criptofonia.

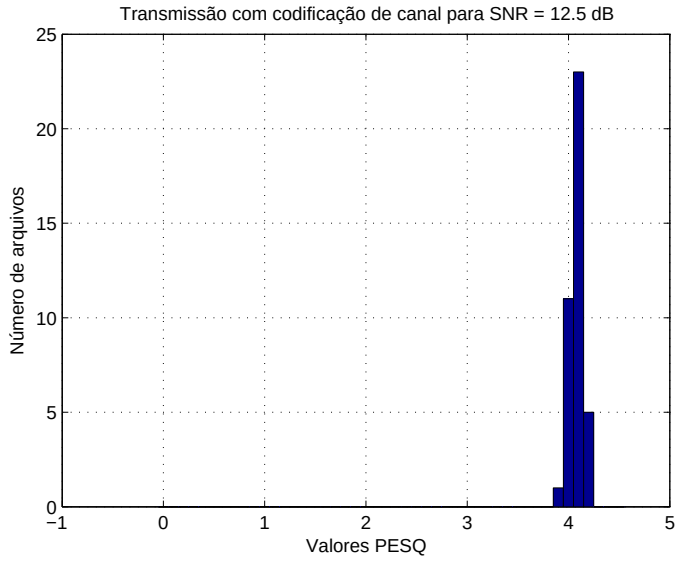

Figura 5.14: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 12.5 dB utilizando uma transmissão sem criptofonia.

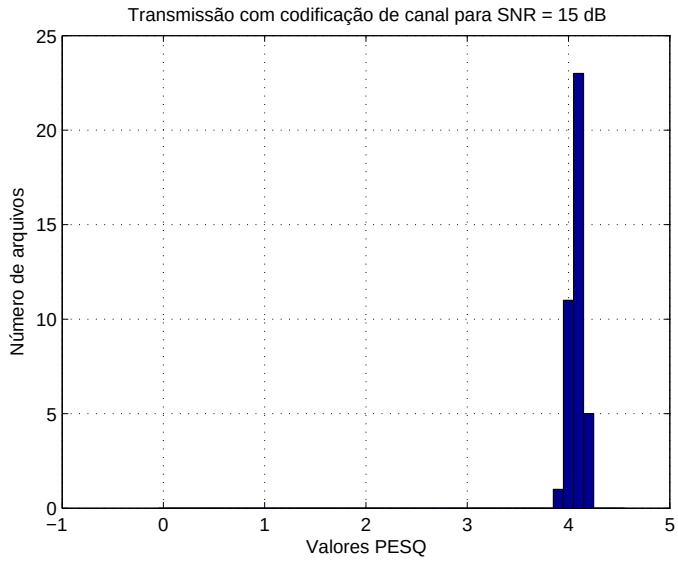

Figura 5.15: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 15 dB utilizando uma transmissão sem criptofonia.

O gráfico apresentado pela Figura 5.16 mostra a curva PESQ x SNR para sistemas sem e com codificação de canal para um sistema GSM sem criptofonia. Cada ponto no gráfico representa a média dos valores PESQ dos arquivos de fala recebidos, valores estes que podem ser observados nos gráficos apresentados nas Figuras de 5.2 a 5.15. As barras horizontais representam a incerteza das medidas em cada ponto, a qual é calculada considerando que os valores PESQ obedecem uma distribuição gaussiana, então utilizamos duas vezes o desvio padrão para uma segurança de 95%.

Este gráfico mostra que para valores de SNR entre 2.5 e 7.5 dB, temos em média um valor PESQ mais alto para sinais cuja transmissão foi feita utilizando codificação de canal, o que significa que estes têm uma qualidade melhor que os sinais transmitidos sem codificação de canal. Se observarmos o gráfico apresentado na Seção 5.1.1, vemos que a codificação de canal diminui a quantidade de erros no sinal recebido e por essa razão temos sinais com uma qualidade maior quando utilizamos este tipo de codificação. Para valores de SNR a partir de 10 dB, o valor PESQ recebido pelos sinais transmitidos sem e com codificação são praticamente iguais, isto acontece pois a recepção para estes valores de SNR acontece com um número consideravelmente reduzido de erros, como podemos observar no gráfico exibido na Figura 5.1.

