César Miguel dos Santos Martins

Implementação em Plataformas SDR de Esquemas de Codificação para Segurança na Camada Física Baseados em Códigos Curtos e Técnicas de *Interleaving*

Dissertação de mestrado em Engenharia Eletrotécnica e de Computadores

Julho de 2017

· U C ·

UNIVERSIDADE DE COIMBRA

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Implementação em Plataformas SDR de Esquemas de Codificação para Segurança na Camada Física Baseados em Códigos Curtos e Técnicas de *Interleaving*

César Miguel dos Santos Martins

Orientadores

Marco Alexandre Cravo Gomes

João Paulo da Silva Machado Garcia Vilela

Júri

Maria do Carmo Raposo de Medeiros	(Presidente)
Marco Alexandre Cravo Gomes	(Orientador)
Vítor Manuel Mendes da Silva	(Vogal)

Julho de 2017

Ao meu avô Arlindo…

Agradecimentos

Começo os meus agradecimentos pelos maiores responsáveis por ter chegado até aqui: os meus pais. A eles e ao meu irmão, o meu obrigado pela educação, pelas oportunidades e por tudo o que me faz ser o que sou e que me permitiu chegar a esta etapa.

Agradeço, também, aos meus orientadores, Marco Gomes e João Vilela, pela oportunidade de trabalhar no projeto SWING2 (PTDC/EEI-TEL/3684/2014), pelo acompanhamento, pela paciência e pela dedicação. Do mesmo modo, não poderia deixar de agradecer ao SWING2, ao Instituto de Telecomunicações e ao Departamento de Engenharia Eletrotécnica e de Computadores da Universidade de Coimbra pelas instalações e material disponibilizados e sem os quais não seria possível realizar este trabalho.

À Joana, por me ter aturado nestes últimos meses e por toda a ajuda e amizade.

Às pessoas que me acompanharam durante o meu percurso, nomeadamente o Ricardo Loureiro, que é uma riqueza, o Pudim e a Andreia, pelas infindáveis noites (e manhãs) no DEEC, ao fritanço, à Solange, ao Zé, aos meus colegas de laboratório e às minhas afilhadas.

Aos Rewind e ao 1185, por compreenderem a minha indisponibilidade e pelos momentos de descontração.

À Sara, por tudo, à Andreia, ao David, ao João e ao Pedro, por coisas, sabe-se lá quais.

À Daniela, pelos anos de partilha de casa, de histórias (muito estranhas) e de apoio.

Ich danke auch den ‚Deutschen' Simone, Francisca und Maria, die meine Sprachlichen Berater sind.

A todos os meus amigos, familiares e professores que contribuíram para o meu crescimento e à Nicola, por vender cápsulas de café bom e barato,

Muito Obrigado.

Este trabalho é suportado pelo projeto SWING2 (PTDC/EEI-TEL/3684/2014), financiado pelos Fundos Europeus Estruturais e de Investimento (FEEI) através do Programa Operacional Competitividade e Internacionalização - COMPETE 2020 e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no âmbito do projeto POCI-01-0145-FEDER-016753.

Resumo

Os últimos anos trouxeram um crescimento das comunicações sem fios, o que se refletiu numa maior preocupação com a segurança dos dados transmitidos entre dispositivos. Apesar da criptografia providenciar níveis satisfatórios de segurança, não é inquebrável, pelo que não se consegue garantir que não haja alguma fuga de informação. Nesse sentido, têm sido desenvolvidos métodos de introduzir segurança também na camada física, como complemento à segurança já existente.

Recentemente, foi proposto um esquema que recorre a uma chave aleatória de *interleaving*, que é usada para baralhar a mensagem a transmitir, e que é, posteriormente, anexada à mesma, sendo ativado um *jammer* durante a transmissão dos bits correspondentes à chave gerada, de forma a criar uma vantagem em relação a um possível *eavesdropper*. Paralelamente a este, foi desenvolvido um esquema idêntico, mas em que os bits correspondentes à chave são apagados depois da mensagem ser codificada. Este último esquema parte do pressuposto da existência de uma vantagem do recetor legítimo face a um possível *eavesdropper*, expressa em termos de uma melhor relação sinal-ruído (por exemplo, a transmissão entre os intervenientes legítimos é realizada dentro de uma sala, garantindo que não existe um *eavesdropper* dentro da mesma), permitindo ao recetor legítimo a recuperação correta da chave, o que não é possível em situações de comunicação mais degradadas.

Nesta dissertação, é realizada a prova de conceito com a implementação em ambiente real de ambos os métodos em plataformas de rádio definido por software (SDR – *Software Defined Radio*), programadas com recurso ao MATLAB/Simulink. São, também, apresentados cenários de teste adequados, onde foram realizadas experiências que mostram a existência de um acréscimo de segurança introduzido por estes esquemas num cenário real. No decorrer da implementação, foi, ainda, proposta uma variação a ambos os esquemas, substituindo o *interleaver* por um *scrambler*, tendo sido obtidos resultados significativamente melhores.

Nesta dissertação, são, também, descritos os problemas e dificuldades inerentes à implementação prática e teste de um sistema de comunicação genérico em plataformas SDR. Em concreto, e a título de exemplo, é proposto e foi implementado um processo de automatização dos testes, assente em comunicação sobre UDP, que permite a sincronização de vários SDR (que modelam os sistemas transmissores e recetores), que pode ser aplicado a outros modelos de Simulink que representem um canal de escuta.

Com estes resultados, mostra-se que os esquemas propostos cumprem os requisitos de garantir uma transmissão segura e fidedigna, podendo vir a ser um importante complemento à segurança

das comunicações, principalmente em dispositivos móveis, como telemóveis (onde a segurança baseada em proximidade está a ganhar força) ou dispositivos da Internet das Coisas de baixo consumo, onde pode não ser possível utilizar os métodos da criptografia.

Palavras-Chave

Segurança na camada física; Codificação de canal; Rádio definido por *software*; *Interleaving*; *Scrambling*; Matlab/Simulink; Implementação/*Testbed*

Abstract

The last few years have brought a growth in the usage of wireless communications, which was reflected in an increased concern for the security of data transmitted between devices. Although cryptography provides satisfactory levels of security, it is not unbreakable, so one cannot guarantee that there is no leakage of information. Therefore, methods have been developed to introduce security also at the physical layer, using codes with finite blocklengths, in addition to existing security.

Recently, a new schema has been proposed that generates a random interleaving key, which is used to shuffle the message to be transmitted, being afterward attached to it for transmission, while a jammer is turned on during the transmission of the key, in order to create an advantage over a possible eavesdropper. Meanwhile, an identical schema was developed, in which the bits corresponding to the key are erased after the message is encoded. This latter schema is based on the assumption of a legitimate receiver's advantage over a possible eavesdropper, expressed in terms of a better signal-to-noise ratio (for example, by performing the transmission within a room, ensuring that there is no eavesdropper inside it), allowing the legitimate receiver to correctly recover the key, which is not possible in more degraded communication situations.

In this dissertation, the proof of concept is performed with the implementation of both methods on Software Defined Radio (SDR) platforms, programmed using MATLAB/Simulink. Appropriate test scenarios are also presented, where experiments were conducted that show the existence of an increase in the security introduced by these schemas in a real scenario. In the course of implementation, a variation of both schemas was proposed, replacing the interleaver with a scrambler, and significantly better results were obtained.

In this dissertation, the problems and difficulties inherent to the practical implementation and testing of a generic communication system on SDR platforms are also described. Particularly, as an example, a test automation process based on communication over UDP is proposed and implemented, which allows the synchronization of several computers operating SDR (which can model the transmitter and receiver systems), which can be applied to other Simulink models using the wiretap channel.

With these results, it has been shown that the proposed schemas ensure a safe and reliable transmission. These may be an important complement to the security of communications, mainly on mobile devices, such as mobile phones (where proximity-based security is gaining traction) or low power devices from the Internet of Things, where typical cryptographic mechanisms may not be applicable.

Keywords

Physical layer security; Channel Coding; Software defined radio; Interleaving; Scrambling; Matlab/Simulink; Testbed implementation

Índice

Lista de figuras

Lista de tabelas

Lista de Abreviações

ADC – Analog-to-Digital Converter

AGC – Automatic Gain Controller

APP – *a posteriori* Probability

AWGN – Additive White Gaussian Noise

BCH – Bose-Chaudhuri-Hocquenguem

BER – Bit Error Ratio

BER-CDF – Bit Error Rate Cumulative Distribution Function

BF – Bit-Flipping

DAC – Digital-to-Analog Converter

DEEC – Departamento de Engenharia Eletrotécnica e de Computadores

ELTED – Early-Late Timing Error Detector

FFT – Fast Fourier Transform

FPGA – Field-Programmable Gate Array

GPS – Global Positioning System

GSM – Global System for Mobile communications

GTED – Gardner Timing Error Detector

ICS – Interleaved Coding for Secrecy

ICS-HK – Interleaved Coding for Secrecy with a Hidden Key

ISI – InterSymbol Interference

LDPC – Low-Density Parity-Check

LLR – Log-Likelihood Ratio

MER – Modulation Error Ratio

MIMO – Multiple-Input and Multiple-Output

MLG – Majority-LoGic

MLTED – Maximum Likelihood Timing Error Detector

MMTED – Mueller and Müller Timing Error Detector

PCHIP – Piecewise Cubic Hermite Interpolating Polynomial

PCI – Peripheral Component Interconnect

PI – Proportional Integral

PLL – Phase-Locked Loop

PSK – Phase-Shift Keying

QPSK – Quadrature Phase-Shift Keying

RRC – Root-Raised-Cosine

SCS – Scrambled Coding for Secrecy

SCS-HK – Scrambled Coding for Secrecy with a Hidden Key

SDR – Software Defined Radio

SMA – SubMiniature version A

SNR – Signal-to-Noise Ratio

SPA – Sum-Product Algorithm

TCP – Transmission Control Protocol

TED – Timing Error Detector

UC – Universidade de Coimbra

UDP – User Datagram Protocol

USB – Universal Serial Bus

USRP – Universal Software Radio Peripheral

VCC – Voltage Controlled Clock

VCO – Voltage Controlled Oscillator

ZCTED – Zero-Crossing Timing Error Detector

1. Introdução

Ao longo das últimas décadas, tem-se assistido a um crescimento exponencial da informação guardada digitalmente e da necessidade de a transferir entre diversos intervenientes. A massificação da Internet, dos dispositivos móveis (sejam eles computadores, telemóveis, *tablets*, etc.) e os avanços no desenvolvimento das redes sem fios vieram trazer comodidade aos utilizadores. No entanto, essa comodidade trouxe, também, novos desafios ao nível da segurança, pois a informação passou a estar disponível para qualquer recetor ao alcance, deixando de haver controlo sobre quem, além do destinatário, está à escuta. A segurança é, nos sistemas atuais, assegurada pelo uso de métodos criptográficos [1], implementados nas camadas superiores da pilha protocolar do sistema de transmissão. No entanto, apesar de existir um número cada vez maior de cifras e estas serem cada vez mais sofisticadas, existem também algoritmos cada vez mais sofisticados e maior poder de processamento para as quebrar. A recente aposta na Internet das Coisas veio tornar esta questão mais premente, pois há ainda mais informação, muita dela sensível, a ser transmitida por todo o lado.

Com o objetivo de encontrar soluções para este problema, a comunidade científica tem visto com crescente interesse o estudo da segurança na camada física como complemento à criptografia. Este conceito visa o aproveitamento das imperfeições dos canais de transmissão, nomeadamente da sua natureza aleatória, para garantir comunicações seguras. Este estudo tem as suas bases no trabalho de Wyner e no modelo de canal por ele proposto, conhecido como *Wiretap Channel* [2], onde um recetor ilegítimo, Eve, tenta intercetar, de forma passiva, uma mensagem enviada de um transmissor, Alice, para um recetor legítimo, Bob [3] [2].

Com este cenário em mente, em [4] e [5] foram propostos dois esquemas – *Interleaved Coding for Secrecy* (ICS) e ICS *with a Hidden Key* (ICS-HK) –, assentes em *interleaving* e *jamming*, para alcançar comunicações seguras e fidedignas, usando como métricas de segurança a taxa de erros (BER) e a função cumulativa de distribuição do BER (BER-CDF), definida em [6]. No primeiro esquema, é gerada uma chave de permutação diferente para cada bloco de dados a transmitir, que é usada para baralhar a mensagem a enviar, por meio de *interleaving*. A chave gerada é codificada por um codificador externo e concatenada com a mensagem baralhada, sendo o bloco concatenado resultante codificado por um codificador interno, antes de ser enviado pelo canal. Para garantir uma vantagem em relação à Eve, este esquema contempla, ainda, a utilização de um *jammer*, ativado apenas durante a transmissão dos bits codificados correspondentes à chave. No segundo esquema, a mensagem é baralhada da mesma forma, recorrendo a uma chave aleatória, mas esta não é codificada pelo codificador interno. Além disso, os bits codificados pelo codificador externo

correspondentes à chave são apagados antes da transmissão, simulando o efeito de um *jammer* de potência infinita. Por isto, este esquema dispensa o uso do *jammer*, mas exige uma vantagem prévia do Bob sobre a Eve. Ambos os esquemas foram validados através de simulações computacionais, provando a sua utilidade em canais AWGN.

1.1. Objetivos

Esta dissertação teve, como principal objetivo, a prova de conceito dos esquemas ICS e ICS-HK, através da sua implementação em ambiente real, em plataformas de rádio definido por *software* (SDR – *Software Defined Radio*). O facto de estas serem plataformas reprogramáveis tornava possível testar diferentes configurações dos sistemas e o desenvolvimento dos mesmos, através de *software*. Diretamente ligado ao objetivo principal, era também objetivo definir cenários de teste reais para o uso destes esquemas de segurança e, por fim, a realização dos testes com os modelos implementados, para que se conseguisse aferir a utilidade dos mesmos e confirmar os resultados teóricos.

1.2. Contribuições

Atendendo aos objetivos da dissertação e ao trabalho desenvolvido para os alcançar, as contribuições desta dissertação podem-se resumir em:

- Proposta de 2 novos esquemas de codificação para segurança – *Scrambled Coding for Secrecy* (SCS) e SCS *with a Hidden Key* (SCS-HK) –, baseados nos esquemas inicialmente propostos – ICS e ICS-HK;
- Implementação dos 4 esquemas referidos em Simulink, para teste em plataformas SDR;
- Proposta de cenários de teste para os esquemas implementados;
- Proposta de processo de automatização de testes para cenários de canal *wiretap*;
- Realização de testes em ambiente real, utilizando plataformas SDR, para os modelos ICS-HK e SCS-HK.

Com os resultados obtidos, este trabalho pretende ser um contributo para o avanço da segurança na camada física, mostrando que se pode conseguir segurança contra a Eve num cenário real, mantendo a fiabilidade da transmissão para o Bob, desde que se garanta a existência de uma determinada vantagem por parte do recetor legítimo.

1.3. Organização da dissertação

Este documento está estruturado em 7 capítulos.

No capítulo 2, são introduzidos conceitos base de segurança na camada física e métricas de segurança usadas neste contexto. São, ainda, apresentadas algumas técnicas usadas no contexto prático de assegurar segurança na camada física e são descritos os métodos de que esta dissertação se propõe fazer prova de conceito, com a sua implementação em SDR. É, ainda, proposta uma variação dos métodos ICS e ICS-HK, com o uso de um *scrambler* no lugar do *interleaver*.

O capítulo 3 apresenta os rádios definidos por *software*. São descritas as suas vantagens, referenciados alguns SDR presentes no mercado e é feita uma breve descrição das suas possíveis funções, quando enquadrados num sistema de comunicação. São referidos, ainda, os possíveis modos de operação e as limitações ao funcionamento destes rádios. Por fim, enumeram-se algumas ferramentas de desenvolvimento para SDR e é apresentado, com mais detalhe, o SDR utilizado.

A teoria que suporta a implementação em SDR do sistema de comunicação digital real utilizado encontra-se no capítulo 4. Nele, descreve-se um sistema Transcetor SDR, apresentando os percursos de transmissão e receção do mesmo. Introduzem-se, também, alguns conceitos de codificação de canal, necessários para implementar os esquemas pretendidos, nomeadamente *interleaving*, *scrambling*, *puncturing* e codificação LDPC. O capítulo termina com a descrição dos processos inerentes à transmissão do sinal, como a modulação e a formatação de pulso, e os processos de recuperação de sinal, como o controlo de ganho e os sincronismos de portadora, relógio e *frame*.

De seguida, no início do capítulo 5, são novamente descritos os modelos implementados, mas numa perspetiva mais prática, apresentando a forma como foram implementados em Simulink. Seguem-se resumos dos parâmetros de configuração dos sistemas implementados – como frequência de transmissão, código, tamanho de mensagem, entre outros – e *hardware* – computadores, SDR e antenas. São, ainda, propostos cenários para os testes de cada um dos esquemas, bem como a forma como foram concretizados. Por fim, descrevem-se alguns problemas decorrentes da implementação e as formas de os ultrapassar.

Ao longo do capítulo 6, definem-se alguns critérios para analisar os dados obtidos nas experiências efetuadas, bem como as bases estatísticas para a análise dos resultados, que é feita no final do capítulo.

Finalmente, no capítulo 7, são apresentadas as principais conclusões do trabalho, face aos resultados obtidos, e são apontadas algumas direções/sugestões para a evolução do trabalho.

Os ficheiros correspondentes à implementação dos esquemas realizada encontram-se no CD que acompanha este texto.

Esses ficheiros estão organizados em pastas com os mesmos nomes dados aos modelos ao longo da secção 5.2. Cada pasta tem 3 ficheiros '.slx', contendo a implementação do esquema para cada interveniente, e 3 ficheiros '.m', usados para inicializar alguns parâmetros da implementação. Na raiz da diretoria está, também, o ficheiro 'README.txt', com informação mais detalhada sobre a organização da mesma.

2. Segurança na Camada Física

A confidencialidade das comunicações, seja em contexto militar (onde grande parte dos avanços tecnológicos acontecem), pessoal ou profissional, é uma das grandes preocupações dos intervenientes nas mesmas. Desde muito cedo que são conhecidas formas de criptografia para proteger as mensagens de recetores indesejados, como a simplista Cifra de César, em 50 A.C. [7].

A preocupação com a segurança manteve-se e, com o avanço do poder computacional, foram-se desenvolvendo cifras cada vez mais poderosas, tipicamente aplicadas nas camadas superiores da pilha protocolar[1]. Enquanto isso, na camada física, foram desenvolvidos códigos com vista a garantir fiabilidade através da capacidade de correção dos erros introduzidos por um canal ruidoso, sem preocupações de segurança.

2.1. O canal de escuta

Com o aparecimento e evolução das redes sem fios, a proteção dos dados tornou-se um desafio ainda maior, dadas as duas principais características destas redes: difusão e sobreposição de sinais. Se a primeira torna difícil retirar os sinais transmitidos do alcance de recetores ilegítimos, a segunda traz problemas de fiabilidade, no caso de existir um grande número de sinais diferentes sobrepostos. Apesar de se terem estudado códigos de canal para segurança desde os anos 70, só com o emergir destas redes, nos últimos anos, se renovou o interesse nesta abordagem, em parte por esta exigir a existência de uma vantagem do recetor legítimo face a um possível atacante [8] [3]. Todos os trabalhos nesta área de investigação são baseados no artigo de Wyner que, em 1975, provou a existência de códigos que garantissem, ao mesmo tempo, fiabilidade e confidencialidade [2].

O canal de escuta ("*Wiretap Channel*"), apresentado por Wyner e ilustrado na Figura 1, pressupõe a existência de três intervenientes, representando, assim, o modelo mais simples em que pode haver uma tentativa de interceção da informação. São eles: um transmissor, um recetor legítimo e um recetor ilegítimo, também denominados Alice, Bob e Eve (ou *eavesdropper*), respetivamente.

Neste modelo, a Alice pretende enviar uma mensagem M, privada, ao Bob. Para isso, codifica-a, obtendo o sinal[2] X^n que é, depois, enviado, pelo "Canal Principal", para o Bob, sendo que este recebe o sinal Y^n e o descodifica, por forma a obter uma estimativa da mensagem enviada, $\widehat{M}$.

[1] Ver capítulo 2 de [55] para explicação detalhada da arquitetura das redes em camadas.

[2] Usa-se a notação X^x para fazer referência ao sinal X de comprimento x. Para não sobrecarregar a notação, este sobrescrito não é usado quando já foi referido anteriormente o tamanho. Desta forma, X^x e X são o mesmo sinal.

Enquanto isso, a Eve está à escuta do sinal emitido pela Alice e obtém uma cópia de X, Z^n, também ela afetada pelo "Canal do *Eavesdropper*".

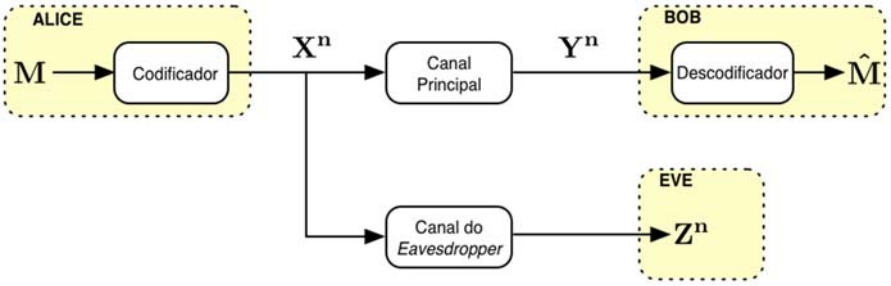

*Figura 1 - O canal de escuta (*The Wiretap Channel*).*

Neste cenário, assume-se, ainda, que o "Canal do *Eavesdropper"* é mais ruidoso que o "Canal Principal" e que o *eavesdropper* é passivo, ou seja, que está à escuta sem se fazer notar nem se fazer passar por outro interveniente. Além disso, considera-se que a Eve possui o mesmo algoritmo descodificador que o Bob e que não possui limitações a nível de poder computacional.

2.2. Métricas de segurança

O conceito de segurança, por si só, é vago. Por isso, quando se avalia a segurança que um esquema consegue providenciar, é necessário que sejam definidas as condições para ele ser considerado seguro ou não. Nesse sentido, Shannon propôs, em 1949, o conceito de segurança perfeita [9], em que se considera seguro um código que garanta que a mensagem codificada não fornece nenhuma informação acerca da mensagem, ou seja, $I(M;X^n) = 0$, i.e., a informação mútua entre M e X^n é nula. No entanto, Shannon concluiu também que, para que se alcance a segurança perfeita, é necessário usar chaves com, pelo menos, o mesmo tamanho da mensagem, o que não é viável [6].

Alguns anos mais tarde, Wyner sugeriu a métrica de segurança conhecida como segurança fraca [2]. Nesta, não se exigia que X^n não possuísse informação sobre a mensagem, mas que o sinal recebido pela Eve, Z^n, revelasse uma pequena quantidade de informação sobre a mensagem. Esta quantidade seria mais pequena quanto maior fosse o comprimento da mensagem, n, ou seja, $\lim_{n\to\infty} \frac{1}{n} I(M;Z^n) = 0$ [6].

Contudo, foi provado que podem ser construídos códigos com problemas de segurança que satisfaçam esta métrica [10]. Por isso, o conceito de segurança forte foi introduzido por Maurer [11], onde se considera seguro um esquema que garanta que a informação mútua entre M e Z^n seja assintoticamente zero, ou seja, $\lim_{n\to\infty} I(M;Z^n) = 0$.

Apesar de garantirem segurança, no que à teoria da informação diz respeito, estas métricas mostraram-se débeis quando aplicadas a canais reais, nalguns casos específicos. Além disso, não são de fácil implementação [6]. Por isso, para este tipo de canais, foram estudadas novas métricas de segurança. Em [12], foi introduzida uma métrica mais prática, que analisa o BER à saída do descodificador e estabelece os valores desejáveis deste, quer para o Bob, quer para a Eve. Encontrando os valores de relação sinal-ruído (SNR – *Signal-to-Noise Ratio*) correspondentes a esses limites, define-se um intervalo de segurança (*security gap*) a partir da diferença, em dB, entre eles. Assim, sabe-se qual a vantagem que o Bob necessita de ter sobre a Eve para se considerar uma transmissão segura. Intuitivamente, percebe-se que o ideal é conseguir um intervalo de segurança o mais pequeno possível, de forma a que se possa operar em segurança em condições em que a diferença de qualidade entre o canal do Bob e da Eve seja mínima.

O uso do BER para medir segurança, apesar de ser simples de implementar, não garante que toda a informação esteja segura, devido ao seu cálculo ser baseado em médias. Para colmatar essa falha, em [6] foi proposta uma nova métrica, BER-CDF, que mede a probabilidade do BER medido em cada palavra de código transmitida/recebida ser superior a um valor próximo de 0.5:

$$Pr\left(\hat{P}_b > 0.5 - \delta\right), \tag{1}$$

onde $\hat{P}_b$ é a proporção de bits errados à saída do descodificador e δ é um valor entre 0 e 0.5, escolhido de acordo com as exigências de segurança pretendidas.

Na análise dos resultados obtidos, será utilizada uma variação do intervalo de segurança, em que se empregará a métrica BER-CDF como limite de segurança e um limite de BER para avaliar a fiabilidade.

2.3. *Jamming* para segurança

Uma vez que o Bob e a Eve possuem os mesmos algoritmos de descodificação, é fundamental explorar a diferença de qualidade entre os seus canais.

Para se obter uma vantagem sobre o canal de escuta, é necessário que este seja pior que o canal principal, seja pela própria natureza dos canais (*e.g.* estar mais distante do transmissor ou ter obstáculos entre eles) ou pela alteração propositada do canal (*e.g.* usar retransmissores, sejam confiáveis [13] ou não [14], para melhorar o sinal do Bob, ou usar *jamming* de forma a degradar o sinal da Eve [4] [1]).

Tradicionalmente, o *jamming* consiste em emitir um sinal aleatório com uma potência elevada, de forma a causar interferências noutros sinais presentes na mesma frequência. Esta é uma técnica de ataque para impedir que um recetor tenha acesso a um sinal que lhe é destinado. Como nem

sempre é possível garantir que o canal da Eve é pior por natureza, recentemente, começou-se a olhar para o *jamming* como uma técnica de segurança, sendo usada para reduzir a SNR do canal do *eavesdropper*. Em [15] e [16], esta ideia foi abordada sob os nomes de ruído artificial e *jamming* cooperativo, respetivamente e, em [1], foram avaliados vários cenários possíveis de uso de *jamming* para segurança, tais como: usar um ou mais *jammers*; *jamming* indiferente às condições do canal; *jamming* só quando o canal do Bob é pior que o da Eve; ou ainda *jamming* apenas quando a qualidade do canal for mais baixa que um determinado limiar.

2.4. Codificação para segurança baseada em códigos curtos e técnicas de *jamming* e *interleaving*

A construção de códigos para o canal *wiretap* que garantam a não existência de fugas de informação tem mostrado ser uma tarefa difícil. Os primeiros códigos práticos foram construídos somente nos últimos anos [17] e apenas para cenários ideais, tais como considerar um tamanho de código a tender para infinito ou um canal principal perfeito, por exemplo, que são condições difíceis de concretizar num contexto prático.

Como estes cenários não permitem alcançar o objetivo principal desta área de investigação, que é a segurança na camada física em sistemas reais, alguns autores dedicaram-se ao estudo de cenários mais realistas, baseadas em códigos LDPC (*Low-Density Parity-Check*) e polares, por exemplo e/ou técnicas de *interleaving*, *scrambling* e/ou *puncturing*[3], que serão abordadas no capítulo 4.2 [12] [18] [19] [20] [21].

Uma dessas abordagens foi proposta por Vilela *et al.* [4] e é um dos esquemas de que se pretende fazer prova de conceito, nesta dissertação, com a sua implementação em plataformas SDR, o que será descrito no Capítulo 5. Este, designado por "*Interleaved Coding for Secrecy*", usa uma combinação de técnicas de *interleaving,* códigos de canal robustos e, ainda, técnicas de *jamming* para segurança. A Figura 2 ilustra o esquema proposto por estes autores. Neste cenário, tal como no modelo do canal *wiretap*, considera-se que a Alice deseja enviar uma mensagem secreta M^m ao Bob, estando a Eve à escuta.

[3] Optou-se por manter as nomenclaturas destas técnicas em inglês para evitar, principalmente, a mistura dos conceitos de *interleaving* e *scrambling*. Quando o contexto for óbvio, poder-se-á usar o adjetivo "baralhada" para denotar uma sequência afetada por *interleaving* ou *scrambling*.
Interleaving pode traduzir-se por embaralhamento e consiste em trocar a posição dos bits segundo uma chave.
Scrambling, apesar de ser frequentemente traduzido como embaralhamento, é a inversão de alguns bits e não uma mera troca de posição.
Puncturing pode ser interpretado como perfuração. Consiste na eliminação de alguns bits, seja de forma sequencial, aleatória ou de acordo com algum tipo de chave.

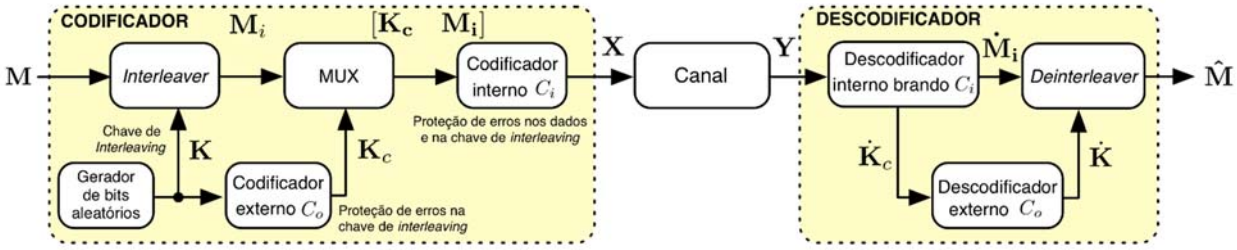

Figura 2 - Interleaved Coding for Secrecy.

No que ao codificador diz respeito, é gerada uma chave de *interleaving* aleatória, K^{k_o}, diferente para cada mensagem M^m, que, após o processo de *interleaving*, dá origem à mensagem baralhada M_i. Dada a sua importância vital na descodificação da mensagem no recetor, a chave, K^{k_o}, é, ainda, codificada com um código externo C_o, de tamanho (n_o, k_o), de onde resulta a chave codificada, $K_c^{n_o}$. A chave codificada, K_c, é concatenada com a mensagem M_i e o bloco concatenado resultante, $[K_c \quad M_i]$, é, finalmente, codificado por um código interno C_i, sendo a sequência codificada resultante enviada pelo canal.

No descodificador, ocorre o processo inverso. Faz-se uma descodificação iterativa com decisão branda (*soft-decision decoding*), de onde se obtém uma estimativa da mensagem baralhada, $\dot{M}_i$, e da chave de *interleaving* codificada, $\dot{K}_c$. Esta é, ainda, descodificada pelo código externo C_o, produzindo uma estimativa da chave, $\dot{K}$, que vai, depois, desembaralhar a mensagem $\dot{M}_i$, obtendo-se uma estimativa da mensagem enviada, $\hat{M}$.

Como a chave de *interleaving* é uma peça fulcral deste esquema, além da proteção oferecida pelo código externo C_o, com vista à correta receção da chave por parte do Bob, é, também, usado um *jammer*, que está ativo apenas durante a transmissão da chave, com o objetivo oposto em relação ao *eavesdropper*, i.e., o de degradar a informação da chave recebida pela Eve. No entanto, para se poder beneficiar deste auxílio, é necessário que o código C_i seja sistemático e que a Alice e o *jammer* estejam perfeitamente sincronizados, para que este esteja ativo unicamente durante a transmissão da chave. O código interno, C_i, deve, ainda, ser suficientemente poderoso para corrigir os erros provocados pelo canal principal, mas não demasiado poderoso para corrigir todos os erros, ou a Eve conseguiria descodificar a mensagem, sendo, por isso, utilizados os códigos LDPC. Já o código externo, C_o, deve ser capaz de corrigir os erros da chave que o interno não consiga, mas não ser capaz de corrigir demasiados erros, devido à possibilidade da Eve também os conseguir corrigir. Por isso, uma boa escolha para este código são os códigos BCH (Bose-Chaudhuri-Hocquenghem), que podem corrigir até um número preciso de erros.

Desta forma, com a escolha de códigos apropriados, garante-se a fiabilidade necessária para que o Bob descodifique a mensagem e a segurança que impeça a Eve de o fazer [4].

2.5. ICS com Chave Escondida

Na sequência do trabalho realizado em [4], foi desenvolvida uma variante deste modelo e apresentada em [5].

Neste modelo, ilustrado na Figura 3, é, de novo, gerada uma chave aleatória K^k, que é usada para proceder ao *interleaving* da mensagem a ser transmitida, M, resultando na mensagem baralhada M_i. De seguida, concatena-se a chave de *interleaving* com M_i e codifica-se $[K \quad M_i]$ com o código C_i, sistemático, de tamanho $(n, k+m)$. De seguida, antes de enviar esta sequência para o canal, são eliminados os bits correspondentes à chave, isto é, apenas são enviados os últimos $n-k$ bits, correspondentes à mensagem baralhada e aos bits de paridade introduzidos pelo código interno C_i. Assim, esconde-se a chave de ambos os recetores (Bob e Eve), encontrando-se a informação sobre a mesma embebida nos bits de paridade. Na prática, esta eliminação da chave de *interleaving* pode ser equiparada, conceptualmente, ao uso de *jamming* com potência infinita sobre a chave.

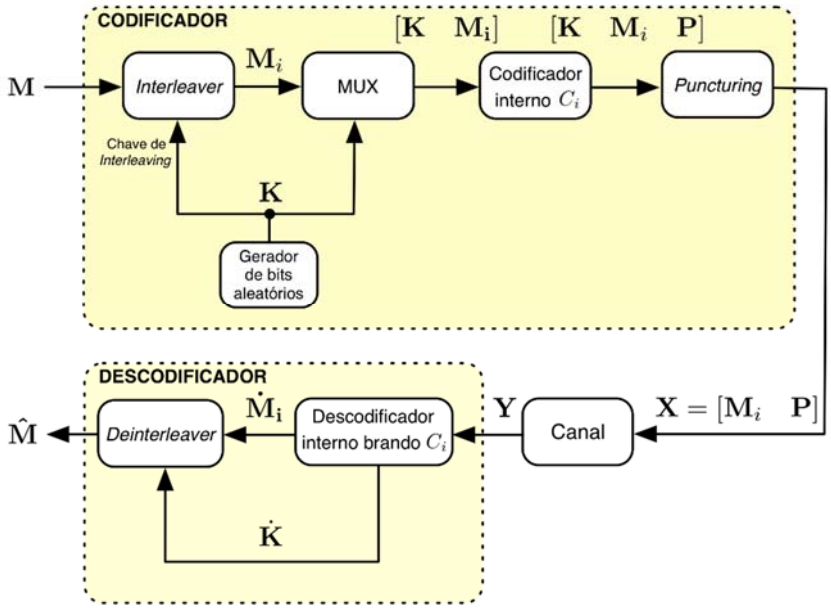

Figura 3 - Interleaved Coding for Secrecy *com chave escondida.*

Do lado do descodificador, faz-se apenas uma descodificação com C_i, como no esquema ICS, obtendo-se uma estimativa da mensagem com *interleaving*, $\dot{M}_i$, e da chave, $\dot{K}$, que é, então, usada no *deinterleaving* de $\dot{M}_i$ obtendo-se uma estimativa da mensagem transmitida, $\widehat{M}$.

2.6. *Scrambled Coding for Secrecy*

O *interleaving* é uma peça crucial nos esquemas apresentados nas duas últimas secções, pois é ele que introduz segurança, através da chave aleatória utilizada para baralhar a mensagem. Este faz com que a descodificação de uma chave de *interleaving* errada possa levar a uma descodificação errada de toda a mensagem. No entanto, esta forma de baralhamento da mensagem

resulta apenas num reposicionamento dos erros obtidos e, por isso, olhou-se para o *scrambler* como uma alternativa a esta técnica. Este consegue que um erro se propague ao longo de vários bits, devido ao efeito de memória criado pelo uso de registos de deslocamento no seu funcionamento, explicado com mais detalhe em 4.2.2.

Assim, durante o desenvolvimento deste trabalho, foram propostas variações aos esquemas ICS e ICS-HK, substituindo o uso do *interleaver* por um *scrambler*, de modo a potenciar a propagação de erros. A estes esquemas, visíveis na Figura 4 e Figura 5, foram atribuídos, intuitivamente, os nomes "*Scrambled Coding for Secrecy*" (SCS) e SCS com chave escondida (SCS-HK – SCS *with a Hidden Key*). Note-se, ainda, que são usadas as condições iniciais dos registos de deslocamento do *scrambler* como chave de baralhamento do mesmo.

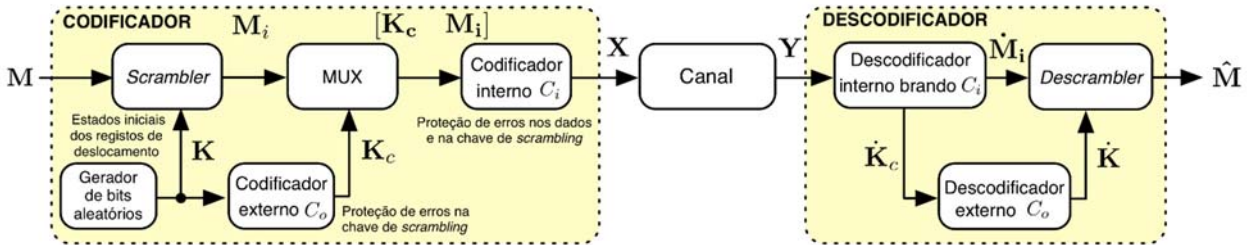

Figura 4 - Scrambled Coding for Secrecy

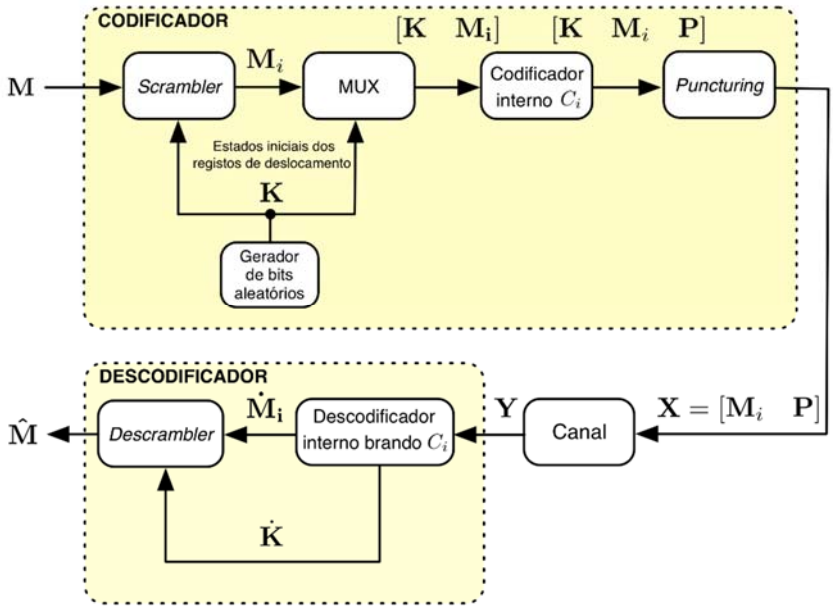

Figura 5 - Scrambled Coding for Secrecy com chave escondida.

Os esquemas ICS e ICS-HK foram avaliados através de simulações. De forma a aferir as suas competências num canal real, bem como as dos esquemas SCS e SCS-HK, impunha-se a sua implementação e avaliação em ambiente real, que será o foco desta dissertação. Para realizar esta prova de conceito, optou-se pela utilização de plataformas SDR, que serão apresentadas no capítulo seguinte.

3. Rádio Definido por Software

Ao longo das últimas décadas, assistiu-se a um avanço na indústria eletrónica, o que conduziu a um grande progresso nos sistemas de microprocessadores, permitindo, assim, que fossem dados passos de gigante em inúmeras áreas. Uma das áreas influenciadas por estes avanços foi o processamento digital de sinal, fazendo com que aparelhos e/ou serviços, hoje em dia de utilização em massa (como, por exemplo, o telemóvel), vissem a luz do dia.

Os SDR [22] são plataformas de comunicação rádio que foram fruto desses avanços. Estas têm a capacidade de oferecer, numa só plataforma, através de *software*, um grande número de funções que são, tipicamente, implementadas em *hardware*, tais como modulação, filtragem, deteção e correção de erros, geração de sinal, entre outros [23].

O facto dos SDR serem plataformas reprogramáveis faz da sua versatilidade o seu grande trunfo face às soluções clássicas para sistemas de comunicação implementadas em *hardware*, permitindo a implementação, num mesmo dispositivo, de diversas normas de comunicação, sem a necessidade de *hardware* dedicado. Assim, o mesmo dispositivo que, num momento, é um rádio FM, pode tornar-se num recetor de GPS em minutos e, logo de seguida, ser uma estação de uma rede GSM privada. Outras duas características merecedoras de destaque das plataformas SDR são o seu baixo custo e a sua portabilidade, uma vez que um transmissor ou recetor completo, com toda a filtragem, modulação e codificação inerentes, pode ter o tamanho de, por exemplo, um *tablet* de 7" [24] ou mesmo de uma *pen* USB [25]. Por fim, estes dispositivos são, ainda, capazes de acompanhar a evolução da tecnologia, uma vez que uma simples atualização de *firmware* pode implementar novas funcionalidades no *hardware* existente [26].

3.1. Funções

Os dispositivos SDR conseguem realizar várias tarefas exigentes em simultâneo, necessárias para implementar o sistema de comunicação desejado. Além disso, são capazes de efetuar quer o processamento digital dos sinais, quer a transmissão e receção dos mesmos.

Por norma, um sistema de comunicação digital consiste numa fonte que pretende transmitir uma informação para um recetor. No caso desta informação ser analógica, deve ser transformada numa representação digital, uma vez que o processamento efetuado nos SDR é exclusivamente digital. Este sinal digital é, posteriormente, analisado pelo transmissor, que o transforma numa onda eletromagnética sinusoidal bem definida em termos de amplitude, frequência e fase, modelando uma (ou mais) destas características com a informação a enviar. Estando a informação embebida nesta onda, torna-se possível enviá-la através de uma antena, pelo ar, para um recetor,

que tem a tarefa de receber essa onda, corrompida pelo canal de transmissão, extrair-lhe a mensagem binária presente e entregá-la ao destinatário na forma pretendida (digital ou analógica).

Na Figura 6, é possível ver um diagrama de blocos de um sistema básico de comunicação digital. O diagrama contém a cadeia de blocos de um transmissor (em cima) e de um recetor (em baixo). Os componentes de cada uma destas cadeias são, essencialmente, os mesmos, mas operam de forma inversa. O sistema é composto por:

- Fonte de informação/Recetor: contém/recebe a mensagem que se quer transmitir/receber. Esta pode ser um simples ficheiro de texto, uma imagem, vídeo, etc.
- Codificador/Descodificador de fonte: remove/reintroduz toda a redundância existente na mensagem.
- Codificador/Descodificador de canal: O codificador introduz redundância nos dados a transmitir, de forma a ser possível corrigir, no recetor, os erros de transmissão
- Modulador/Desmodulador: no transmissor, converte bits em símbolos (conjuntos de bits), que irão modelar uma das características físicas (amplitude, fase ou frequência) do sinal a transmitir. No recetor, converte símbolos em bits.
- Conversor digital-analógico (DAC – *Digital-to-Analog Converter*) e conversor analógico-digital (ADC – *Analog-to-Digital Converter*): é nestes dispositivos que os mundos analógico e digital se encontram. O DAC converte as amostras dos símbolos vindos do modulador em sinais analógicos que possam ser transmitidos, enquanto o ADC realiza a operação inversa.
- Interface RF: transporta os sinais da banda base para uma frequência RF ou desta de volta para a banda base.