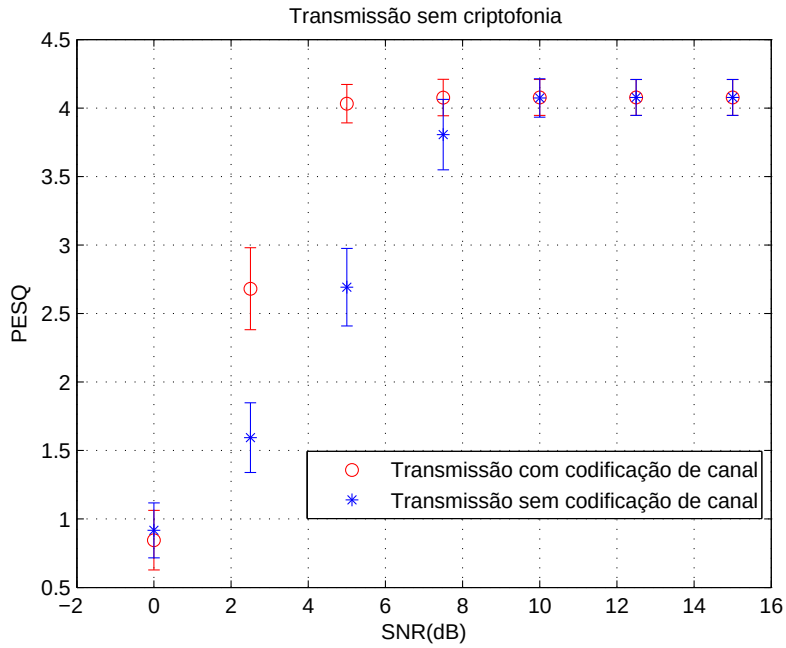

Figura 5.16: Média dos valores PESQ para sinais sem criptofonia usando transmissão GSM

5.3 Simulação III

Esta Simulação é bem parecida com a Simulação II descrita na Seção 5.2, a diferença é que antes do sinal de voz ser entregue ao CODEC AMR, ele é cifrado utilizando a técnica CSI-F baseada em transformadas ortogonais. A transformada ortogonal implementada para esta simulação foi a DCT (*Discrete Cosine Transform*). Temos por objetivo comparar os resultados encontrados nesta simulação com os apresentados na Seção 5.2.1 e a partir destes dados observar os efeitos da critofonia no sinal de audio recebido.

Os dados utilizados nesta simulação foram:

- Frequencia de amostragem: 8 kHz;
- Número de frases: 40;
- Número mínimo de frases por locutor: 5;
- Total de valores de SNR: 7 pontos igualmente distribuídos entre 0 e 15 dB;
- Duração de cada bloco de voz: 20 ms;
- Número de subbandas no qual o sinal foi dividido: 8;

- Rotação provocada pela matriz de permutação: $\Phi_I = \Phi_I^{max} = 53{,}97°$;
- Número de pontos usado no cálculo da DCT para cada bloco: 160;

5.3.1 Resultados

Os gráficos representados nas Figuras de 5.17 a 5.30 mostram as distribuições dos arquivos de fala segundo o seu valor PESQ para uma dada SNR, utilizando transmissão sem e com codificação de canal.

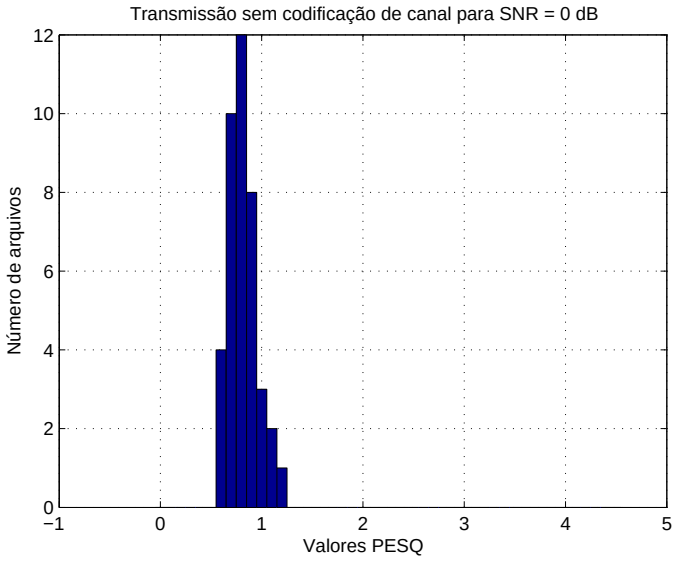

Figura 5.17: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 0 dB utilizando uma transmissão com criptofonia.

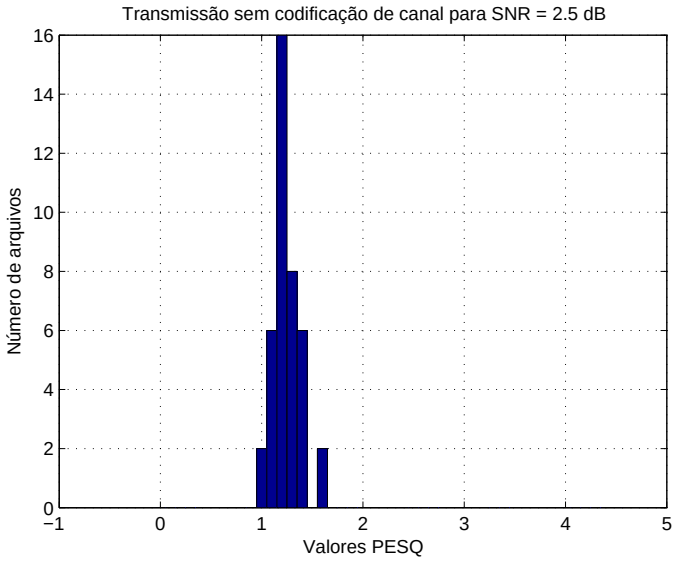

Figura 5.18: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 2.5 dB utilizando uma transmissão com criptofonia.

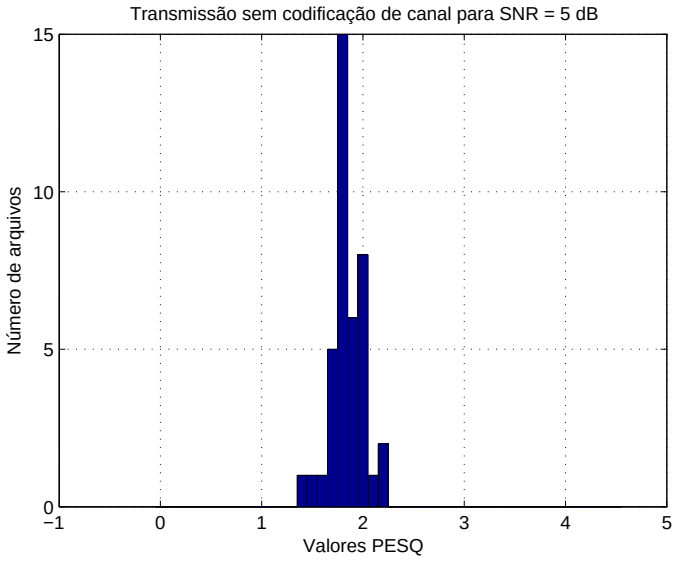

Figura 5.19: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 5 dB utilizando uma transmissão com criptofonia.

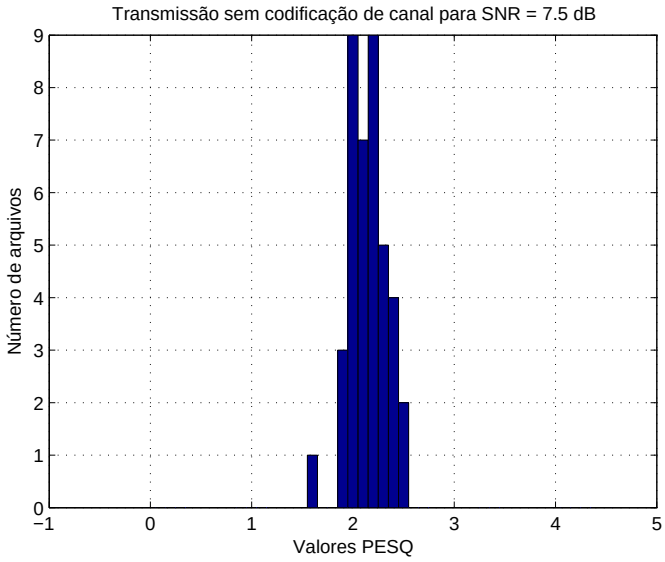

Figura 5.20: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 7.5 dB utilizando uma transmissão com criptofonia.