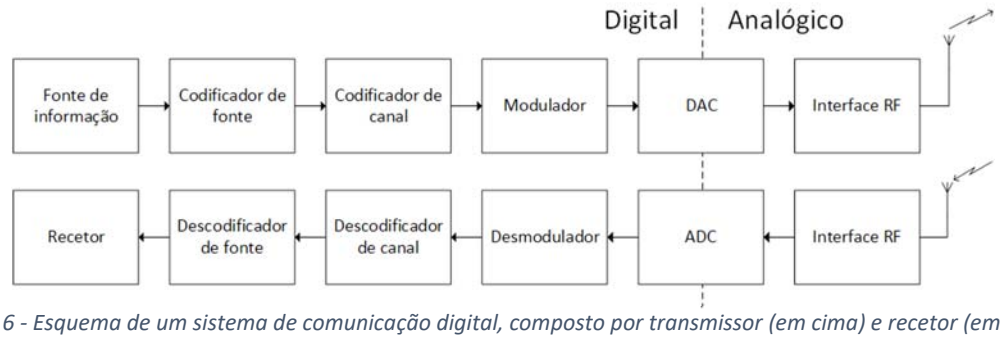

Figura 6 - Esquema de um sistema de comunicação digital, composto por transmissor (em cima) e recetor (em baixo).

As plataformas SDR permitem que quer os codificadores/descodificadores de fonte e canal, quer o modulador/desmodulador sejam componentes configurados em *software*. São, assim, extremamente úteis, pois, a qualquer momento, nalguns casos mesmo quando em funcionamento, é possível, com uma simples alteração em *software*, efetuar alterações ao transcetor, tais como a

dimensão da constelação usada, o nível de proteção do código de canal, ou mesmo permitindo que uma modulação em amplitude passe a ser feita em frequência, ou em fase, ou alterar a largura de banda do sinal transmitido. Além disso, a interface RF é configurável, podendo, assim, escolher-se a frequência, a potência de transmissão ou o fator de interpolação, entre outros [26].

3.2. Limitações

Apesar desta polivalência e aparente simplicidade, o desenvolvimento de *software* para SDR pode ser complexo, pois a análise e manipulação dos sinais requerem elevadas capacidades de processamento, bem como a implementação de métodos de sincronismo e equalização que compensem as perturbações introduzidas pelo meio de transmissão. Os SDR podem ser operados acoplados a um computador, através de uma interface de conexão ao mesmo, tais como *Ethernet* [24] [27], USB [28] [25] ou PCI-Express [29]. As normas usadas por cada uma destas interfaces têm várias versões, que permitem diferentes taxas de transmissão de dados, pelo que, quer a interface, quer a sua versão, se podem traduzir numa limitação à operação da plataforma. O tipo de alimentação também deve ser tido em conta, uma vez que, dependendo da utilização, uma plataforma pode ser alimentada com recurso a USB ou pode precisar de uma alimentação externa dedicada. Por fim, a capacidade de processamento do computador que faz a análise do sinal também deve ser um fator a ter em conta. Uma forma de contornar eventuais limitações de processamento do computador e/ou da interface de ligação pode ser a inclusão de algum processamento no SDR, permitida por alguns modelos [28] [24]. Contudo, esta opção obriga à programação da FPGA neles incluída, para a tornar capaz de fazer o processamento de sinal desejado, o que leva a um maior esforço de programação.

3.3. Ferramentas de desenvolvimento

De forma a programar a componente de *software* necessária para o desenvolvimento dos modelos a implementar, há, essencialmente, três ferramentas, sendo apenas a primeira gratuita:

- *GNU Radio* [30]: é a ferramenta *open-source* de eleição para quem trabalha com SDR e tem a vantagem de ser gratuita. Funciona em conjunto com o *GnuRadio Companion*, que providencia um ambiente gráfico para uma programação por blocos. No entanto, apesar do seu grande potencial, a falta de documentação, que nem sempre consegue ser suprida pela prestável comunidade que a suporta, torna complicada a implementação de sistemas rádio de aparente baixa complexidade. Aliando isto ao facto de a alternativa ser a programação baseada em C++ e Python, o que representa um elevado esforço de programação, pode levar novos utilizadores a desistirem.

- *LabVIEW* [31] : é o ambiente de desenvolvimento gráfico da National Instruments. É mais focado na instrumentação e na automação, mas bastante versátil. Pertence à mesma empresa que controla a Ettus Research [32], empresa que domina o mercado de SDR. Nesta área, tem um conjunto de ferramentas dedicadas à construção de sistemas de comunicação (LabVIEW *Communications System Design Suite*), alguns exemplos de sistemas já construídos e permite, ainda, programar as FPGA.
- *MATLAB/Simulink* [33]: é, provavelmente, uma das linguagens mais usadas na área da Engenharia. Desenvolvida pela MathWorks, é uma ferramenta focada, essencialmente, no cálculo matricial. Permite que programas de cálculo matemático que seriam complexos, se escritos noutras linguagens, sejam muito mais simples. O *Simulink* é um ambiente gráfico, baseado em diagramas de blocos, que permite simular cenários das mais diversas áreas da engenharia e comunicar com *hardware*, como SDR. Tem inúmeros blocos predefinidos, a possibilidade de criar novos blocos em MATLAB e uma vasta documentação, além de uma equipa de suporte dedicada. A *toolbox* de comunicação [34] permite a simulação e desenvolvimento de sistemas de comunicação, incluindo, entre outros, blocos de modulação, codificação de canal ou métricas de desempenho. Esta *toolbox* tem, ainda, um pacote de suporte para SDR [35]. Em conjunto com a *toolbox* de processamento de sinal [36], consegue-se um grande número de funções, blocos e exemplos já implementados para trabalhar. Um outro ponto positivo desta ferramenta é a existência de uma documentação bastante precisa e clara, aliada a exemplos de utilização.

3.4. Modelo utilizado

Na implementação efetuada nesta dissertação, foi utilizada a plataforma SDR USRP B210, da Ettus Research [28]. Esta é uma plataforma de baixo custo e de dimensão média, equipada com uma FPGA Spartan 6 XC6SLX150 e capaz de operar com uma largura de banda de até $56\ MHz$. O processamento pode ser efetuado quer num computador, usando uma ligação por cabo USB 3.0, quer na própria placa, através da programação da FPGA, o que pode permitir maiores velocidades de processamento. Este SDR cobre uma vasta gama de frequências de operação ($70\ MHz - 6GHz$), o que, dependendo das antenas conectadas à SDR através de conectores SMA, permite obter alguma flexibilidade. É, ainda, capaz de operar em MIMO 2×2 *full duplex*, que é uma característica útil para este trabalho, como referido em 5.2.

Como ferramenta de desenvolvimento, por uma questão de familiarização e consequente conforto, foi escolhido o MATLAB/Simulink.

4. Fundamentos para a implementação em SDR de sistemas de comunicação digital

As simulações que acompanham, frequentemente, os trabalhos teóricos são uma ferramenta importante para que se consigam testar exaustivamente os cenários propostos de forma puramente teórica. Esta possibilidade traz benefícios à Ciência, pois, de forma muito mais rápida e barata que a da implementação em ambiente real, consegue-se fazer uma primeira avaliação de uma determinada proposta. No entanto, os testes em ambiente real nem sempre acompanham os resultados teóricos e/ou simulados, devido aos modelos teóricos usados não reproduzirem na perfeição as características dos sistemas em ambiente real. Nesse sentido, as simulações não podem substituir estes testes. Devido a essas mesmas especificidades, a implementação feita em ambiente real é mais complexa, pois está sujeita a dificuldades que não surgem nas simulações.

Este capítulo começa por apresentar a visão geral de um transcetor SDR e, de seguida, são descritas algumas das dificuldades inerentes à implementação de cada um dos esquemas propostos, bem como as bases teóricas que os permitem ultrapassar.

4.1. Visão geral de um sistema Transcetor SDR

Uma comunicação pressupõe sempre dois intervenientes: um transmissor (Tx) e um recetor (Rx). Nesta secção, faz-se uma descrição genérica da sua implementação em plataformas SDR, sendo os diagramas de blocos simplificados de ambos representados na Figura 7.

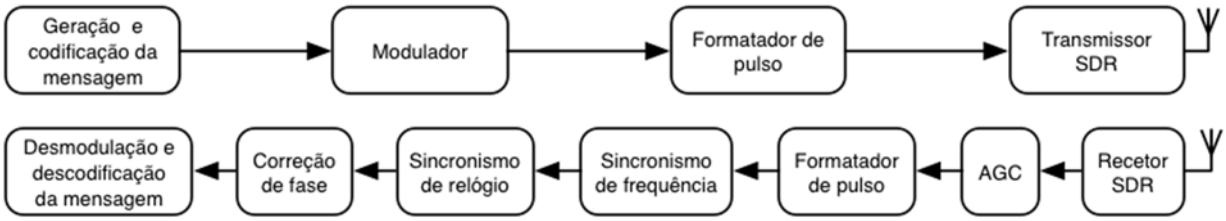

Figura 7 - Percurso de transmissão (cima) e receção (baixo) do sistema de transmissão digital real implementado.

O transmissor é composto por quatro componentes, sendo que, no primeiro bloco, a mensagem a enviar é gerada e codificada. De seguida, passa por um modulador e por um formatador de pulsos, para que seja possível enviá-la pelo bloco "Transmissor SDR". Este efetua a conversão do sinal digital para analógico, a translação do sinal para a frequência de portadora escolhida e amplifica o sinal antes de ser radiado pela antena da placa SDR.

O sistema recetor é consideravelmente mais complexo, pois, após o bloco "Recetor SDR" converter o sinal analógico recebido para sinal digital, tem a missão de identificar e decifrar a mensagem incluída nas amostras recebidas. Isto implica a correção das alterações introduzidas no sinal transmitido pelo canal de transmissão, tais como desvios de frequência e atenuação do sinal,

a estimação do melhor instante de amostragem e a posterior descodificação da mensagem. Mais especificamente, o sinal capturado pela antena passa por um bloco de controlo automático de ganho (AGC – *Automatic Gain Controller*), que ajusta o ganho aplicado ao sinal recebido, de forma a tornar a sua amplitude constante e suficientemente alta para que o sinal possa ser processado. De seguida, o sinal é filtrado por um filtro adaptado correspondente ao filtro formatador de pulso usado no transmissor, que maximiza a relação sinal-ruído (SNR) na receção e minimiza a interferência intersimbólica nos instantes ótimos de amostragem[4]. Segue-se o processo de recuperação de sincronismo de frequência e de estimação do instante ótimo de amostragem (através do sincronismo de relógio). Depois de todo este processo, o sinal é, ainda, alvo de uma correção de fase e, por fim, é desmodulado e descodificado, sendo obtida uma estimativa da mensagem originalmente enviada.

As secções seguintes explanarão, de forma mais detalhada, todos os componentes destes subsistemas, quer de transmissão, quer de receção.

4.2. Codificação de Canal

Por motivos que se prendem, essencialmente, com segurança e/ou robustez dos dados enviados, bem como com a sua deteção do lado do recetor, são, frequentemente, usadas algumas técnicas de codificação que alteram a mensagem a enviar. No que respeita à implementação dos esquemas de segurança da Figura 2 e Figura 3 numa plataforma SDR, foi necessário implementar os mecanismos de *interleaving*, *scrambling, puncturing* e codificação LDPC.

4.2.1. *Interleaving*

No que diz respeito à fiabilidade da transmissão, os erros introduzidos por um canal ruidoso conseguem, em geral, ser bem detetados e corrigidos pelos códigos corretores de erros existentes (LDPC, códigos turbo, códigos convolucionais, etc. [37]), desde que aqueles sejam esporádicos e independentes. No entanto, quando estes erros acontecem em rajadas, como é frequente em canais multipercurso com desvanecimento, muitas vezes os códigos não conseguem dar a resposta desejada.

Para contornar estas limitações, além de terem sido desenvolvidos códigos específicos para corrigir rajadas de erros, podem-se usar técnicas que tornam eficazes os códigos projetados para corrigir erros independentes ou rajadas mais curtas. O *interleaving* é uma dessas técnicas [37] e consiste, essencialmente, numa reordenação, no transmissor, de partes da mensagem codificada,

[4] Para que, posteriormente, seja possível corrigir o erro de relógio e tentar encontrar os instantes ótimos de amostragem, a conversão analógico-digital é feita com sobreamostragem, sendo o sinal subamostrado aquando da correção do erro referido.

que podem ser bits ou conjuntos de bits. Desta forma, no recetor, se a mensagem chegar afetada por uma rajada de erros, estes serão espalhados no processo de *de-interleaving*, resultando em vários erros individuais e/ou rajadas mais curtas [38]. Este procedimento pode ser visualizado de forma mais evidente na Figura 8[5].

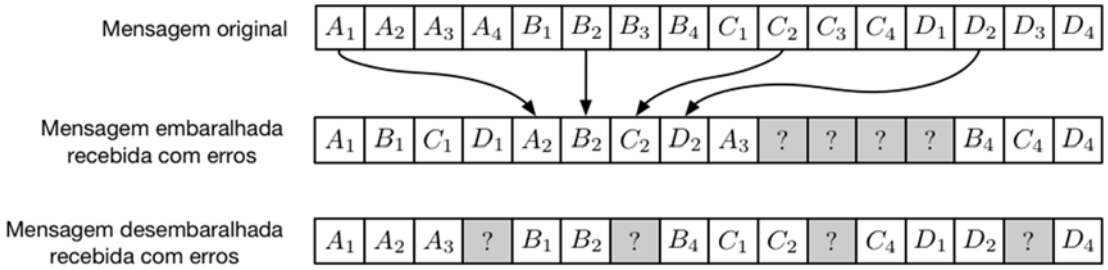

Figura 8 - Ilustração do espalhamento de erros (representados pelos blocos a cinza) através de interleaving.

Apesar desta não ser a sua principal função, neste trabalho, o *interleaving* é usado como elemento de segurança e não de robustez, ao baralhar a mensagem, tal como foi descrito nas secções 2.4 e 2.5. Para isso, é aplicado diretamente à mensagem a ser transmitida, sendo usada, por cada pacote a transmitir, uma chave diferente de *interleaving*, gerada de forma aleatória.

Na Figura 9, que mostra a implementação, em Simulink, desta técnica, no esquema ICS-HK, vê-se que, em cada iteração, é gerada uma chave de inteiros em 'Permutation Vector Generator', que é fornecida ao bloco de *interleaving* de forma a baralhar a mensagem a transmitir, fornecida pelo bloco 'Message Source'.

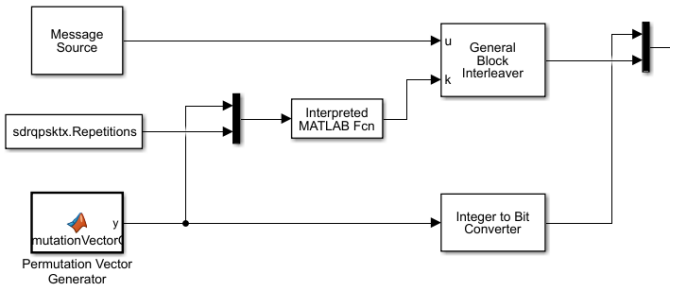

Figura 9 - Implementação em Simulink do interleaving *da mensagem.*

4.2.2. *Scrambling*

Outra técnica, usada em circunstâncias semelhantes às do *interleaving*, é o *scrambling* [39]. Apesar de, na sua essência, serem técnicas bastante distintas, podem ser, por vezes, confundidas, principalmente quando os seus nomes são traduzidos para português, como referido em 2.4. O *scrambling*, por definição, procura eliminar sequências longas de bits iguais que possam causar problemas na sincronização do recetor, aumentando a densidade de transição de bits [39]. Os *scramblers* são construídos com registos de deslocamento com realimentação linear, como apresentado na Figura 10. Nela, observa-se um *scrambler* genérico, com *M* registos, definido por

[5] Para não sobrecarregar a imagem, as setas ilustram a alteração de apenas 4 blocos, sendo as restantes trocas de posição óbvias.

um polinómio $[1\ p_1\ p_2\ \dots p_{M-1}\ p_M]$, que indica a posição dos interruptores que afetam a soma colocada à saída, e com estados iniciais $[k_1\ k_2\ \dots k_{M-1}\ k_M]$. Para tornar esta descrição mais clara, na Figura 11, é mostrado um exemplo concreto de um *scrambler* com 5 registos, definido pelo polinómio [1 0 1 0 1 1] e com estados iniciais [1 0 0 1 0]. Neste caso, à entrada são somados apenas os valores dos registos 2, 4 e 5. Num *scrambler*, a cada iteração, os valores dos registos são deslocados para o registo seguinte, sendo o último descartado. Do lado do recetor, o *descrambler* anula os efeitos introduzidos pelo *scrambler*, sendo a saída o resultado da diferença entre a entrada e a soma dos valores dos registos multiplicados por uma constante binária, p_m, como pode ser visualizado na Figura 12. Importa referir que todas as operações lógicas são realizadas em aritmética módulo-2.

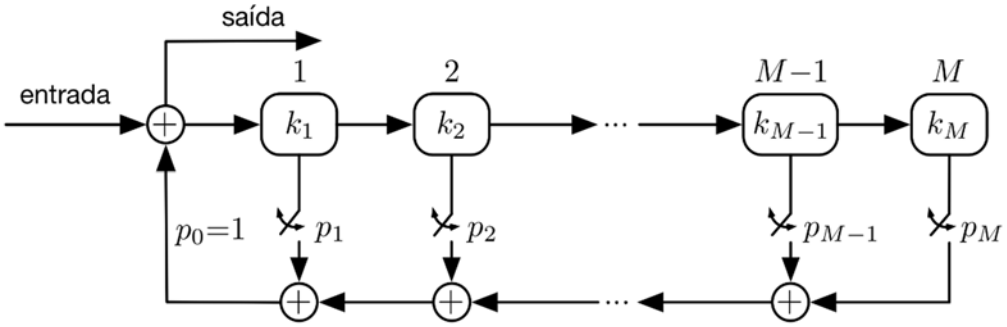

Figura 10 - Scrambler *com M registos.*

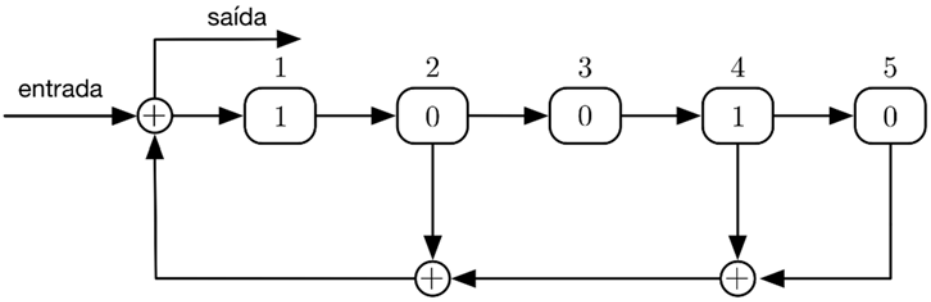

Figura 11 - Exemplo de scrambler *com 5 registos de deslocamento.*

Tal como no caso anterior, neste trabalho, o *scrambler* foi, igualmente, usado para efeitos de segurança, sendo aplicado, também, na mensagem original. Com esta abordagem, pretende-se que, numa transmissão livre de erros (por via de uma descodificação com sucesso do código interno), a recuperação da mensagem seja bem-sucedida; já para uma transmissão com ruído considerável (em que o código interno é incapaz de recuperar todos os erros de transmissão) pretende-se que o *scrambler* potencie o número de erros não corrigidos, uma vez que cada bit à saída do *descrambler* depende dos M bits anteriores. O esquema de segurança pressupõe, pois, a existência de uma vantagem prática de uma melhor SNR do Bob em relação à Eve. Uma vez que o polinómio gerador do *scrambler* tem de ser fixo durante toda a transmissão, são usados os estados iniciais de cada registo como chave, sendo gerados de forma aleatória para cada pacote.

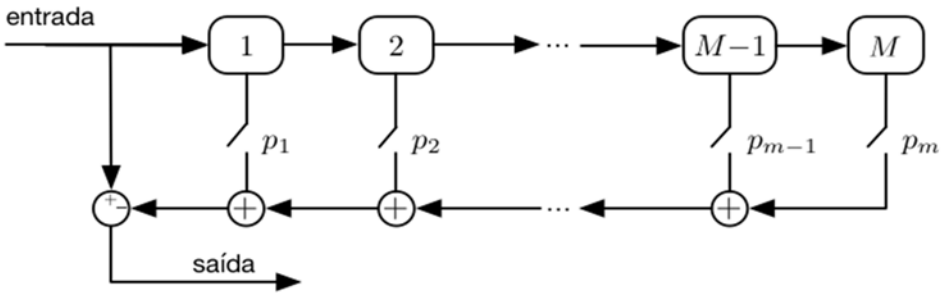

Figura 12 - Descrambler *com M registos.*

Os blocos responsáveis pela implementação desta técnica em Simulink são mostrados na Figura 13, onde se pode ver que, para cada pacote, é gerada uma chave aleatória com os estados iniciais dos registos de deslocamento do *scrambler* e a mensagem é baralhada pelo mesmo.

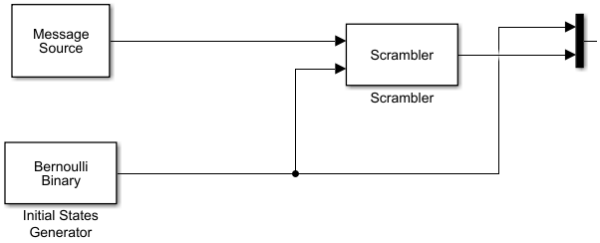

Figura 13 - Implementação em Simulink do scrambling da mensagem.

4.2.3. *Puncturing*

A terceira e última técnica que importa referir é o *puncturing* [40]. Este é usado, geralmente, com o propósito de aumentar a taxa de informação de um código, eliminando alguns bits da mensagem codificada, diminuindo, assim, a redundância da mesma. Neste trabalho, foi usada esta técnica com vista a aumentar a segurança (ver Figura 3), eliminando a parte correspondente à chave, K, da mensagem codificada por C_i. Assim, a informação desta é enviada apenas nos bits de redundância, procurando catapultar os efeitos de um canal com baixa SNR. Este processo, como seria de esperar, tem um outro que contrapõe os seus efeitos do lado do recetor, introduzindo o mesmo número de amostras que foram eliminadas, todas a zero. Para que esta abordagem funcione, é fulcral que, durante a codificação, não se percam os bits correspondentes à chave a ser enviada. Por isso, têm de ser usados códigos sistemáticos, ou seja, códigos em que cada palavra possa ser dividida numa parte correspondente à mensagem e noutra correspondente aos bits de redundância introduzidos. Um exemplo de um código (n, k) sistemático à esquerda encontra-se representado na Figura 14, onde a mensagem codificada terá n bits, sendo, por exemplo, os primeiros k bits de informação e os últimos $n - k$ bits de redundância [37].

Figura 14 - Exemplo de estrutura de um código sistemático.

4.2.4. Códigos LDPC

No que respeita à implementação do bloco de codificação/descodificação interno responsável pela fiabilidade da transmissão (i.e. a correção dos erros de transmissão), optou-se por usar códigos sistemáticos LDPC (*Low Density Parity Check Codes*) devido a possuírem, tipicamente, uma elevada capacidade corretora, o que faz deles códigos populares [41].

Os códigos LDPC, propostos por Gallager [42], são códigos lineares de bloco, cuja matriz de teste de paridade, H, tem uma densidade baixa de uns. Esta contém w_c uns em cada coluna e w_r uns em cada linha, sendo que, para um código (n, k), a matriz H tem dimensões $(n-k)\times n$. É, ainda, necessário que não haja mais que um 1 em comum entre quaisquer duas colunas e, por fim, que $w_c \ll n-k$ e $w_r \ll n$ [37] [43].

Com estas restrições, rígidas ou com o relaxamento de algumas delas, podem-se gerar matrizes esparsas H, através de diferentes algoritmos, sendo o contributo de MacKay [44] digno de registo. No entanto, neste trabalho, optou-se por utilizar uma matriz do standard $802.16e$ [45] para gerar um código $(1536{,}1280)$. Uma vez obtida a matriz H, o processo de codificação passa por a colocar na forma $H = [P^T : I]$, tornando possível a construção da matriz geradora $G = [I : P]$, que permite, então, codificar a mensagem desejada, $\overline{m}$, por

$$\bar{c} = \overline{m}G = [\overline{m} : \overline{m}P] \quad , \tag{2}$$

onde $\bar{c}$ é a mensagem codificada.

Mais complexo é o processo de descodificação destes códigos, que pode ser rígida ou branda e, dentro destas, pode seguir vários algoritmos diferentes, como: *majority-logic* (MLG), *bit-flipping* (BF), BF pesado, probabilidade *a posteriori* (APP – *a posteriori probability*) e algoritmo de soma-produto (SPA – *sum-product algorithm*) [37]. No que diz respeito às descodificações rígidas, a descodificação MLG é a mais simples, mas com grandes taxas de erro, enquanto a BF é mais precisa que esta, mas também mais complexa. Do lado das descodificações brandas, quer a APP, quer a SPA são mais precisas que as rígidas, mas a sua complexidade também é bastante mais elevada. Entre elas, que são, de entre as referidas, as que obtêm os melhores resultados, há uma diferença considerável em termos de complexidade: a SPA é relativamente simples de implementar, enquanto a implementação da APP é de um grau bastante elevado de dificuldade [37]. Neste caso, foi usado o algoritmo SPA e, por isso, será merecedor de uma explicação mais detalhada. No entanto, atendendo a que este algoritmo contém um elevado número de multiplicações, que são pesadas computacionalmente, usa-se, normalmente, uma variação do

mesmo, implementado no domínio logarítmico, onde as multiplicações são convertidas em adições.

O algoritmo SPA é um algoritmo iterativo que calcula os valores das probabilidades *a posteriori* em cada iteração e os utiliza para refinar o cálculo na iteração seguinte. Este processo repete-se até ser atingido um número máximo de iterações definido ou até que as equações de teste de paridade sejam todas verificadas.

Para melhor se perceber o funcionamento do algoritmo SPA, é útil falar primeiro de grafos de Tanner, desenvolvidos por Michael Tanner para representar códigos corretores de erros [46]. Estes são grafos bipartidos, ou seja, com duas secções distintas, em que cada nodo só se encontra ligado a nodos da outra secção. As duas secções são os nodos de bit e os nodos de teste, usualmente representados por círculos e quadrados, respetivamente. O nodo de bit i está ligado ao nodo de teste j se o elemento h_{ji} de H for 1. Assim, para um código de bloco (n, k), haverá n nodos de bit e $n-k$ nodos de teste, estando cada nodo de teste ligado a w_r nodos de bit e cada nodo de bit ligado a w_c nodos de teste, para um código LDPC regular. Na Figura 15, é mostrado um exemplo para um código de Hamming (7,4) com matriz de teste de paridade:

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

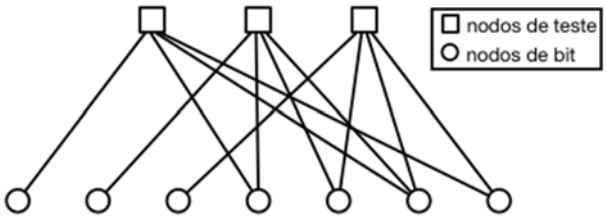

Figura 15 - Grafo de Tanner para código de Hamming (7,4).

O algoritmo SPA usa os grafos de Tanner para passar "mensagens" entre os nodos. Utilizando a mesma notação apresentada na documentação do MATLAB R2016b para o bloco descodificador de LDPC [47], define-se $L(c_i)$ como a máxima verosimilhança logarítmica (LLR – *Log-Likelihood Ratio*):

$$L(c_i) = \ln\left(\frac{Pr(c_i = 0 | y_i)}{Pr(c_i = 1 | y_i)}\right) \quad , \tag{3}$$

sendo c_i o i-ésimo bit da palavra c transmitida e y_i o bit recebido correspondente a c_i.

Definindo, ainda, r_{ji} como a mensagem enviada do nodo de teste j para o nodo de bit i, q_{ij} como a mensagem enviada do nodo de bit i para o nodo de teste j e Q_i como a pseudoprobabilidade *a posteriori*, prova-se, em [43], que as equações de atualização para cada iteração são:

$$L(r_{ji}) = \ln\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right) = 2\tanh^{-1}\left(\prod_{i'\epsilon V_j\setminus i}\tanh\left(\frac{1}{2}L(q_{i'j})\right)\right) \quad , \tag{4}$$

$$L(q_{ij}) = \ln\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right) = L(c_i) + \sum_{j'\epsilon C_i\setminus j} L(r_{j'i}) \quad , \tag{5}$$

$$L(Q_i) = \ln\left(\frac{Q_n(0)}{Q_n(1)}\right) = L(c_i) + \sum_{j'\epsilon C_i} L(r_{j'i}) \quad , \tag{6}$$

sendo C_i e V_j os vetores com os índices dos uns da coluna i e da linha j de H, respetivamente, e o algoritmo inicializado com $L(q_{ij}) = L(c_i)$.

A cada iteração, uma estimativa da LLR *a posteriori* para o bit transmitido c_i é dada pelo valor de $L(Q_i)$, que é o resultado do algoritmo para uma decisão branda. No caso de uma decisão rígida, o algoritmo devolve 1 se $L(Q_i) < 0$ ou 0 caso contrário.

No final de cada iteração, são analisadas as equações de teste de paridade $\hat{c}H^T = 0$, sendo $\hat{c}$ a palavra descodificada e, caso sejam verificadas ou tenha sido atingido um número máximo de iterações, o algoritmo termina a sua execução. Neste trabalho, nos esquemas de segurança e fiabilidade implementados, estipulou-se 10 como o número máximo de iterações. Este foi escolhido de forma empírica, depois de serem realizadas simulações para vários valores, de forma a tentar encontrar um compromisso entre tempo de processamento e desempenho que permita que o computador consiga analisar o sinal em tempo real sem diminuição significativa da taxa de transmissão do sistema.

4.3. Modulação

Os sinais binários não podem ser enviados por um meio físico tal como são. Para que se possam transmitir pelo ar, como desejado, é necessário convertê-los numa onda eletromagnética. Esse processo é chamado de modulação e consiste na alteração de um parâmetro de uma onda portadora, que pode ser amplitude, frequência ou fase.

Neste trabalho, foi usada a modulação QPSK para modular os bits a serem transmitidos. Esta modulação faz uso de duas portadoras sinusoidais, desfasadas de $\frac{\pi}{2}$ radianos, um seno e um cosseno. Cada um destes pode tomar dois valores de fase diferentes, 0 e π, obtendo-se, assim, a possibilidade de modular 4 símbolos diferentes, em que cada um deles representa um conjunto de

2 bits: $00, 01, 10, 11$. Assim, porque seno e cosseno têm um desfasamento de $\frac{\pi}{2}$ entre eles e considerando um desfasamento de $\frac{\pi}{4}$, um símbolo i é modulado como:

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f_c t - (2i-1)\frac{\pi}{4}\right), \quad i = 1,2,3,4 \quad , \tag{7}$$

onde E é a energia de símbolo, T a sua duração e f_c a frequência da portadora. Com uma manipulação simples desta equação, podem-se distinguir duas funções ortogonais (que constituem uma base ortonormada) sobre as quais o sinal é transmitido. São elas:

$$\phi_1(t) = \sqrt{\frac{2}{T}} \cos(2\pi f_c t) \ , \quad 0 \leq t \leq T \quad , \tag{8}$$

$$\phi_2(t) = \sqrt{\frac{2}{T}} \sin(2\pi f_c t) \ , \quad 0 \leq t \leq T \quad . \tag{9}$$

Daqui, resulta que o diagrama de constelação de uma modulação QPSK seja o apresentado na Figura 16, estando os símbolos posicionados em $\left(\pm\sqrt{\frac{E}{2}}, \pm\sqrt{\frac{E}{2}}\right)$ [40].

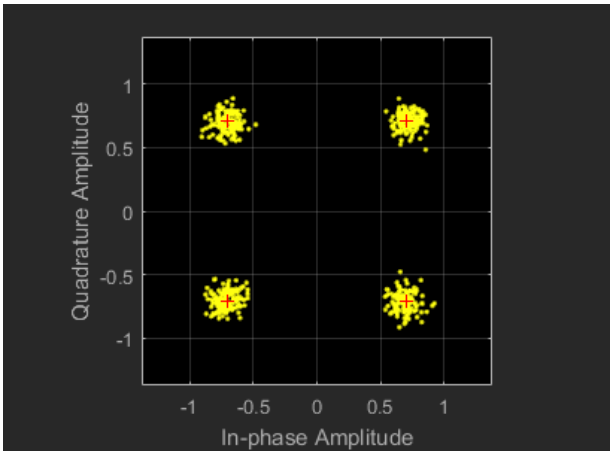

Figura 16 – Exemplo de uma constelação QPSK com ruído (pontos amarelos) e a constelação QPSK ideal ('+' vermelho).

No recetor, são calculadas as LLR, para que possam ser fornecidas ao bloco descodificador de LDPC, que tomam em conta a SNR do canal e a distância do ponto recebido face a cada ponto da constelação.

4.4. Formatação de pulso

O último passo antes de um sinal poder ser transmitido é a formatação de pulso. Apesar de um sinal, em teoria, poder ser enviado sem a fazer, a largura de banda limitada do canal vai deformar o sinal transmitido, causando interferência intersimbólica (ISI), que se traduz num elevado número

de erros na receção. Esta acontece devido ao espalhamento temporal que o canal induz, conduzindo a uma sobreposição de partes de dois pulsos adjacentes, o que altera a forma de onda recebida.

Com vista a contornar esta limitação, foram estudados filtros aplicáveis a um sistema de comunicação que não introduzissem ISI e que garantissem uma largura de banda o mais baixa possível. Para isso, estes filtros devem cumprir o Critério de Nyquist, que estabelece que, para não existir ISI, a resposta a impulso do filtro deve ser

$$h(nT) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}, \tag{10}$$

para todos os inteiros, n, e período de símbolo T. Esta condição garante que, em cada instante de amostragem $t' = n'T$, a amplitude do sinal seja dependente apenas do símbolo transmitido nesse período (i.e. $n'T \leq t < (n'+1)T$), sendo a contribuição dos restantes pulsos igual a zero. Desta forma, a soma dos pulsos recebidos não é afetada pelos pulsos adjacentes.

A melhor abordagem para eliminar a ISI é a utilização de um filtro, H, que tenha uma resposta em frequência retangular:

$$H(f) = \begin{cases} \frac{1}{2B}, & -B < f < B \\ 0, & |f| > B \end{cases}, \tag{11}$$

com $B = 1/2T$ a largura de banda mínima de Nyquist.

No domínio temporal, este filtro corresponde à função *sinc*, definida por:

$$h(t) = \mathrm{sinc}(2Bt) = \frac{\sin(2\pi Bt)}{2\pi Bt} \ . \tag{12}$$

No entanto, este filtro não é realizável, devido à sua não-causalidade e ao seu atraso infinito.

A solução encontrada para este problema é a utilização do pulso de cosseno elevado, com um fator de excesso de banda, β, que pode tomar valores entre 0 e 1. Este define quanta largura de banda é utilizada a mais em relação a B, correspondente ao filtro ideal. Assim, a resposta em frequência deste filtro, que pode ser vista na Figura 17, é composta por uma parte constante e outra sinusoidal [40]:

$$H(f) = \begin{cases} \frac{1}{2B}, & 0 \leq |f| < B(1-\beta) \\ \frac{1}{4B}\left(1 - \sin\left(\frac{\pi(|f| - B)}{2B\beta}\right)\right), & B(1-\beta) \leq |f| < B(1+\beta) \\ 0, & |f| \geq B(1+\beta) \end{cases} . \tag{13}$$

Outro problema que afeta a qualidade de uma transmissão é o ruído que é introduzido pelo canal. Com vista a melhorar a estimação das mensagens transmitidas do lado do recetor, usa-se o conceito de filtro adaptado, que faz uso, no recetor, de um filtro equivalente (versão espelhada e atrasada no tempo) ao filtro usado na transmissão. Isso permite maximizar, no recetor, a SNR do sinal a desmodular [23]. Em suma, o que interessa é que a resposta do conjunto filtro transmissor $H_T(f)$ e filtro recetor $H_R(f)$ seja um filtro cosseno elevado $H_{RC}(f)$, i.e.

$$H_{RC} = H_T(f) \cdot H_R(f) \ . \tag{14}$$

Considerando módulos iguais para ambos,

$$|H_T(f)| = |H_R(f)| = \sqrt{|H_{RC}|} \ , \tag{15}$$

obtendo-se, assim, a resposta dos filtros raiz de cosseno elevado (RRC). No caso deste trabalho, foi usada uma implementação digital dos mesmos, considerado um fator de excesso de banda $\beta = 0.5$ e uma sobreamostragem de 4.

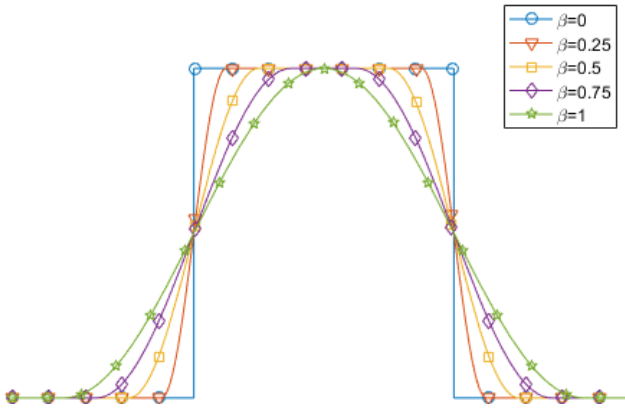

Figura 17 - Resposta em frequência do filtro de cosseno elevado para diferentes valores de β, sendo o filtro ideal $\beta = 0$.

4.5. Controlo de ganho automático

O recetor tem a tarefa mais complicada num sistema de comunicação, uma vez que apenas tem como matéria prima um sinal degradado, do qual se pretende recuperar a informação. Fazem parte do processo de acondicionamento do sinal anterior à sua descodificação algumas funções de recuperação de sincronismo. Para que estas funcionem bem, é fulcral haver um bom nível de sinal e, se possível, o seu ajuste a uma dada amplitude de referência [48] [49].

Uma vez que o nível de atenuação do sinal recebido pode variar ao longo do tempo, a forma de satisfazer essas condições é o uso de um controlador automático de ganho (AGC – *Automatic Gain Controller*) [49]. Este dispositivo, visível na Figura 18, compara a amplitude do sinal à saída com um valor de referência, R, que se deseja obter. Dependendo da relação entre ambos, o sinal de feedback vai aumentar ou diminuir o ganho, conforme necessário. Essa variação do ganho é dada pela constante α, que simboliza o passo com que o ganho é alterado.

Considerando o sinal de entrada amostrado de um AGC, dado por

$$x[n] = A_r[n]\cos(\Omega_0 n + \theta_r[n]) \quad , \tag{16}$$

tem-se interesse em compensar as variações de $A_r[n]$ em torno do valor de referência. Admitindo que o AGC está a trabalhar sobre o sinal amostrado na banda passante, que é o caso do AGC usado nesta implementação, sabe-se que a diferença de amplitudes entre o sinal desejado e o sinal recebido pode ser dada por

$$e_A[n] = R - |g[n]A_r[n]| \quad , \tag{17}$$

Este erro é usado como estimativa para atualizar o ganho da iteração seguinte:

$$g[n] = g[n-1] + \alpha e_A[n] \ . \tag{18}$$

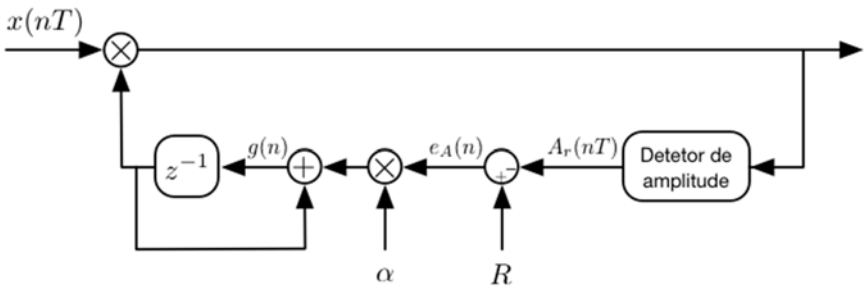

Figura 18 - Diagrama de blocos de um AGC linear.

Um exemplo de funcionamento deste algoritmo pode ser visto na Figura 19. No entanto, é visível, na saída do sistema, que este tem uma resposta muito rápida quando a amplitude é alta, mas, quando esta é baixa, o tempo de convergência é muito elevado. Para contornar esta limitação, usa-se o controlador no domínio logarítmico. Assim, analisando o logaritmo da amplitude do sinal, consegue-se eliminar este problema, resultando no esquema da Figura 20, cujo funcionamento pode ser visto na Figura 21, usando o mesmo exemplo da Figura 19.

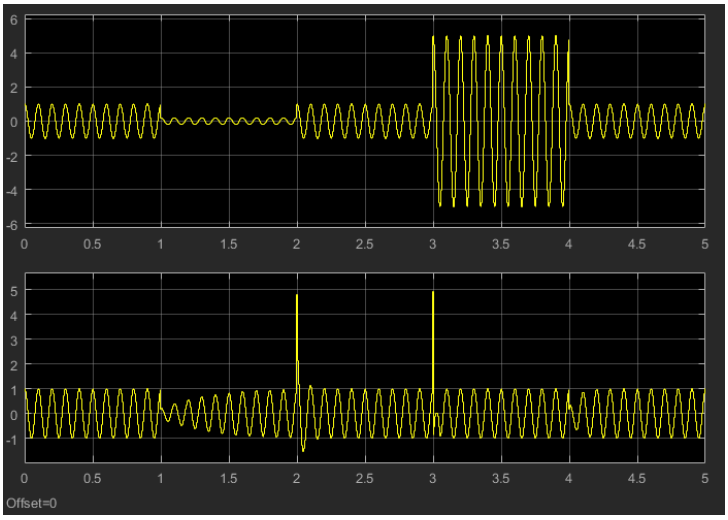

Figura 19 - Exemplo de entrada (cima) e saída (baixo) de um AGC linear para R=1.

O bloco que implementa este esquema no MATLAB tem uma ligeira variação em relação a este, calculando a amplitude do sinal de entrada e não do de saída, conseguindo uma convergência mais rápida [50]. Refira-se, ainda, que o AGC utiliza um detetor de média quadrática para o cálculo da amplitude:

$$A_r[n] = \frac{1}{N} \sum_{m=nN}^{(n+1)N-1} |y[m]|^2 \quad , \tag{19}$$

onde $y[m]$ é a saída do AGC e N o período de atualização.

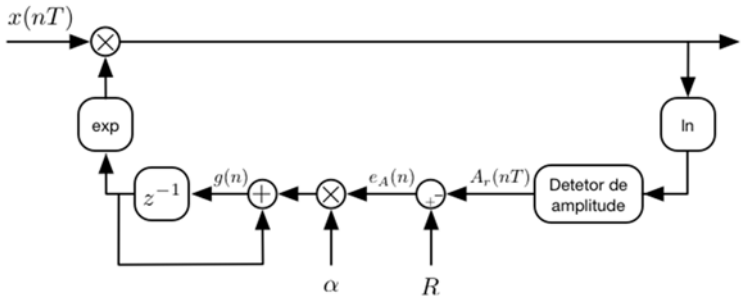

Figura 20 - Diagrama de blocos de um AGC logarítmico.