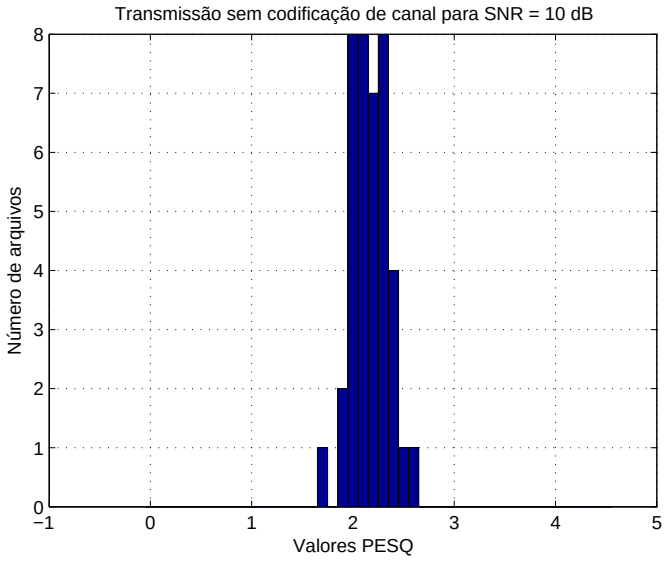

Figura 5.21: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 10 dB utilizando uma transmissão com criptofonia.

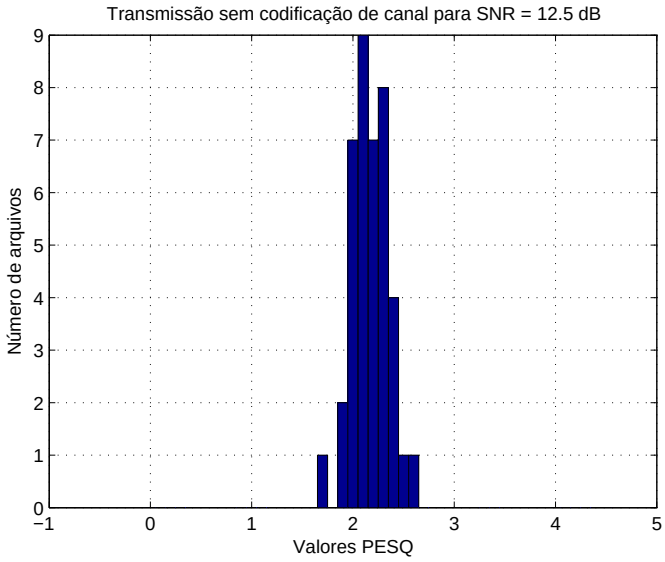

Figura 5.22: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 12.5 dB utilizando uma transmissão com criptofonia.

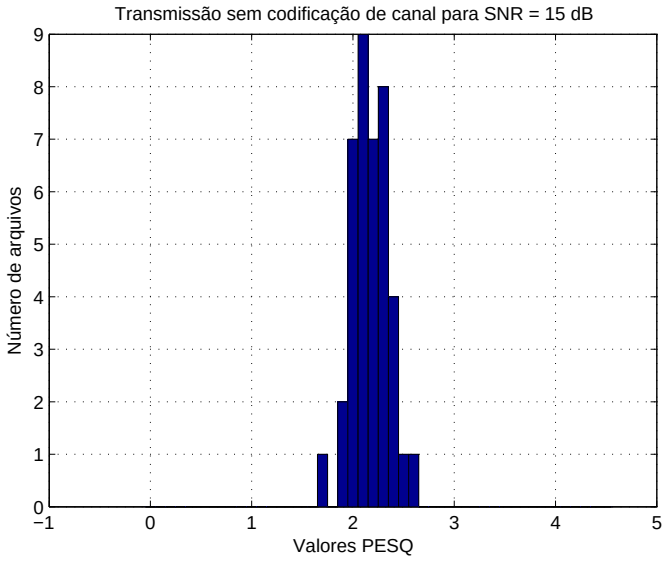

Figura 5.23: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 15 dB utilizando uma transmissão com criptofonia.