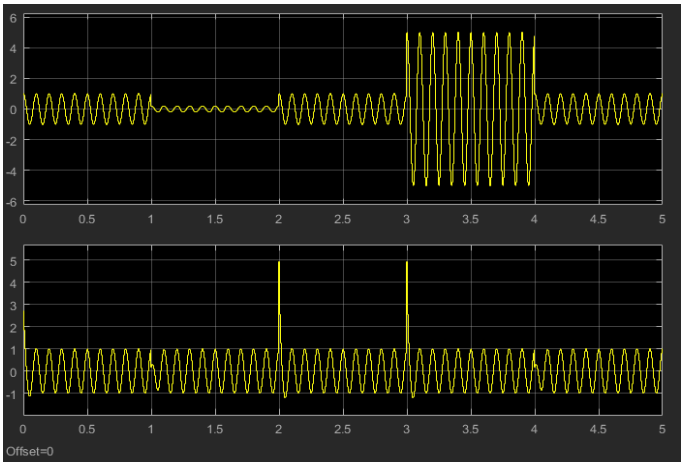

Figura 21 - Exemplo de entrada (cima) e saída (baixo) de um AGC logarítmico para R=1.

4.6. Sincronismo

A descodificação de uma mensagem não pode ser feita com qualidade sem que os sinais transmitido e recebido estejam sincronizados. Isso significa que o recetor tem de saber a frequência e fase da portadora, saber qual o melhor instante de amostragem e, ainda, conseguir identificar a estrutura segundo a qual os dados foram enviados, como, por exemplo, identificar os cabeçalhos dos pacotes de informação.

Como um dos componentes mais utilizados para os diferentes tipos de sincronismo é a *Phase Locked Loop* (PLL), é útil abordar ligeiramente o seu funcionamento antes de especificar os métodos usados em cada tipo de sincronismo.

As PLL são circuitos construídos com vista a sincronizar o sinal de saída da mesma com um sinal de entrada de referência [51]. São compostas, tipicamente, por um detetor de fase, um *loop filter* e um oscilador controlado por tensão (VCO), dispostos em malha fechada, como mostrado na Figura 22.

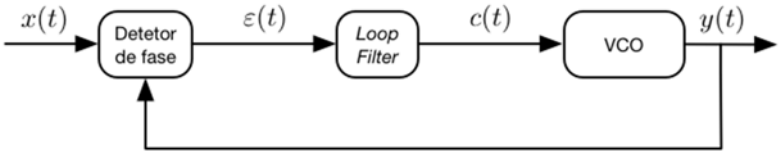

Figura 22 - Diagrama de blocos típico de uma PLL em tempo contínuo.

O detetor de fase tem como objetivo a medição da diferença de fases entre o sinal de entrada da PLL e o sinal de saída da mesma. O erro é, então, fornecido ao *loop filter*, que o transforma num sinal de controlo que, por sua vez, é passado ao VCO, que devolve o sinal corrigido. No VCO, pretende-se que, caso a fase do sinal de saída esteja em atraso/avanço a saída atualizada seja uma versão adiantada/atrasada em fase.

4.6.1. Sincronismo de Portadora

O primeiro passo para se conseguir uma receção acertada dos dados transmitidos é conseguir estimar corretamente a frequência e fase da portadora, pois um erro destas leva a uma rotação da constelação e, se esta rotação for suficientemente grande, pode fazer com que um símbolo transmitido seja descodificado erroneamente.

Nesta implementação, esse processo é feito em dois momentos distintos. No primeiro, faz-se uma estimativa grosseira da frequência, através do método da potência de M [23]. Tomando, como exemplo, uma modulação M-PSK (no caso concreto desta implementação, a modulação é QPSK, $M = 4$) e que a mesma sofreu um desvio de frequência Δf, se se elevar o sinal de entrada, no tempo, a uma potência de M, é criado um tom em $M \times \Delta f$. Desta forma, encontrando o índice do máximo da *Fast Fourier Transform* (FFT) do sinal, idx, torna-se possível encontrar o desvio de frequência, Δf, em Hz:

$$\Delta f = \frac{idx \times F_{len}}{FFT_{len} \times M} \quad , \tag{20}$$

sendo F_{len} o tamanho do sinal analisado e FFT_{len} o tamanho da FFT realizada.

No entanto, dada a importância deste sincronismo, esta correção pode não ser suficiente, pelo que é necessária uma compensação fina da mesma, feita no segundo momento falado anteriormente, com a ajuda de uma *Phase Locked Loop* (PLL).

Para uma modulação QPSK, com pontos de constelação em $(\pm A, \pm A)$, considerando $x(kT_s)$ e $y(kT_s)$, respetivamente, como as projeções dos vetores em fase e em quadratura do sinal à saída

do filtro adaptado e $x'(kT_s)$ e $y'(kT_s)$ como versões rodadas destes, pelo compensador de frequência, o vetor de erro de fase, $e(k)$, que calcula as diferenças entre as fases recebida e transmitida é dado por [49]:

$$e(k) = y'(kT_s)\hat{a}_0(k) - x'(kT_s)\hat{a}_1(k) \ , \tag{21}$$

com

$$\hat{a}_0(k) = A \cdot \mathrm{sgn}\big(x'(kT_s)\big) \ , \tag{22}$$

$$\hat{a}_1(k) = A \cdot \mathrm{sgn}\big(y'(kT_s)\big) \ . \tag{23}$$

De acordo com o funcionamento de uma PLL, o erro é, então, fornecido ao *loop filter*, que consiste num filtro proporcional integrativo (PI), como mostrado na Figura 23.

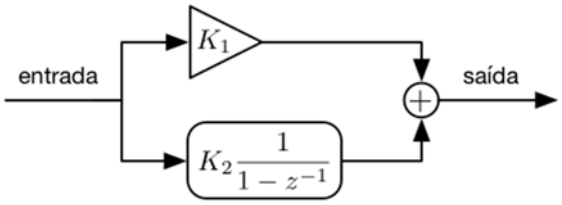

Figura 23 - Estrutura do loop filter. *A entrada é o sinal de erro e a saída corresponde ao sinal de controlo que vai ajustar a entrada do sistema.*

As constantes do filtro podem ser calculadas para uma largura de banda de ruído B_nT_s, um fator de amortecimento ζ e um número de amostras por símbolo N definidos, pelas expressões [49]:

$$K_p = 2KA^2 \ , \tag{24}$$

$$\theta_n = \frac{B_nT_s}{N\left(\zeta + \frac{1}{4\zeta}\right)} \ , \tag{25}$$

$$K_pK_0K_1 = \frac{4\zeta\theta_n}{1 + 2\zeta\theta_n + \theta_n^2} \ , \tag{26}$$

$$K_pK_0K_2 = \frac{4\theta_n^2}{1 + 2\zeta\theta_n + \theta_n^2} \ . \tag{27}$$

4.6.2. Sincronismo de Relógio

Mesmo que se consiga um sincronismo perfeito de fase, isso não é suficiente para descodificar corretamente um sinal transmitido. É preciso, antes de avançar para essa etapa, sincronizar os sinais de relógio do transmissor e do recetor. Uma solução simples para resolver este problema

seria o envio de um sinal de sincronismo junto com o sinal de dados, mas isso não é, de forma nenhuma, eficiente [49]. Por isso, a alternativa é extrair o sinal de relógio a partir do sinal recebido.

Um erro de sincronismo de relógio traduz-se na realização da amostragem num instante de amostragem errado, i.e., amostrando num ponto que não é o de abertura máxima no diagrama de olho, levando a uma dispersão dos símbolos na constelação, como se pode ver na Figura 24.

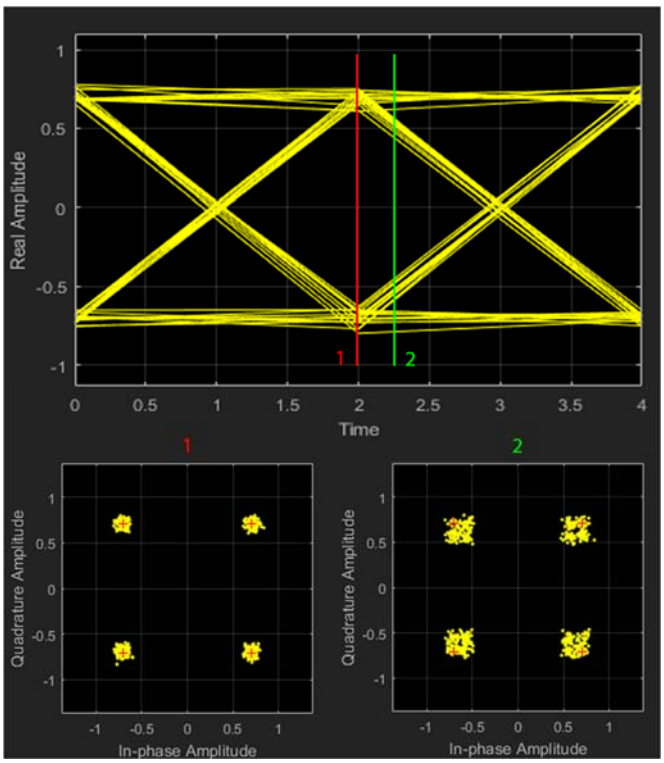

Figura 24 - Diagrama de olho (cima) e constelações (baixo) de um sinal modulado em QPSK, ilustrando o efeito de espalhamento devido a uma escolha errada do instante de amostragem. (1) corresponde ao instante ótimo de amostragem e (2) a um desvio.

A apresentação do sincronismo de relógio partilha algumas semelhanças com a do sincronismo de fase. É, também, feita com recurso a uma PLL e é composta por um detetor de erro de relógio (TED – *Timing Error Detector*), por um *loop filter* e por um relógio controlado por tensão (VCC – *Voltage Controlled Clock*). O funcionamento é semelhante, com o detetor de erro de relógio a calcular a diferença de relógio entre o sinal recebido e o da saída do VCC, que é então filtrado pelo *loop filter* e ajusta o sinal à saída do VCC.

Uma primeira abordagem poderia passar por ter um circuito de sincronismo que controlasse a amostragem feita no ADC, como mostrado na Figura 25. No entanto, por o circuito de sincronismo operar no domínio digital e o ADC operar no domínio analógico, seria necessário um outro conversor, um DAC, para transformar o sinal de controlo do VCC (digital) para o domínio analógico em que opera o ADC. Isso introduziria atraso no sistema e ruído no sinal. Além disso, como o processo de ADC é feito nas SDR, essa abordagem é, também, impraticável. Por isso, como alternativa, usa-se um interpolador à saída do filtro adaptado para deslocar as amostras para

os tempos de relógio pretendidos, de acordo com o esquema da Figura 26 [49]. Para que essa correção aconteça, é necessário, primeiro, calcular qual o erro a colmatar.

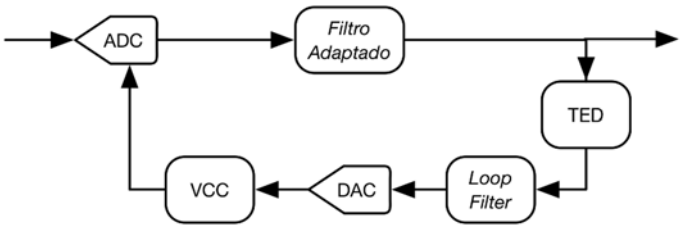

Figura 25 - Esquema de sincronismo de relógio através de ajuste de relógio.

Há várias formas de calcular o erro de relógio, como o TED de máxima verosimilhança (MLTED – *Maximum Likelihood Timing Error Detector*), que analisa o declive do diagrama de olho, o TED *early-late* (ELTED – *Early-Late Timing Error Detector*), que é uma variação do MLTED, o TED de passagens por zero (ZCTED – *Zero-Crossing Timing Error Detector*) ou o TED de Gardner (GTED – Gardner *Timing Error Detector*), onde se procuram as passagens por zero do diagrama de olho, ou ainda o TED de Mueller e Müller (MMTED – Mueller *and* Müller *Timing Error Detector*) [49].

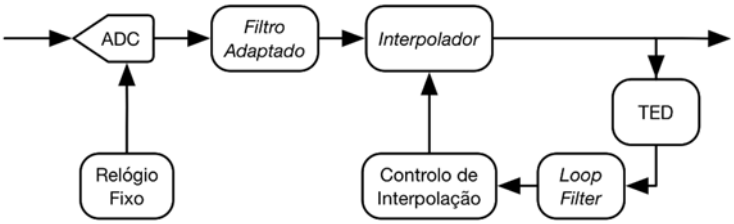

Figura 26 – Esquema de sincronismo de relógio através de interpolação.

Os modelos implementados usam o ZCTED, que faz uso dos sinais de saída do filtro RRC à taxa de 2 amostras/símbolo. Desta forma, assumindo sincronismo ideal, cada símbolo será amostrado no ponto de abertura máxima do diagrama de olho e no ponto em que este cruza o eixo.

A Figura 27 mostra um exemplo do funcionamento deste detetor. O erro de relógio, numa modulação QPSK, sendo $x(kT_s)$ e $y(kT_s)$, respetivamente, as projeções dos vetores em fase e em quadratura do sinal à saída do filtro adaptado, é dado por [49]:

$$e(k) = x\left(\left(k - {}^{1}/_{2}\right) T_s + \hat{\tau}\right)\left(\hat{a}_0(k-1) - \hat{a}_0(k)\right) + \\ + y\left(\left(k - {}^{1}/_{2}\right) T_s + \hat{\tau}\right)\left(\hat{a}_1(k-1) - \hat{a}_1(k)\right) \quad , \tag{28}$$

com $\hat{\tau}$ a indicar a estimativa do atraso de relógio e

$$\hat{a}_0(k) = A \times \mathrm{sgn}\left(x(kT_s + \hat{\tau})\right) \quad , \tag{29}$$

$$\hat{a}_1(k) = A \times \text{sgn}\big(y(kT_s + \hat{\tau})\big) \quad . \tag{30}$$

À exceção de K_p, todas as constantes do *loop filter* são calculadas da mesma forma que em 4.6.1. Quanto a K_p, para modulação PAM binária e $\beta = 0.5$ toma o valor de, aproximadamente, 2.68 [49] e, portanto, para o QPSK desejado, $K_p \approx 2 \times 2.68$, uma vez que este pode ser interpretado como duas modulações PAM binárias ortogonais.

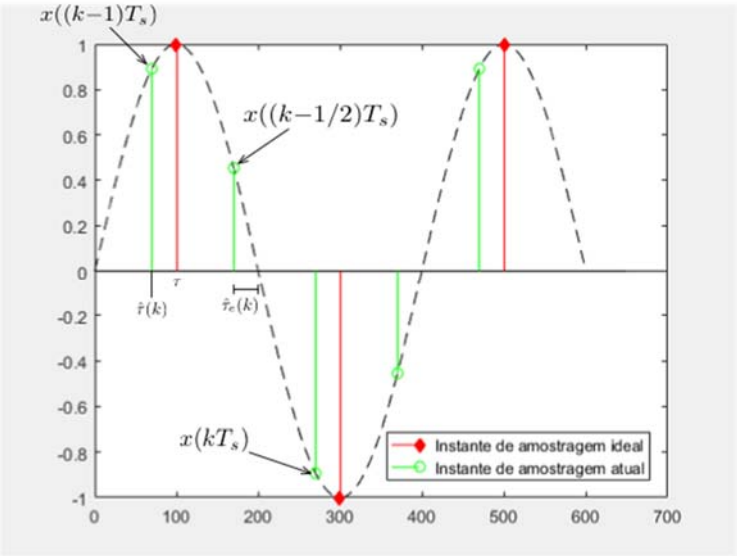

Figura 27 - Funcionamento de um ZCTED.

Antes do interpolador, há ainda um elemento do sistema sincronizador de relógio que transforma o sinal à saída do *loop filter* num sinal de controlo para o interpolador. Dois dos métodos de implementação do controlo de interpolação são o controlo recursivo de interpolação e o controlo de interpolação de contador de módulo 1 [49]. Enquanto o primeiro, como indica o nome, se baseia num processo recursivo, onde cada iteração é calculada com base na iteração anterior, o segundo usa um contador de módulo 1, que provoca *underflow* a cada N amostras. Este foi o controlador utilizado e, antes de explicar o seu funcionamento, importa definir algumas variáveis e conceitos. Para o sinal amostrado $x(nT_s)$, em que T_s é o período de amostragem do sinal recebido, considera-se um sinal interpolado, $y(kT_i)$, onde T_i é o período do sinal interpolado a partir de $x(nT_s)$. Dá-se o nome de k-ésimo interpolante à k-ésima amostra do sinal $y(kT_i)$. O índice n para o qual o k-ésimo interpolante se situa entre $x(nT_s)$ e $x((n+1)T_s)$ designa-se por k-ésimo índice de base e é representado por $m(k)$. Por fim, chama-se k-ésimo intervalo fracional, $\mu(k)$, à fração de T_s entre $m(k)T_s$ e kT_i. Estas variáveis podem ser expressas por [49] [52]:

$$m(k) = \left\lfloor \frac{kT_i}{T_s} \right\rfloor \quad , \tag{31}$$

$$\mu(k) = \frac{kT_i}{T_s} - m(k) \ . \tag{32}$$

Note-se que $0 \leq \mu(k) < 1$. A Figura 28 ajuda a explanar estes conceitos.

Voltando ao controlador do interpolador, este é o responsável por calcular os valores de $\mu(k)$ e o sinal que deteta os *underflows*.

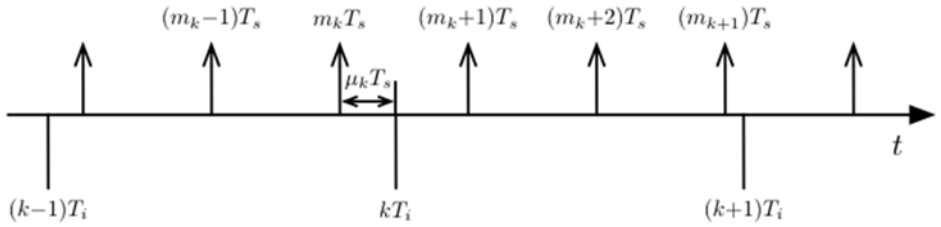

Figura 28 - Relação entre períodos de amostragem real e interpolado.

Como se pode ver na Figura 29, ao sinal de saída do *loop filter*, $v(n)$, é somado o valor $1/N$, sendo o resultado, $W(n)$, que estima a relação T_i/T_s, subtraído ao sinal da iteração anterior, $\eta(n)$. Assim [49],

$$\eta(n+1) = \big(\eta(n) - W(n)\big) \ mod \ 1 \ \ . \tag{33}$$

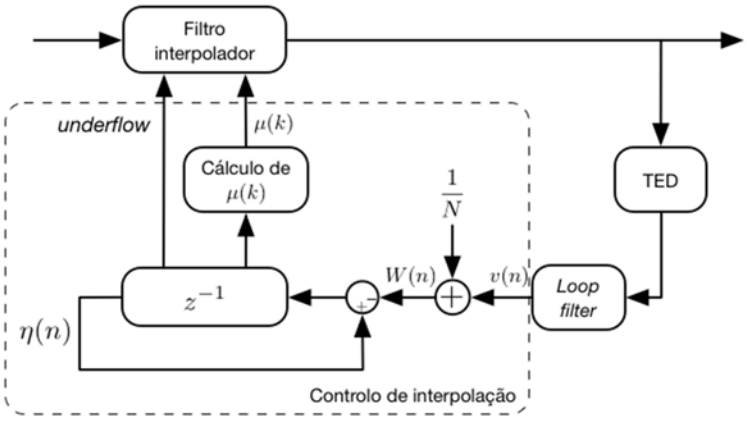

Figura 29 - Diagrama de blocos da implementação do interpolador.

Se um sinal estiver sobreamostrado por N, o decremento de $1/N$ a cada iteração vai fazer com que ocorra um *underflow*, aproximadamente, a cada N amostras, como ilustrado na Figura 30. O sinal de relógio é construído a partir dos *underflows* observados e o cálculo dos intervalos fracionais dado por [49]:

$$\mu\big(m(k)\big) = \frac{\eta\big(m(k)\big)}{W\big(m(k)\big)} \ \ . \tag{34}$$

Por fim, o interpolador recebe os valores de $\mu(k)$ e o sinal que indica as amostras em que ocorreu *underflow*. No âmbito deste trabalho, foi usado o filtro interpolador parabólico por partes com $\alpha = {}^{1}/_{2}$ com a estrutura de Farrow definida na Figura 31 e com a seguinte expressão:

$$x\Big(\big(m(k)+\mu(k)\big)T_s\Big) = v(0) + \mu(k)\big(v(1) + \mu(k)v(2)\big) \quad . \tag{35}$$

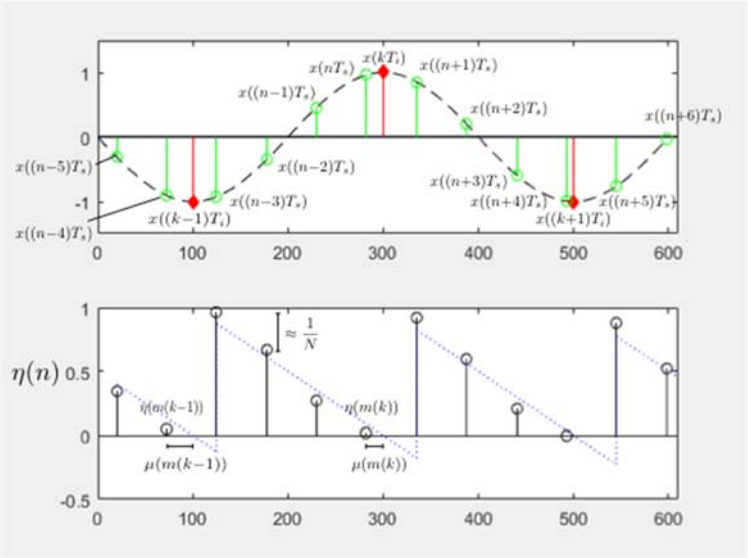

Figura 30 - Funcionamento do controlo de interpolação de módulo 1.

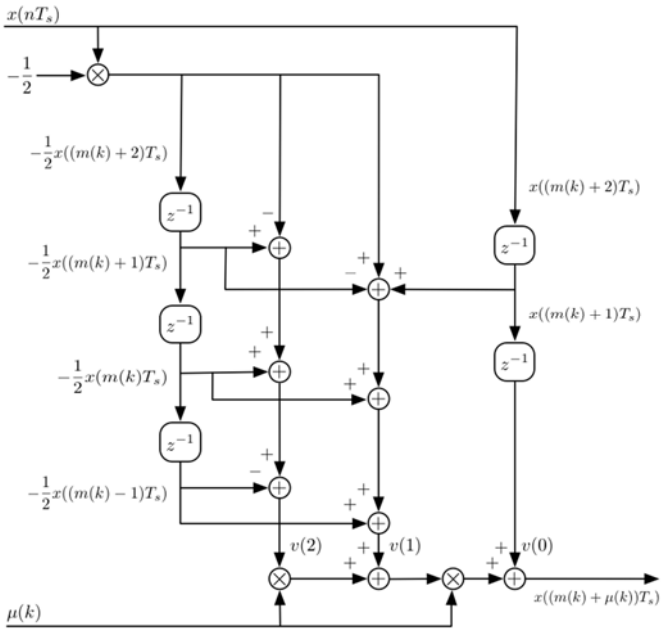

Figura 31 – Diagrama do filtro interpolador.

4.6.3. Sincronismo de *Frame*

Um funcionamento correto dos sincronismos de fase e relógio permite, finalmente, alinhar os bits enviados com os recebidos, possibilitando uma recuperação correta da sequência enviada. No entanto, é necessário distinguir, na sequência recebida, o que corresponde à mensagem e o que

corresponde a dados aleatórios antes ou depois da transmissão da mensagem, bem como onde começa cada pacote de bits, para que seja possível identificar possíveis cabeçalhos, descodificar os dados e reconstruir a mensagem.

A forma de solucionar este problema é anexar um prefixo conhecido à mensagem que possa ser identificado de forma clara no recetor. Como o ruído introduzido pelo canal pode inverter alguns bits, em canais ruidosos pode-se tornar impossível recuperar o início de cada pacote se forem analisados apenas os bits do prefixo. Por isso, uma boa alternativa é analisar a correlação cruzada entre o sinal e o prefixo enviado, sabido de antemão pelo recetor. Este método é mais resistente ao ruído e permite identificar as fronteiras de cada pacote mesmo em canais com ruído elevado. No entanto, para que este método funcione, é importante escolher uma sequência com autocorrelação baixa em todos os pontos, exceto no central, onde deve existir um valor elevado.

Um bom exemplo que cumpre estes critérios são as sequências de pseudorruído (PN), que consistem em sequências semelhante a ruído [39]. Um subconjunto destas foi proposta em 1953 por Barker e dá pelo nome de Códigos de Barker [53]. Nestes, no caso de um código de tamanho n, o pico de autocorrelação toma o valor $c(0) = n$ e os restantes valores são mínimos, idealmente, $|c(k)| \leq 1\,,\ \forall\, k \neq 0$, como se pode ver na Figura 32. São conhecidas, de momento, apenas 8 destas sequências com tamanhos 2, 3, 4, 5, 7, 11 e 13 [53].

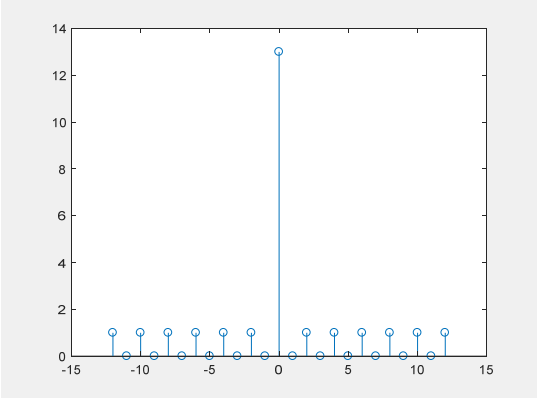

Figura 32 - Autocorrelação da sequência de Barker de tamanho 13.

Neste trabalho, optou-se pelo uso do código de Barker de tamanho 13:

$$+1 \quad +1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1\ ,$$

por ser o de maior tamanho conhecido. Optou-se, ainda, por duplicar o mesmo, resultando num prefixo de tamanho 26 adicionado a cada pacote de dados.

Este prefixo ajuda ainda a que, no recetor, seja corrigida a ambiguidade de fase deixada pelo sincronismo de fase. Para o caso da modulação QPSK usada, a PLL pode bloquear quando o sinal

estiver desfasado da portadora de $\frac{k\pi}{2}$, $k = 0,1,2,3$. Assim, comparando o prefixo recebido com o que se sabe ter sido enviado, consegue-se corrigir esta ambiguidade.

5. Implementação dos modelos propostos

Uma vez analisados os principais blocos componentes de um sistema de comunicação digital, podem, finalmente, ser combinados, de forma a implementar os esquemas de codificação para segurança propostos no capítulo 2. Ao longo deste capítulo, descrever-se-á a implementação de cada um desses esquemas, bem como os cenários dos testes, as condições em que os mesmos foram realizados e os equipamentos utilizados para o efeito.

5.1. Geração da mensagem

O primeiro passo para que haja uma comunicação é haver a necessidade de enviar uma mensagem de uma pessoa/máquina para outra. Neste trabalho, em concreto, a mensagem a ser enviada consistiu numa sequência pseudo-aleatória com 10^5 bits, obtida com recurso à função 'randi' do MATLAB, que gera inteiros pseudoaleatórios com uma distribuição uniforme [54]. A sequência gerada foi gravada num ficheiro, para que fosse usada a mesma mensagem em todas as experiências realizadas.

A utilização de codificação de canal na mensagem, abordada na secção 4.2, exige que esta seja dividida em blocos de comprimento adaptado ao código utilizado. No entanto, caso o tamanho do ficheiro não seja um múltiplo inteiro do tamanho de cada bloco, são adicionados alguns bits à mensagem inicial, de forma a completar o tamanho do último pacote. Com a mensagem a transmitir adaptada ao código são selecionados, em cada iteração, os bits a enviar nesse pacote.

5.2. Modelos implementados

Ao longo do capítulo 2, foram apresentados e descritos dois modelos de codificação para segurança: o ICS e o ICS com chave escondida (ICS-HK). Durante o desenvolvimento deste trabalho, foram propostas, ainda, duas ligeiras variações a ambos, SCS e SCS-HK, substituindo o uso do *interleaver* por um *scrambler*, de modo a aproveitar a propagação de erros referida na secção 4.2. A implementação destes quatro esquemas vai ser apresentada nas próximas secções, bem como a de um esquema de referência, para efeitos de comparação.

5.2.1. ICS

Este foi o primeiro modelo implementado e, como referido, baseia-se no *interleaving* da mensagem segundo uma chave gerada aleatoriamente que é, posteriormente, enviada junto com a mensagem. Durante a transmissão da chave é ativado um *jammer*, com vista a degradar o sinal para que a Eve não consiga obter a chave de *interleaving* sem erros.

Transmissor ICS

O transmissor implementado na plataforma Matlab/Simulink+SDR encontra-se representado na Figura 33.

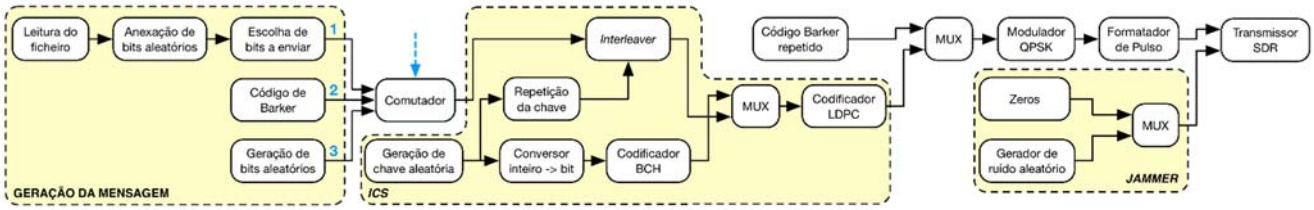

Figura 33 - Diagrama de blocos do transmissor ICS.
NOTA: A azul representam-se as posições e o sinal de controlo do comutador.

Considerando, para já, o instante inicial de transmissão como o momento em que a mensagem começa a ser transmitida (ou seja, quando o comutador, identificado na Figura 33, está na posição 1 – as outras posições serão explicadas em 5.7), o funcionamento do transmissor consiste em: o ficheiro a enviar é lido e, a este, é anexado o número de bits aleatórios necessário para que o tamanho perfaça um múltiplo do tamanho dos pacotes (ditado pelas dimensões do código usado). Em cada iteração, são selecionados os bits a enviar em cada pacote e estes são passados ao *interleaver*. Este, além da mensagem, recebe, também, a chave da permutação a aplicar à mensagem, composta por números inteiros, únicos e de comprimento igual ao da mensagem, com a posição de cada bit. Esta chave é gerada aleatoriamente em cada iteração para um conjunto de posições e cada chave é repetida as vezes necessárias para perfazer o tamanho da mensagem. Por exemplo, para uma mensagem de 12 bits, se se gerar uma chave de 3 inteiros, $[3, 1, 2]$, a sua relação de posições seria repetida 4 vezes, adicionando o tamanho da chave, 3, a cada repetição. Deste modo, a chave da mensagem completa seria $[3, 1, 2, 6, 4, 5, 9, 7, 8,12,10,11]$. Paralelamente, a chave é convertida em bits, codificada com um código BCH e anexada ao início da mensagem baralhada. De seguida, o codificador LDPC adiciona redundância e, ao início da mensagem codificada, são anexados os bits correspondentes a dois códigos de Barker de tamanho 13, para efeitos de sincronismo[6]. Por fim, o resultado é modulado em QPSK e filtrado por um filtro RRC. A mensagem codificada e modulada é, então, transmitida para o SDR. Em paralelo, é gerado ruído aleatório de tamanho igual ao número de amostras correspondente à chave do *interleaver* codificada e é enviado para outro canal do SDR, aproveitando as suas capacidades MIMO para fazer *jamming*.

Neste esquema em concreto, o código LDPC utilizado é de tamanho $(1536,1280)$. Logo, considerando uma chave codificada com tamanho 255 bits, cada pacote comporta $1280 - 255 = 1025$ bits de informação. Como o ficheiro a enviar é composto por 100000 bits, são adicionados

[6] São usados 2 códigos de Barker para facilitar o sincronismo de *frame*.

a este 450 bits, perfazendo um total de $\left\lceil \frac{100000}{1025} \right\rceil \times 1025 = 100450$ bits. Para o tamanho da chave em inteiros, K_{len}, e um código BCH $(255, k)$, é necessário que o tamanho desta seja, ao mesmo tempo, divisor de 1025 e k, sendo que o resultado de $\frac{k}{K_{len}}$, correspondente ao número de bits por cada inteiro, deve ser suficiente para que seja possível representar o maior inteiro da chave. Na ausência de um múltiplo de ambos, adicionam-se zeros à esquerda da chave codificada para completar o tamanho k.

Recetor ICS

O recetor é construído como mostrado na Figura 34.

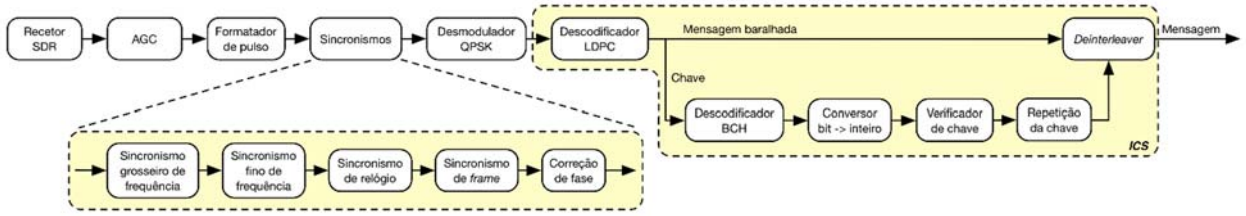

Figura 34 – Diagrama de blocos do recetor ICS.

O sinal capturado pela antena recetora do SDR é amostrado, o AGC ajusta a sua amplitude e é-lhe aplicado o filtro adaptado RRC. Então, dá-se o sincronismo de frequência, relógio, *frame* e fase, descritos na secção 4.6, o sinal é desmodulado e descodificado. Aí, separam-se as sequências obtidas em dois caminhos: um transporta a mensagem até ao *deinterleaver* e o outro descodifica a chave desta. Este processo engloba um descodificador BCH, um conversor de bits para números inteiros e ainda um bloco que verifica se a chave descodificada é uma chave possível. Uma vez que o *deinterleaver* tem de receber uma mensagem e uma chave de permutação composta por inteiros únicos e compreendidos entre 1 e o tamanho da mensagem, a chave que for descodificada só pode ser uma possível chave da mensagem se cumprir estes requisitos. Por esse motivo, o bloco de verificação da chave analisa a chave candidata nestes parâmetros e, se não forem cumpridos, cria uma chave aleatória para tentar adivinhar a chave real. Este processo de adivinhação pretende simular o comportamento de um *eavesdropper*, que, não conseguindo receber uma chave correta, tenta adivinhá-la.

5.2.2. SCS

Este modelo é semelhante ao ICS, variando apenas a forma de baralhamento da mensagem, que usa um *scrambler* no lugar do *interleaver*. A chave gerada aleatoriamente e enviada junto com a mensagem, como referido anteriormente, consiste nas condições iniciais dos registos de deslocamento usados no *scrambler*. Durante a transmissão da chave é, também, ativado um *jammer*.

Transmissor SCS

A implementação do transmissor do SCS pode ser vista na Figura 35.

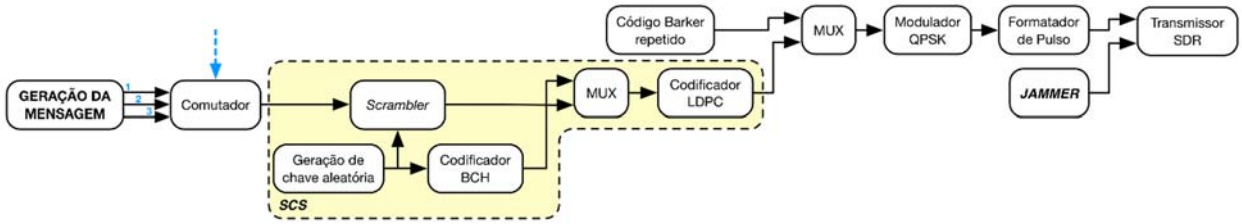

Figura 35 - Diagrama de blocos do transmissor SCS.
A azul representam-se as posições e o sinal de controlo do comutador. A negrito e maiúsculas são representados os subsistemas com o respetivo nome visíveis na Figura 33.

À exceção do conjunto de blocos denominado por SCS, o funcionamento deste transmissor é igual ao transmissor ICS. Neste conjunto, a geração da chave é feita através da função 'Bernoulli Binary Generator' do Simulink, que gera números binários aleatórios seguindo uma distribuição de Bernoulli com probabilidade de 0 igual a p e probabilidade de 1 igual a $1 - p$. Ora, definindo $p = 0.5$, tem-se uma distribuição uniforme de 0 e 1, usada para gerar uma chave aleatória de um tamanho especificado que estabeleça os estados iniciais de cada registo usado no *scrambler*.

Recetor SCS

A Figura 36 ilustra a forma como o recetor SCS é construído.

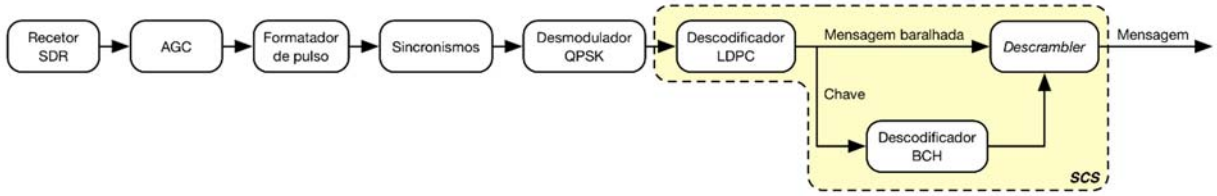

Figura 36 - Diagrama de blocos do recetor SCS.

Tal como no caso do transmissor, o recetor SCS funciona da mesma forma que o recetor ICS, à exceção dos blocos a sombreado. Nestes, os primeiros bits da mensagem à saída do descodificador LDPC seguem o caminho da chave do *scrambler*, enquanto os últimos seguem diretamente para este. A chave deste modelo passa apenas por um descodificador BCH equivalente ao codificador usado no transmissor. Neste caso, ao contrário do método de *interleaving*, a chave descodificada é sempre uma chave possível, pelo que não há necessidade de proceder à verificação da mesma.

5.2.3. ICS-HK

Como visto no capítulo 2, este modelo surge como uma variação do modelo ICS, em que se assume uma vantagem do canal do Bob em relação ao da Eve e se dispensa o uso de um *jammer*, pois a chave é eliminada da mensagem codificada por perfuração do código sistemático. Em

termos práticos, esta supressão pode ser interpretada como um *jammer* com potência infinita a atuar na chave, maximizando a equivocação do recetor.

Transmissor ICS-HK

Pode-se observar, na Figura 37, o esquema do transmissor ICS-HK.

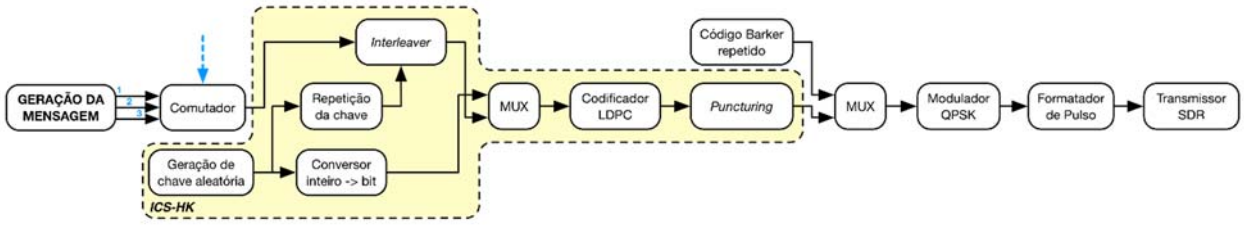

Figura 37 - Diagrama de blocos do transmissor ICS-HK.
A azul representam-se as posições e o sinal de controlo do comutador. A negrito e maiúsculas é representado o subsistema com o mesmo nome visível na Figura 33.

Mais uma vez, à exceção da zona sombreada, o funcionamento é semelhante ao descrito na secção do transmissor ICS (5.2.1), sendo a única diferença a ausência do *jammer*. Como descrito em 2.5, neste esquema, a chave não é codificada com o código BCH, mas é apagada depois da mensagem ser codificada com o código LDPC, no bloco *Puncturing*.

Assim, assumindo, mais uma vez, um código LDPC $(1536,1280)$, se se considerar uma chave de 60 bits, a informação útil corresponde a $1280 - 60 = 1220$ bits de informação em cada pacote. Pretende-se enviar o mesmo ficheiro que anteriormente, pelo que são anexados 40 bits ao mesmo para preencher todos os pacotes $\left(\left\lceil\frac{100000}{1220}\right\rceil \times 1220 = 100040\right)$. Mantêm-se, ainda, as restrições de que o tamanho, em inteiros, da chave seja divisor de 1220 e de o número de bits usados para representar cada inteiro ser suficiente para simbolizar o maior inteiro da chave. Note-se, ainda, que, devido ao *puncturing*, apenas os últimos $1536 - 60 = 1476$ bits codificados são transmitidos.

Recetor ICS-HK

A Figura 38 revela o diagrama do recetor ICS-HK.

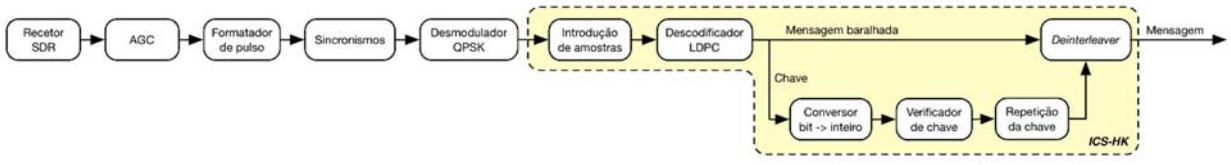

Figura 38 - Diagrama de blocos do recetor ICS-HK.

Na zona sombreada, além da não utilização, óbvia, do descodificador BCH, a diferença em relação ao recetor ICS prende-se com a introdução, no início da mensagem, antes do descodificador LDPC, do número de amostras correspondente ao tamanho da chave, todas a 0. Isto deve-se ao facto de o descodificador LDPC esperar 1536 amostras e não (para o caso de uma

chave com 60 bits) 1476, que, devido à exclusão dos bits correspondentes à chave, é a quantidade enviada.

5.2.4. SCS-HK

Por fim, foi implementado também o esquema SCS-HK, semelhante ao ICS-HK, mas com o papel do *interleaver* a ser desempenhado por um *scrambler*. Como consequência direta desta alteração, a chave passa a ser o conjunto dos estados iniciais dos registos de deslocamento do *scrambler* e esta é, também, anexada à mensagem, codificada e apagada de seguida, tal como acontece no ICS-HK.

Transmissor SCS-HK

Na Figura 39, pode ser visualizado o esquema do transmissor SCS-HK.