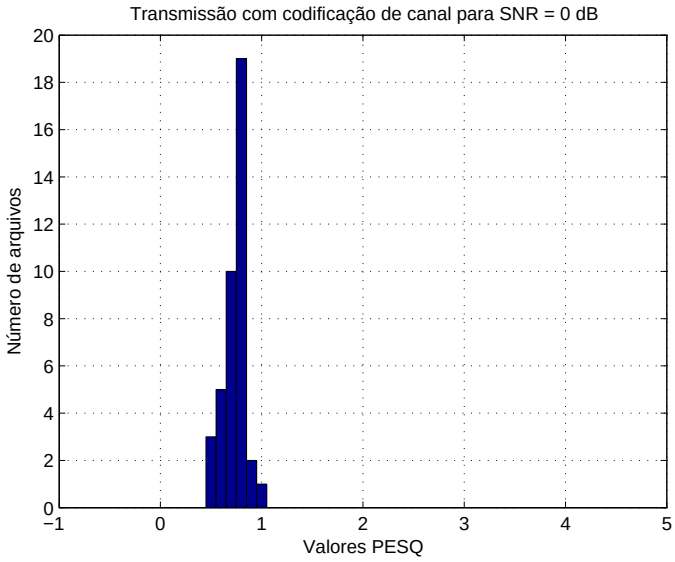

Figura 5.24: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 0 dB utilizando uma transmissão com criptofonia.

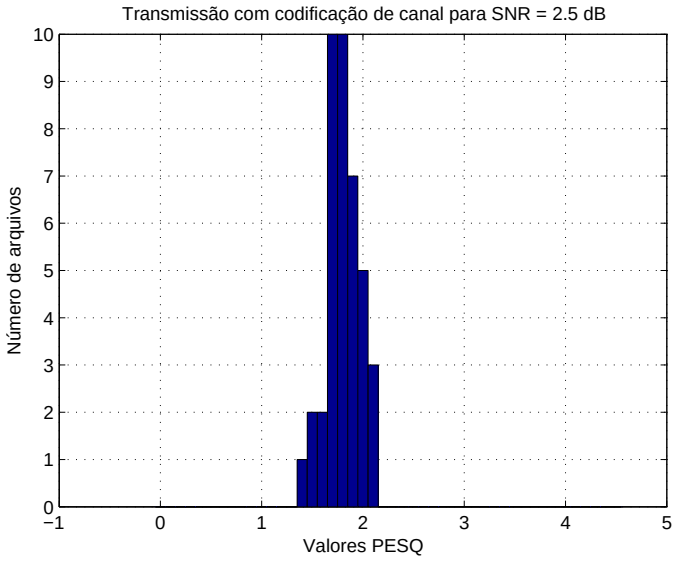

Figura 5.25: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 2.5 dB utilizando uma transmissão com criptofonia.

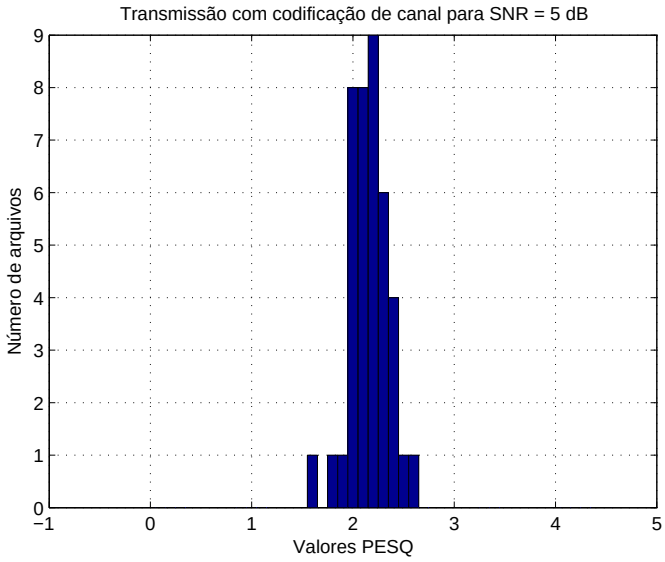

Figura 5.26: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 5 dB utilizando uma transmissão com criptofonia.