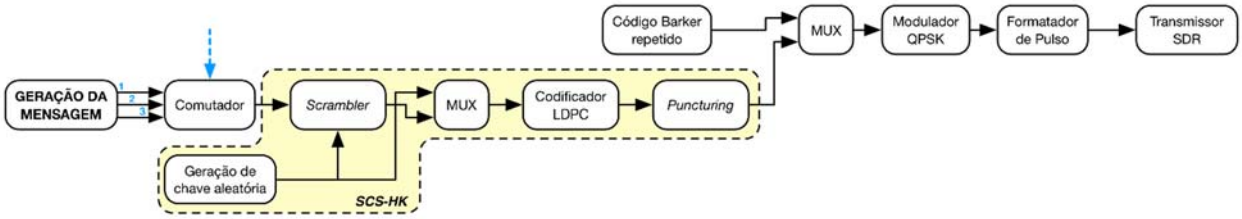

Figura 39 - Diagrama de blocos do transmissor SCS-HK.
A azul representam-se as posições e o sinal de controlo do comutador. A negrito e maiúsculas é representado o subsistema com o mesmo nome visível na Figura 33.

Neste transmissor, como expectável, a mensagem é baralhada pelo *scrambler* segundo uma chave aleatória, que indica os estados iniciais dos registos deste e que é, posteriormente, incorporada no início da mensagem a ser transmitida. Este conjunto é, então, codificado com um código LDPC e são removidos os primeiros K_{len} bits, em que K_{len} é o tamanho da chave. Este transmissor é, em tudo o resto, igual ao transmissor ICS-HK.

Recetor SCS-HK

O recetor SCS-HK apresenta-se como na Figura 40.

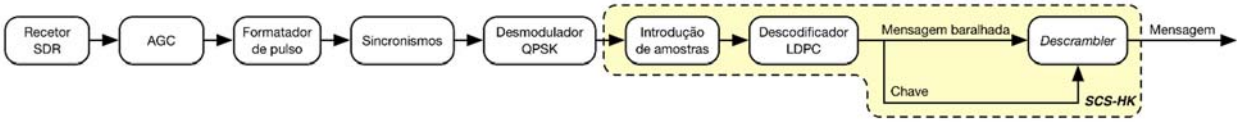

Figura 40 - Diagrama de blocos do recetor SCS-HK.

Neste recetor, semelhante ao recetor ICS-HK, usa-se um *descrambler* para desembaralhar a mensagem transmitida segundo o esquema SCS-HK. Tal como no ICS-HK, são introduzidas amostras a zero antes da descodificação LDPC, para perfazer o tamanho esperado pelo descodificador.

5.2.5. Modelo de referência

Para validar o desempenho dos modelos com chave escondida, foi implementado, também, um esquema de referência, em que o código LDPC é usado, apenas e só, para garantir uma transmissão robusta (i.e. sem erros) de dados, sem que haja restrições de segurança. A Figura 41 e a Figura 42 mostram, respetivamente, o transmissor e recetor deste esquema.

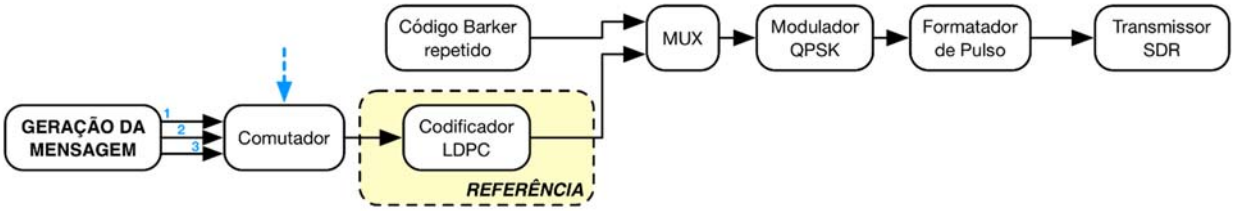

Figura 41 - Diagrama de blocos do transmissor de referência.
A azul representam-se as posições e o sinal de controlo do comutador. A negrito e maiúsculas é representado o subsistema com o mesmo nome visível na Figura 33.

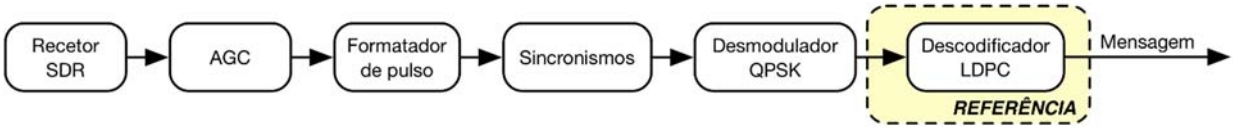

Figura 42 - Diagrama de blocos do recetor de referência.

Como se pode ver, neste caso, os dados são enviados inalterados, i.e., sem que lhes seja aplicada qualquer codificação de segurança (*interleaving* ou *scrambling*). Tomando como exemplo o mesmo código LDPC (1536,1280), como não há nenhuma chave a anexar neste esquema, são enviados 1280 bits de informação em cada pacote.

5.3. Parâmetros gerais

Apesar dos modelos apresentados na secção anterior apresentarem diferenças entre si, o funcionamento base é transversal a todos. Por isso, para uma melhor compreensão, optou-se por apresentar em conjunto, na Tabela 1, os parâmetros relativos aos diversos modelos.

Frequência de amostragem	200 kHz *
Ficheiro a transmitir	100000 bits aleatórios
Código LDPC	(1536,1280) WiMAX (IEEE 802.16e)
Tamanho do código de Barker	13
Fator de expansão do filtro formatador de pulso	4
Fator de excesso de banda	0.5

Frequência de transmissão	$5\ GHz$
Número de ficheiros transmitidos	50
Iterações máximas do descodificador LDPC	10
K_p da PLL do sincronismo de fase	$2KA^2$
K_0 da PLL do sincronismo de fase	1
K_p da PLL do sincronismo de relógio	2×2.68
K_0 da PLL do sincronismo de relógio	-1
Largura de banda normalizada das PLL	0.01
Fator de amortecimento das PLL	1

Tabela 1 - Parâmetros comuns.

** Por exigir um menor esforço computacional, foi possível alcançar uma taxa de $300\ kHz$ no modelo de referência.*

5.4. Hardware utilizado

No decorrer deste trabalho, foram utilizadas 3 plataformas SDR USRP B210 (uma para cada interveniente: Alice, Bob e Eve), equipadas com antenas omnidirecionais VERT2450 e ligadas, cada uma delas, a um computador diferente, estando as características relevantes destes equipamentos descritas na Tabela 2. Na Figura 43, pode-se observar a montagem da Eve, como exemplo.

		Alice	**Bob**	**Eve**
Computador	Modelo	Toshiba Satellite $P50$ - C - $15L$	Asus $X555LD$	Asus $FZ50VX$
	Processador	Intel Core $i7$ - $5500U$ $2\times2.4\ GHz$	Intel Core $i7$ - $4510U$ $2\times2.6\ GHz$	Intel Core $i7$ - $6700HQ$ $4\times2.6\ GHz$
	RAM	$16\ GB$	$8\ GB$	$8\ GB$
	Sistema Operativo	Windows 10		
	Plataforma de desenvolvimento	MATLAB $R2016b$		
	Versão USB	3.0		

SDR	Modelo	USRP B210
	Gama de frequências	$70\ MHz - 6\ GHz$
	Largura de banda	Até $56\ MHz$
	Versão USB	3.0
	Suporte MIMO	Sim
Antena	Modelo	VERT2450
	Gama de frequências	$2.4\ GHz - 2.5\ GHz$ $4.9\ GHz - 5.9\ GHz$
	Tipo	Omnidirecional
	Ganho	$3\ dBi$

Tabela 2 - Hardware utilizado.

Apesar dos computadores terem boas especificações, a frequência de amostragem é baixa dado que os cálculos são todos efetuados no computador e não há paralelização. Por isso, a vantagem de ter vários *cores* não é aproveitada, uma vez que apenas um é utilizado, o que limita bastante a velocidade de processamento.

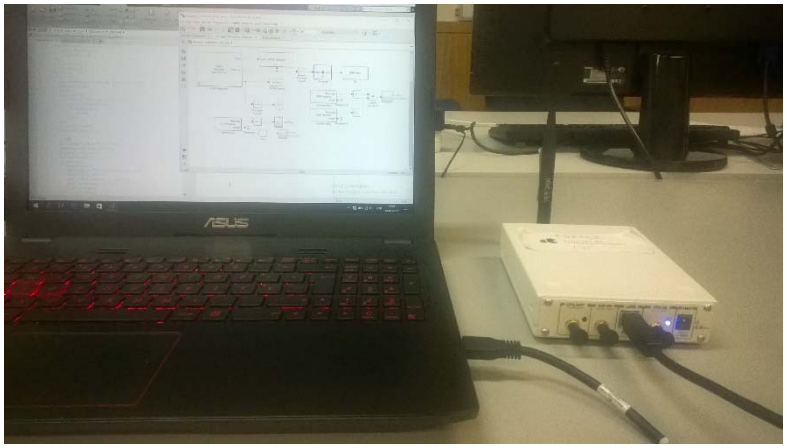

Figura 43 - Montagem da Eve: Computador ligado por USB 3.0 a uma USRP B210 com uma antena VERT2450 acoplada.

5.5. Cenários de testes

Para testar a eficácia dos esquemas propostos no aumento da segurança de uma transmissão de dados sem fios, é necessário estabelecer cenários de teste que possam avaliar o desempenho de cada modelo num ambiente semelhante àquele em que se propõe a sua utilização.

Por isso, nos primeiros esquemas, em que se considera a utilização de um *jammer* para degradar o canal da Eve, seria interessante aferir as suas capacidades realizando uma transmissão em espaço aberto, com uma antena omnidirecional, para dois recetores à mesma distância, mas em posições diferentes. Um seria o Bob e o outro a Eve, que teria ainda uma antena direcional apontada a ela a emitir ruído aleatório. Este cenário é mostrado na Figura 44.

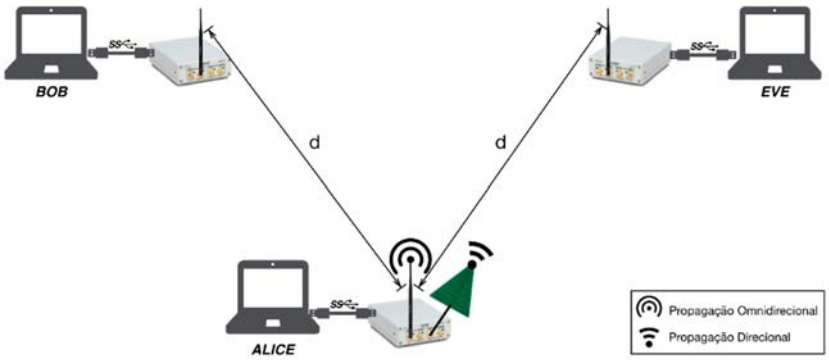

Figura 44 - Cenário de teste para esquema com jammer.

No cenário da Figura 44, adequado aos esquemas com *jammer*, seria desejável ter controlo sobre a relação de potências entre o transmissor e o *jammer*. Infelizmente, as antenas disponíveis (omnidirecional e direcional) mostraram valores de potência transmitida muito díspares quando sujeitas ao mesmo ganho, situação pela qual não foi possível efetuar medições para estes casos, embora os esquemas estejam implementados e funcionais.

Para os restantes esquemas, com a chave escondida, em que é necessário um canal degradado para a Eve, o cenário da Figura 44 não é aplicável. Tendo em conta uma hipotética situação real, um bom cenário de utilização destes modelos seria efetuar a transmissão legítima de ficheiros dentro de um compartimento fechado, ou seja, cercado por paredes e/ou portas, e a colocação de um *eavesdropper* encostado a uma das paredes desse compartimento, no exterior, tentando a melhor aproximação possível do transmissor. De forma a avaliar o efeito da parede na degradação do canal, seria interessante que os dois recetores, Bob e Eve, distassem o mesmo valor do transmissor, Alice. Um esquema com este cenário pode ser visto na Figura 45.

Figura 45 - Cenário de teste para esquema com chave escondida.

Atendendo a estas características desejadas e aos espaços aos quais seria possível aceder, foram escolhidas as salas de aula T5.1 e T5.2 do DEEC da UC, contíguas, como sala onde a transmissão legítima é efetuada e espaço onde a Eve tenta intercetar a mesma, respetivamente. Os

dois recetores foram colocados em linha com o transmissor e a distância entre cada um deles e a Alice foi estabelecida em 1.5 metros. O caminho entre a Alice e o Bob foi desimpedido e entre a Alice e a Eve existia apenas uma parede de 30 cm de espessura. De forma a minimizar os efeitos multipercurso, que pudessem influenciar os resultados, e, tendo em conta as limitações impostas pela necessidade de alimentação dos computadores e da disposição dos pontos de corrente na sala, procurou-se, também, afastar os rádios das portas (envidraçadas). Com esse objetivo, conseguiu-se um afastamento de 2.7 m. Na Figura 46, pode ser observada uma planta das salas usadas e o posicionamento de cada interveniente.

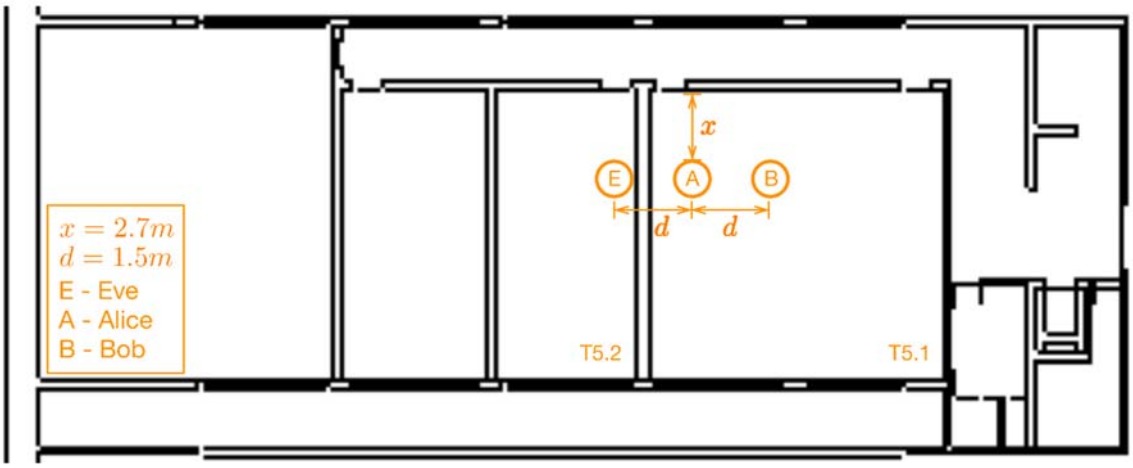

Figura 46 - Configuração do cenário real de testes.
NOTA: Figura não está à escala.

5.6. Calibração de frequências entre SDR

Antes de realizar os testes referidos, importa, ainda, proceder à calibração de frequências entre as placas usadas. Este é um passo necessário, dadas as imperfeições na construção dos osciladores de cada uma delas, conduzindo a pequenos desvios de frequência, mesmo que todas as placas sejam configuradas para a mesma frequência. Estes desvios introduzem erro na comunicação e podem conduzir a resultados adulterados.

Para solucionar este problema, repetindo a experiência para cada par transmissor-recetor: Alice-Bob e Alice-Eve, enviou-se um sinal à frequência de interesse, $5\,GHz$, e usou-se um analisador de espetros no recetor para medir a frequência do sinal recebido, obtendo-se, assim, a diferença entre a frequência recebida e a frequência desejada. De seguida, somou-se o valor obtido aos $5\,GHz$, usando o resultado como frequência de operação nos recetores e conseguindo-se, deste modo, calibrar as frequências entre as SDR.

5.7. Identificação da mensagem

Como visto anteriormente, os vários sincronismos a que um recetor de comunicação digital está sujeito permitem que este consiga extrair informação de um conjunto aparentemente aleatório

de bits. Um mau processo de sincronismo pode fazer com que não seja possível recuperar um sinal, mesmo que tenha sido transmitido em excelentes condições: transmissor potente, distância curta, poucas interferências e bons códigos de recuperação de erros.

No entanto, estes podem trazer outro tipo de problemas que exijam uma reformulação do sistema. A existência de um controlador PI nas PLL usadas nos sincronismos de frequência e relógio faz com que, na estimação do erro de frequência e/ou relógio, sejam considerados todos os valores desde o arranque do sistema para correção dos erros introduzidos pelo canal. Isto deve-se à função de memória implementada pela componente integrativa deste controlador. Em termos práticos, esta particularidade impede que exista um bom sincronismo quando o recetor é ligado antes de existir sinal enviado pelo transmissor. Por isso, para que os dois recetores considerados neste trabalho possam escutar exatamente a mesma informação ao mesmo tempo, optou-se por construir um transmissor que envia bits aleatórios desde que é inicializado até que o comutador presente nos esquemas da secção 5.2, que recebe um sinal de controlo, altere o seu estado, altura em que começa a ser enviada a mensagem que se deseja, efetivamente, transmitir.

Estes bits aleatórios são, tal como os da mensagem, enviados em pacotes. De forma a ser possível, no recetor, identificar em que pacote acabam os bits aleatórios e, consequentemente, onde começa a mensagem, foi testado, inicialmente, um mecanismo de anexação de um cabeçalho com um número, sequencial, em cada pacote, para que, do lado do recetor, se pudesse encontrar uma sequência e, assim, perceber onde começava e terminava a transmissão. Com esta proposta, era usado um cabeçalho de 20 bits e cada pacote tinha a forma apresentada na Figura 47.

Figura 47 - Estrutura do pacote transmitido com o seu número associado.

Esta abordagem tem, além da óbvia desvantagem de diminuir o número de bits de informação presentes em cada pacote, o problema deste cabeçalho ser, também ele, afetado por SNR baixas. Este problema é ainda mais significativo neste trabalho, em que se exploram, precisamente, as zonas em que essas relações são baixas. Por isso, qualquer estratégia de identificação baseada em abordagens deste género (utilizando a informação presente em cabeçalhos ou apêndices) falha nestas condições.

Como alternativa, considerou-se a utilização de códigos de Barker, apresentados no capítulo 4, devido às suas boas propriedades de autocorrelação. Assim, entre o último pacote com bits aleatórios e o primeiro com mensagem, é enviado um outro com um código de Barker de comprimento 13 repetido ao longo de todo o pacote. Do lado do recetor, ao efetuar o cálculo do

valor máximo da correlação cruzada dos dados de cada pacote com o código de Barker, para boas condições de SNR, é possível identificar, de forma clara e inequívoca, os pacotes que marcam o início e o fim da transmissão, correspondentes à localização do código de Barker, como se pode ver na Figura 48, que diz respeito a um teste com probabilidade de erro aproximadamente igual a zero.

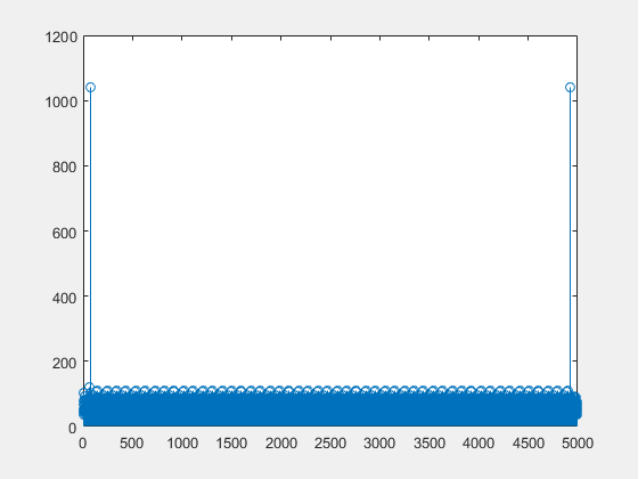

Figura 48 - Máximos da correlação cruzada do código de Barker com os 4850 pacotes de informação e pacotes de bits aleatórios enviados numa transmissão com o esquema de referência, numa situação com probabilidade de erro aproximadamente nula.

Além de dispensar o uso de um cabeçalho, aumentando o número de bits de informação por pacote, a vantagem desta abordagem é que funciona mesmo para SNR baixas, ou seja, é possível identificar os pacotes sinalizadores de início e fim de transmissão até quando as SNR conduzem a probabilidades de erro da mensagem transmitida próximas de 0.5, caso que é visível na Figura 49.

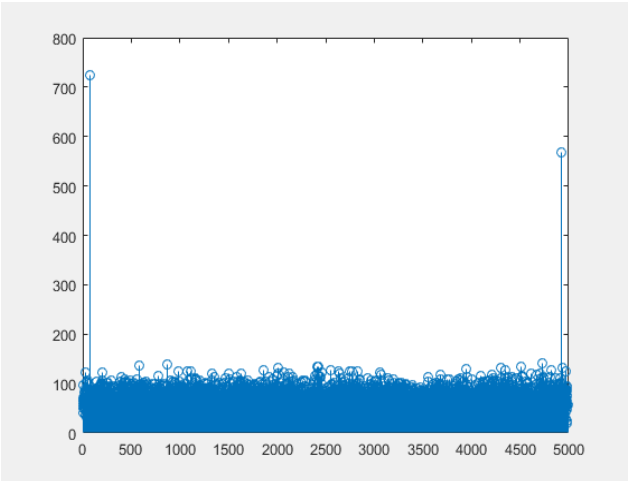

Figura 49 - Máximos da correlação cruzada do código de Barker com os 4850 pacotes de informação e pacotes de bits aleatórios enviados numa transmissão com o esquema de referência, numa situação com probabilidade de erro aproximadamente igual a 0.5.

No entanto, como todas, esta abordagem também tem problemas. O maior é que, em caso de falha na deteção do início da transmissão, mesmo que toda a transmissão seja desprovida de erros, é impossível saber que dados analisar.