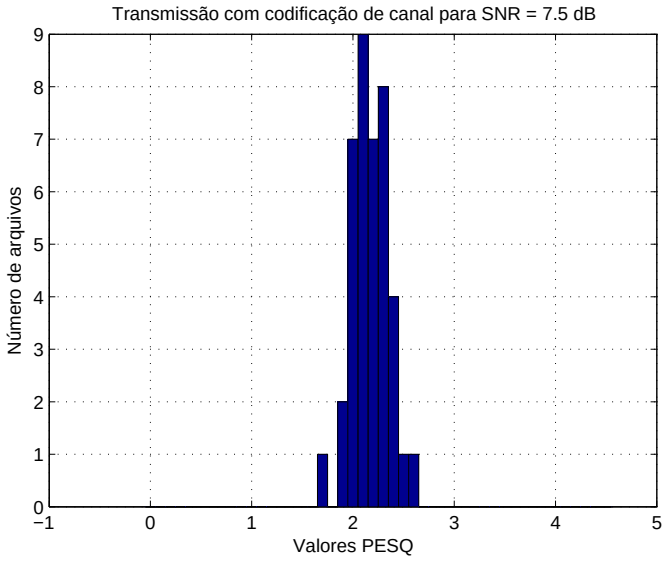

Figura 5.27: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 7.5 dB utilizando uma transmissão com criptofonia.

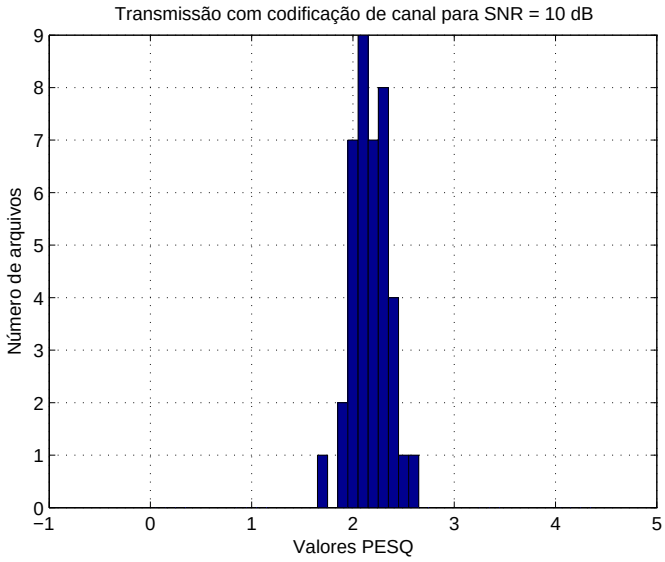

Figura 5.28: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 10 dB utilizando uma transmissão com criptofonia.

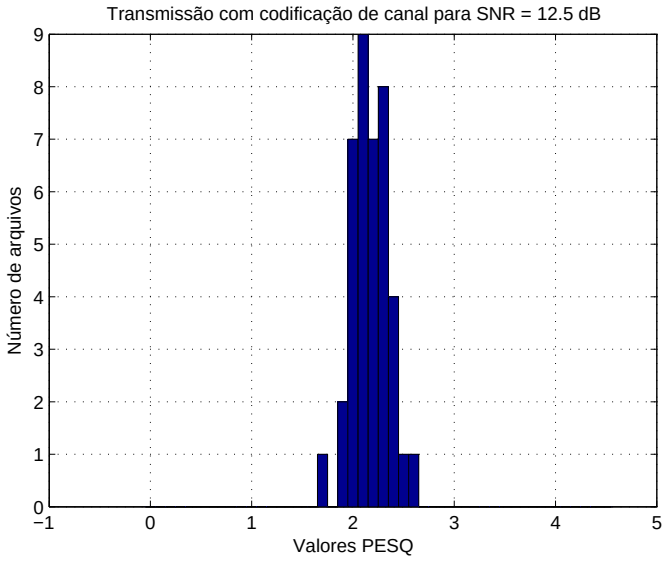

Figura 5.29: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 12.5 dB utilizando uma transmissão com criptofonia.

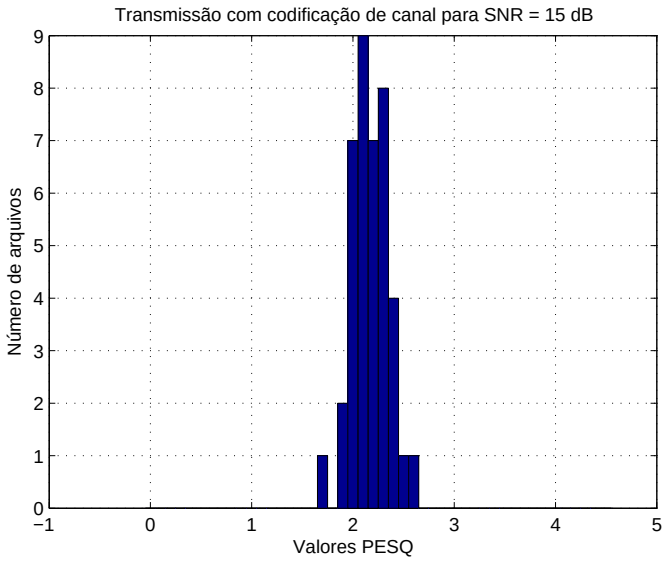

Figura 5.30: Distribuição dos arquivos de fala segundo seus valores PESQ para uma SNR = 15 dB utilizando uma transmissão com criptofonia.

O gráfico da Figura 5.31 é similar ao gráfico da Figura 5.16 e comparando os dois podemos notar que a qualidade do sinal transmitido é menor se usarmos a criptofonia no sistema GSM. Apesar de obtermos valores PESQ em torno de 2 que é considerado um valor baixo, ainda assim o audio tem qualidade suficiente para que o destinatário possa compreender o que foi dito. Os sinais que são transmitidos com codificação de voz, assim como foi observado na Seção 5.2.1, também recebem valores PESQ mais altos que os transmitidos sem a codificação de canal para valores de SNR entre 2.5 e 7.5 dB e valores de PESQ praticamente iguais para SNR acima de 10 dB.

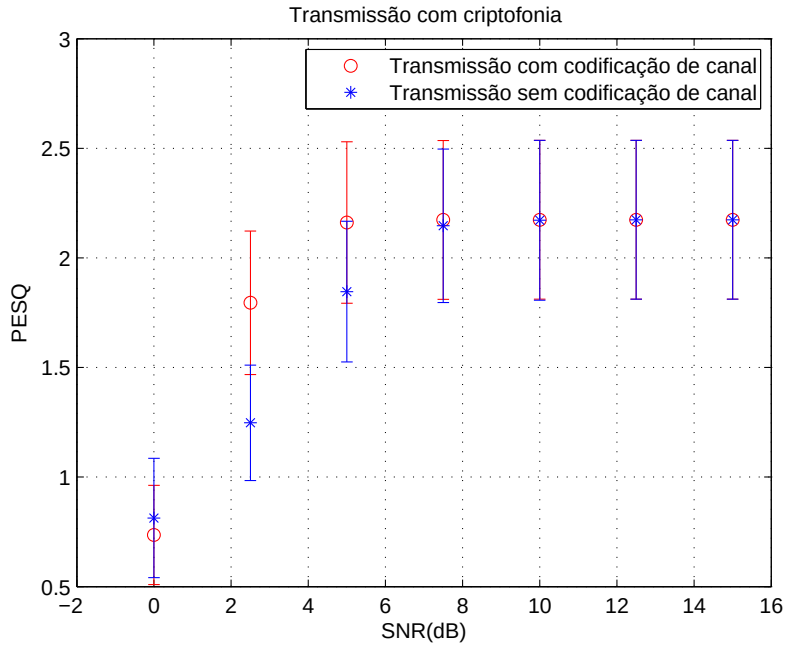

Figura 5.31: Média dos valores PESQ para sinais com criptofonia usando transmissão GSM

Capítulo 6

Conclusões

6.1 Resumo e Principais Conclusões

Este projeto visa observar o funcionamento da codificação de canal, que é uma importante técnica utilizada para melhorar a recepção de sinais transmitidos no sistema GSM, e também investigar o funcionamento de uma das técnicas de criptofonia proposta em [3] verificando como o sistema se comporta com a adição desse elemento.

No Capítulo 1, fomos introduzidos ao tema do projeto através da história da telefonia celular, que mostra o surgimento e desenvolvimento dos sistemas de telefonia móvel e apresenta o problema do sigilo das conversas telefônicas realizadas nesse meio. Também fomos apresentados à arquitetura do sistema GSM, sendo descritos os principais subsistemas da rede e suas funcionalidades.

No Capítulo 2, temos uma explicação sobre a teoria dos processos de codificação de canal. As codificações cíclicas e convolucionais, que são técnicas empregadas pelo sistema GSM para a melhora dos sinais recebidos, são apresentadas assim como os processos para desfazer a codificação de canal, recuperando o máximo possível do sinal originalmente transmitido.

O Capítulo 3 é dedicado ao tratamento que o sinal de voz recebe no simulador implementado. Nos foi apresentado uma explicação geral sobre os tipos de codificadores de voz existentes e as técnicas de criptofonia. A criptofonia CSI-F baseadas em bancos de filtro e em transformadas ortogonais foram tratadas com uma atenção especial por serem as mais adequadas para o projeto proposto, uma vez que elas podem ser implementadas sem fazer alteração de hardware e produzem sinais com as características desejadas para o que é proposto.

O Capítulo 4 apresenta uma descrição detalhada de cada módulo do simulador implementado. Nesta parte do trabalho entendemos como o simulador está dividido, quais as funções de cada módulo, quais os sinais trocados entre os módulos e outras características do projeto.

O Capítulo 5 apresenta os resultados obtidos através de simulações feitas com o sistema de codificação de canal implementado. A primeira simulação nos mostra que a codificação de canal proposta pelas especificações atende o que ela propõe, que é diminuir a quantidade de erros no sinal recebido. As Simulações II e III servem para observamos o comportamento da criptofonia quando aplicada a um sistema GSM. Os dados obtidos nos mostram que fazendo uso da criptofonia perdemos um pouco de qualidade do sinal de voz se comparado a recepção que não usa a cifragem. Apesar da perda de qualidade ainda é possível compreender o que foi dito, comprovamos que a técnica proposta atende bem ao objetivo de criptofonia para um sistema de telecomunicações móveis.

6.2 Trabalhos Futuros

Como sugestão para trabalhos futuros que podem melhorar o simulador e nos ajudar a achar explicações melhores para o funcionamento da criptofonia usando CSI-F são os seguintes:

- Implementar a troca de chaves para criptofonia;
- Implementar a criptofonia CSI-F baseada em banco de filtros;
- Utilizar metodologias para geração e troca automática de chaves para criptofonia.

Referências Bibliográficas

[1] M. MOULY, M. B. P., *The GSM System for Mobile Communications*. Cell & Sys., 1992.

[2] LEE, W. C. Y., *Mobile Cellular Telecommunications (Analog and Digital Systems)*. $2^{a.}$ Edição. McGraw-Hill, Inc., 1995.

[3] Andrade Jr., J. F., *Criptofonia Aplicada a Sistemas Modernos de Comunicações*. Tese de M.Sc., COPPE Elétrica, Outubro 2008.

[4] LIN, S., Costello Jr., D. J., *Error Control Coding*. $2^{a.}$ Edição. Pearson Prentice Hall, 2003.

[5] DINIZ, P. S. R., "Principles of Wireless Communication.", 2008. Notas de aula da disciplina Transmissão Digital do DEL/UFRJ.

[6] HAYKIN, S., *Sistemas de Comunicação Analógicos e Digitais*. $4^{a.}$ Edição. Bookman Editora S.A., 2004.

[7] PROAKIS, J. G., *Digital Communications*. $3^{a.}$ Edição. McGraw-Hill Internations Editions, 1995.

[8] EKSTRØM, A. N., MIKKELSEN, J. H., "GSMsim: A MATLAB Implementation of a GSM Simulation Plataform", 1997. Relatório Técnico do Instituto de Sistemas Eletrônico da Universidade de Aalborg.

[9] 3GPP, "Technical Specification 3rd Generation Partnership Project; Technical Specification Group GERAN; Channel coding", 2001. 3GPP TS 45.003 v4.0.0 (2001-01).

[10] MARTINS, W. A., "Influência Mútua de Técnica de Supressão de CCI no GPRS", Agosto 2007. Projeto de Final de curso do DEL/UFRJ.

[11] 3GPP, “Technical Specification 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; ANSI C code for the Adaptive Multi Rate speech codec”, 2001. 3GPP TS 26.073 v4.0.0 (2001-01).