Apesar destes problemas, este foi o procedimento escolhido para este trabalho, por, ainda assim, ser robusto. Para isso, e porque é expectável que, em boas condições, sejam recebidos valores que conduzam a um vetor de máximos de correlações cruzadas semelhante ao da Figura

48, para cada transmissão efetuada são analisados os máximos das correlações cruzadas e são consideradas válidas apenas aquelas em que se acredita ter encontrado os sinalizadores referidos.

5.8. Condições de teste

De forma a obter resultados estatisticamente válidos, cada experiência foi repetida, pelo menos, 30 vezes, o que se mostrou suficiente para obter resultados precisos.

A frequência de operação escolhida, 5 GHz, teve em conta a gama de frequências suportadas pelo SDR e as especificações da antena, bem como um estudo prévio das frequências em uso por outros aparelhos no cenário de teste (nomeadamente bandas de operação da cobertura Wi-Fi do edifício).

Para os modelos com chave escondida (e respetivo modelo de referência), os testes foram efetuados com os intervenientes em posições fixas, como descrito na secção 5.5, onde o ganho aplicado ao transmissor foi variado, variando, desta forma, a SNR.

Assim, atendendo à restrição do transmissor dever ser ligado em primeiro lugar, de forma a garantir o sincronismo da receção, e que devem ser efetuadas pelo menos 30 repetições para cada ganho, a sequência de teste realizada para cada modelo encontra-se descrita na Figura 50.

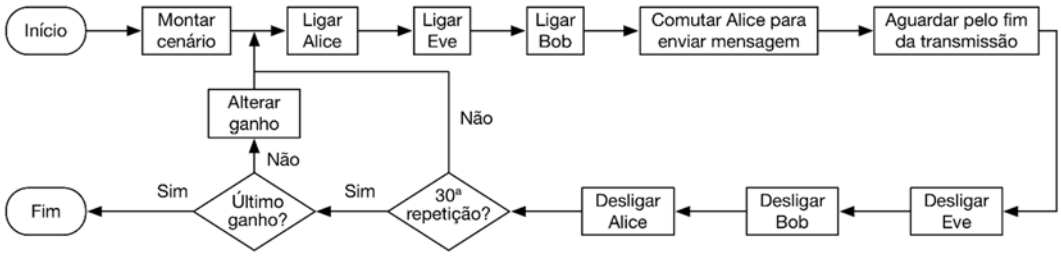

Figura 50 - Fluxograma dos testes efetuados para cada modelo.

Uma vez que os SDR estavam localizados em duas salas diferentes e não estavam posicionados junto dos acessos das mesmas, o processo de os ligar e desligar era demorado e cansativo. Tomando, como exemplo, o esquema de controlo, cuja taxa de transmissão é de 300 kHz e transmite $50\times\left\lceil\frac{100000}{1040}\right\rceil = 50\times97$ pacotes[7], o tempo de transmissão é de pouco mais de 41 segundos. Segundo medidas tiradas sem rigor, para se efetuar uma transmissão de, aproximadamente, 41 segundos, o tempo gasto era da ordem dos 2 minutos, correspondendo ainda a uma distância percorrida de cerca de 1 km por cada ganho diferente (30 experiências). Atendendo a que a gama de ganhos de interesse é dos 38 dB aos 84 dB, usando incrementos de $2dB$ e ignorando outros fatores de perda de tempo, como o cansaço ou um maior tempo de reação, o teste deste modelo levaria, sensivelmente, 24 horas, sendo que o tempo de transmissão

[7] No momento em que esta medição foi feita, estava a ser usado um código LDPC (1248,1040).

corresponderia a pouco mais de 8 horas. Estes valores tomam ainda maiores proporções nos restantes modelos, em que a taxa de transmissão é mais baixa. Isto, aliado ao facto de o cenário de testes possível ser composto por duas salas de aula, limitando o acesso às mesmas para testes a períodos em que não estavam ocupadas, tornava muito complicado efetuar as medições pretendidas e obrigava à montagem e desmontagem do cenário, introduzindo erro nos resultados.

Pelos motivos descritos, tornou-se necessário encontrar uma alternativa a este processo manual, que automatizasse o processo e diminuísse, de forma drástica, o tempo necessário.

5.9. Automatização dos testes

A necessidade de automatizar estas medições tornou-se premente pelos motivos descritos na última secção. A primeira solução, mais simples, seria a de centralizar o controlo de todo o processo num só computador, fazendo uso de acesso remoto para comandar os outros dois, nomeadamente o início e paragem dos modelos dos outros. No entanto, este é um processo pesado computacionalmente e implicaria uma redução significativa da taxa de transmissão, pelo que não consistia numa solução interessante.

Outra abordagem possível, mais trabalhosa, mas muito menos exigente, do ponto de vista computacional, é a implementação de um sistema de *sockets* para comunicação entre os computadores. Os principais tipos de *sockets* funcionam em cima dos protocolos TCP ou UDP. Enquanto o primeiro providencia uma comunicação fiável, em que a ligação é estabelecida antes da informação ser enviada, garantindo a entrega dos pacotes e pedindo retransmissão, se necessário, o segundo envia os pacotes sem qualquer garantia de entrega [55] [56].

Porque, para o propósito deste trabalho, é exigida uma comunicação em tempo real, escolheu-se o protocolo UDP para trocar informações entre os computadores, uma vez que o processo de estabelecimento de ligação do TCP bloqueia os processos até obter uma resposta.

Este procedimento foi realizado, essencialmente, com base nos blocos ‘UDP Send’ e ‘UDP Receive’ do Simulink, que enviam e recebem informação, respetivamente, mas também com as funções ‘udp’, ‘fwrite’ e ‘fread’ do MATLAB, que criam um objeto UDP, enviam e recebem informação através dele, respetivamente. Um diagrama temporal com as informações trocadas entre os diferentes computadores pode ser visto na Figura 51.

Neste diagrama, vê-se que o primeiro passo é ligar todos os computadores à mesma rede, para que consigam comunicar. De seguida, os dois recetores ficam à espera que a Alice inicie a transmissão de bits aleatórios. Quando isso acontece, esta envia uma mensagem para ambos a informar que está ligada. De seguida, os dois recetores iniciam-se e, como o transmissor já está

ligado, sincronizam. Estando todos os SDR prontos, os recetores enviam uma mensagem à Alice, não necessariamente ao mesmo tempo, a informar disso. Quando o transmissor toma conhecimento da disponibilidade dos recetores, comuta o seu estado e começa o envio dos pacotes relativos à mensagem a transmitir. Finda a transmissão da mensagem, a Alice envia para o Bob e a Eve uma mensagem a informar desse fim e termina a sua execução. Os recetores, ao receberem esta mensagem, terminam também.

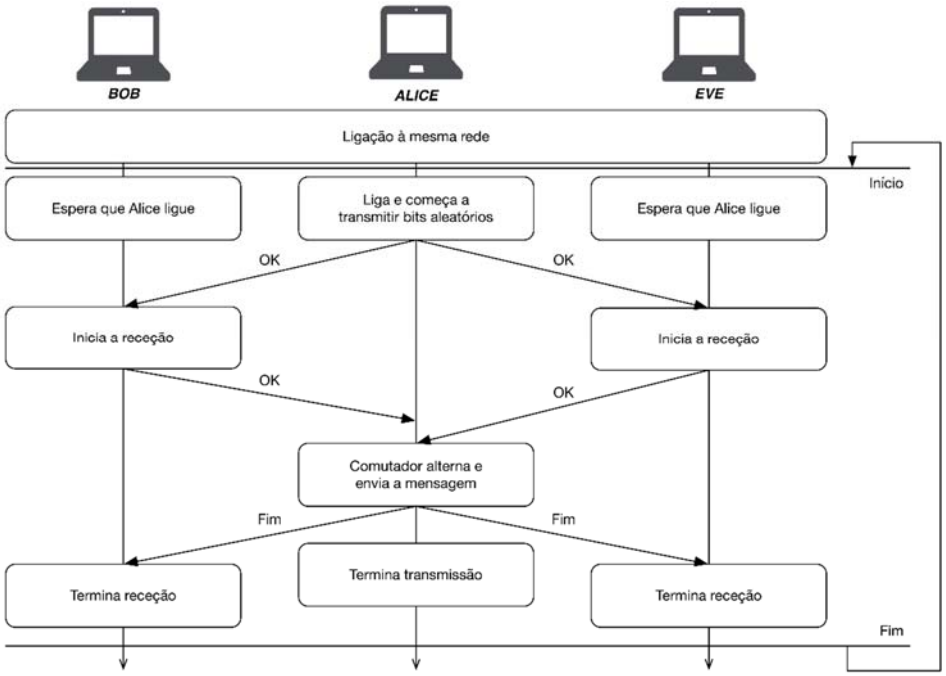

Figura 51 - Diagrama temporal com procedimentos e trocas de pacotes UDP entre os computadores afetos a cada SDR.

Uma vez implementado, este procedimento foi colocado dentro de um ciclo que incrementa o número do teste (de 1 a 30) e os ganhos. Apesar das taxas de transmissão terem sido testadas e adequadas a cada modelo e de cada computador ter sido alvo de diversas otimizações, ocorrem, por vezes, por motivos relacionados com a própria gestão do sistema operativo, *overflow*s. Nestes casos, pacotes são perdidos e o teste deixa de ser válido. Por isso, em paralelo, foi implementado, também, um esquema de mensagens UDP que informe os outros dois computadores sempre que um dos intervenientes tiver um erro causado por *overflow*. Quando isto acontece, esse teste é repetido.

Apesar deste processo de automatização, esta execução deve ser acompanhada, devido à possibilidade de ocorrerem outros erros, relacionados com falhas de rede, por exemplo, embora isso ocorra com uma frequência relativamente baixa.

Por fim, deve ser ainda referido que este procedimento pode ser usado para qualquer modelo com estes três intervenientes: Alice, Bob e Eve.

6. Resultados de Simulação em USRP

Depois de descrito o processo de implementação dos modelos propostos e a sua utilização, este capítulo focar-se-á na análise dos resultados obtidos com as experiências descritas e na discussão das suas implicações práticas. Importa, ainda, recapitular que, devido a divergências de eficiência das antenas disponíveis, não foi possível realizar os testes dos esquemas ICS e SCS, como referido em 5.5.

6.1. Validação de resultados

Devido à imprevisibilidade introduzida pela realização de experiências em ambiente real, é importante definir, antes de tomar em consideração os resultados obtidos, se estes são válidos ou se foram alvo de interferências capazes de os adulterar. Nesta secção, são definidos os critérios de validação de resultados.

Para um conjunto de dados ser considerado válido, deve-se verificar se é possível identificar que parte dos dados corresponde à mensagem transmitida e, para isso, calculou-se a correlação cruzada entre os dados recebidos e a sequência de Barker de tamanho 13, com o objetivo de identificar os pacotes sinalizadores introduzidos na transmissão, como descrito em 5.7.

A identificação destes pacotes sinalizadores mostrou-se um desafio maior do que o previsto, pois, em situações de SNR baixas, nem sempre a localização destes correspondia aos valores máximos de correlação, como expectável, embora, visualmente, se conseguissem identificar. Um exemplo deste caso é mostrado na Figura 52.

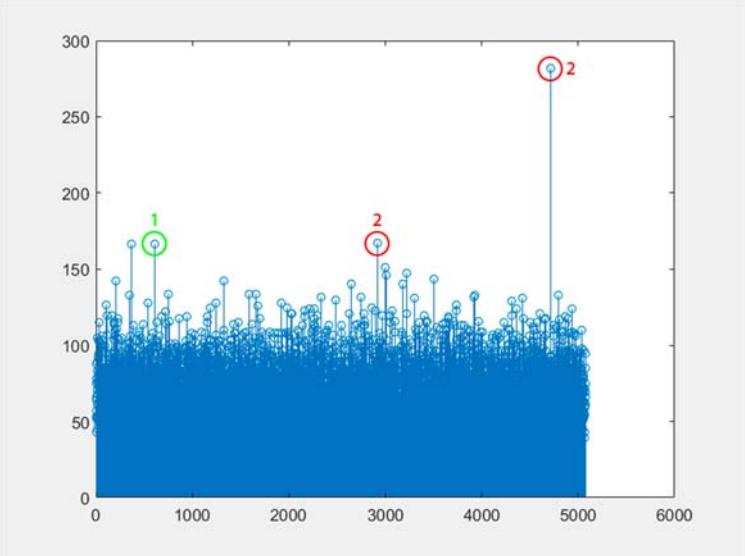

Figura 52 - Exemplo de erro na deteção de pacotes sinalizadores através dos dois valores máximos. O gráfico traça o valor da autocorrelação com o código de Barker de tamanho 13 em cada pacote recebido.

Esta figura corresponde a uma transmissão de 4100 pacotes. Os dois valores máximos de correlação estão representados na figura a vermelho e com o número 2. Pelo seu valor e posição,

percebe-se que o máximo mais à direita corresponde ao pacote sinalizador de fim de transmissão. No entanto, por inspeção visual apenas, percebe-se que, dado o número de pacotes transmitidos, o máximo mais à esquerda não pode corresponder ao pacote sinalizador de início de transmissão. Dois fortes candidatos a serem esse pacote são os correspondentes aos dois primeiros picos destacados, que têm o mesmo valor de correlação. Pelo número de pacotes transmitidos, conclui-se que o pacote sinalizador de início de transmissão é o assinalado a verde e com o número 1, na Figura 52.

Como não é exequível inspecionar manualmente os mais de 4000 resultados obtidos, foi construído um algoritmo de pesquisa destes pacotes sinalizadores. Esta pesquisa foi feita em duas fases distintas, explicadas de seguida.

Atendendo a que se assume que o recetor legítimo é conhecedor do tamanho da mensagem a enviar em cada transmissão e que o *eavesdropper* tem os mesmos conhecimentos do recetor legítimo, para que os erros de identificação de sinalizadores sejam minimizados, foram divididos os dados recebidos em cada transmissão em duas zonas distintas: a que contém as posições em que pode estar situado o sinalizador de início de mensagem (Z_i) e a que engloba as posições em que se pode situar o sinalizador de fim de mensagem (Z_f), definidas como:

$$Z_i: [1, N - M - 1] \quad , \tag{36}$$

$$Z_f: [M + 2, N] \quad , \tag{37}$$

onde N é o tamanho do ficheiro recebido e M o tamanho da mensagem enviada.

A grande maioria das transmissões efetuadas em condições suficientes para a receção de alguma parte da mensagem transmitida têm o comportamento esperado, em que os pacotes sinalizadores se destacam do resto dos bits enviados, no que ao valor da correlação cruzada com o código de Barker diz respeito e, por isso, faz sentido começar por pesquisar os valores máximos desta. Deste modo, a primeira abordagem do referido algoritmo consiste em encontrar os índices dos 3 maiores valores de cada zona e, por ordem descendente, verificar se a diferença entre cada par de índices corresponde ao tamanho esperado, M.

Como, em alguns casos, um dos pacotes sinalizadores chegou ao recetor afetado por ruído suficientemente alto para que não se destaque dos outros em termos dos valores de correlação referidos, podem existir transmissões válidas em que nenhum dos pares de máximos analisados corresponde aos pacotes sinalizadores enviados. Nessa situação, avança-se para a segunda fase do algoritmo, em que se tenta encontrar o pacote sinalizador de início de transmissão, pois, como se tem conhecimento do número de pacotes transmitidos, isso é suficiente para detetar a mensagem.

Apesar de ser possível identificar a mensagem quando é detetado apenas o pacote sinalizador de fim de transmissão, foram aceites apenas as transmissões em que o sinalizador detetado foi o de início de transmissão, uma vez que, numa análise na perspetiva de um *eavesdropper*, que estaria à escuta de uma transmissão, faria sentido que ele começasse a gravar apenas quando detetasse o sinalizador de início, uma vez que, após a deteção de um sinalizador de fim, não poderia voltar atrás no tempo para gravar os dados já perdidos.

Para encontrar o pacote sinalizador de início de transmissão, o algoritmo procura apenas na zona Z_i, identifica o valor máximo e verifica se está acima de um determinado limiar e se é o único nessas condições. Se isso acontecer, a transmissão é considerada válida.

Quanto ao valor do limiar, este teve de ser previamente calculado. Esse cálculo teve por base os valores da correlação cruzada de todos os testes com o código de Barker utilizado. A partir de todos esses valores, foi estimada a função de repartição (CDF – *Cumulative Distribution Function*), de onde se extraiu o valor de correlação acima do qual estariam os N_s valores mais elevados, em que N_s é o número total de pacotes sinalizadores (por exemplo, para testes que consideram 20 ganhos diferentes, com 30 repetições cada, para os 2 recetores, resulta $N_s = 2{\times}20{\times}30{\times}2 = 2400$). O uso deste valor como limiar de decisão baseia-se no pressuposto de que, em condições ideais, os N_s valores mais elevados seriam, efetivamente, os dos sinalizadores.

Em resumo, uma transmissão é considerada válida se forem detetados os dois pacotes sinalizadores (de início e fim de transmissão) ou só o sinalizador de início de transmissão. As transmissões inválidas são descartadas para que não influenciem os resultados finais, sendo apresentado, para cada conjunto de resultados, um gráfico de barras indicador do número de transmissões inválidas para cada ganho de transmissão testado.

6.2. Considerações sobre a análise de resultados

Como forma de análise e comparação dos esquemas propostos, foi usado, como métrica de segurança, o conceito de intervalo de segurança, usando o BER e o BER-CDF, explicados em 2.2. No caso do BER, este é analisado para cada ficheiro transmitido, pelo que, para se obter um valor por ganho de transmissão, calculou-se uma média simples do BER de todos os ficheiros recebidos (e válidos) em cada ganho. No entanto, a média da amostra recolhida, *per se*, não é um indicador fiável e, por isso, deve ser acompanhada por um intervalo de confiança [57]. Este permite afirmar, com um determinado grau de confiança, que a média real de uma determinada população está contida num intervalo construído a partir da amostra recolhida [57]. Nas análises apresentadas de seguida, foram utilizados graus de confiança de 95%.

Em [57], apresenta-se a distribuição de Student, que permite construir intervalos de confiança para populações gaussianas de média e desvio padrão desconhecidos. Contudo, esta pode ser utilizada sem restrições apenas para um número elevado de amostras, sensivelmente acima de 40. Uma vez que o mínimo de amostras a considerar é de 50, correspondentes a uma transmissão válida, a restrição apresentada não é problemática para esta análise. Assim, os intervalos de confiança foram definidos como:

$$\bar{x} \pm t(n-1)\frac{s}{\sqrt{n}} \quad , \tag{38}$$

sendo $\bar{x}$, s e n a média, o desvio padrão e o tamanho da amostra, respetivamente, e t o valor da distribuição de Student de cauda dupla em 0.95 (por ser usado grau de confiança de 95%), para $n-1$ graus de liberdade.

6.3. Escalas utilizadas

As simulações teóricas realizadas anteriormente a este trabalho mostram valores de BER e BER-CDF em função da relação entre a energia de bit de informação e a densidade espetral de ruído (E_b/N_0). No entanto, em cenários práticos, como estes, as dificuldades inerentes à estimação do ruído impedem o uso desta relação. Uma alternativa é a construção das simulações teóricas em função da variação da SNR. No entanto, esta é, também ela, uma relação teórica e que depende da estimação do ruído. Contudo, há um outro fator de qualidade, denominado *Modulation Error Ratio* (MER), que, assumindo um canal AWGN, pode ser equiparado à SNR [58].

O MER providencia uma medida da degradação do sinal recebido através do cálculo dos vetores de erro de modulação, e, que medem a distância de cada símbolo recebido ao respetivo ponto na constelação ideal, no plano I-Q, como ilustrado na Figura 53.

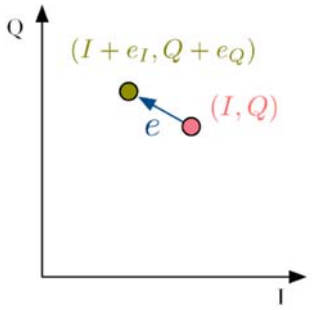

Figura 53 - Vetor erro de modulação no plano I-Q. Este é composto pela distância entre o símbolo ideal, a rosa, e o símbolo recebido, a verde.

Definindo I_j e Q_j, respetivamente, como as componentes em fase e quadratura do símbolo ideal j, e_{I_j} e e_{Q_j}, respetivamente, como as componentes em fase e quadratura do vetor de erro de modulação do símbolo j e N como o número de símbolos recebidos, pode-se definir o MER como [58]:

$$\text{MER} = 10\times\log_{10}\left(\frac{\sum_{j=1}^{N}\left(I_j^2+Q_j^2\right)}{\sum_{j=1}^{N}\left(e_{I_j}^2+e_{Q_j}^2\right)}\right)[dB]\ . \tag{39}$$

Esta medida foi calculada ao longo das experiências realizadas, de forma a conseguir estabelecer um termo de comparação entre as simulações efetuadas e os resultados práticos. Porém, neste cálculo, há uma fragilidade que pode adulterar a escala final. Nos casos em que o ruído é muito elevado, o símbolo enviado pode ser recebido num quadrante diferente do desejado. Por isso, como o MER apenas calcula os vetores de erro para o símbolo mais próximo, ocorre uma subestimação do erro de modulação. Por outro lado, quando a SNR é muito boa, o erro tende para 0, conduzindo a divisões por valores próximos de 0, e que, por sua vez, induzem uma sobrestimação do ruído.

Por este motivo, construiu-se uma escala de conversão entre o ganho aplicado ao transmissor e o MER, baseada unicamente na zona aproximadamente linear da escala medida. Com estes pontos, fez-se uma simples regressão linear, que serviu de conversão para os pontos desejados. Uma vez que este cálculo é simples de efetuar, embora as regressões entre os diversos modelos implementados sejam muito semelhantes, foi feita uma escala diferente para cada um deles, de forma a introduzir o mínimo de erro possível nos resultados. O resultado destas regressões para o esquema SCS-HK, como exemplo, é mostrado na Figura 54.

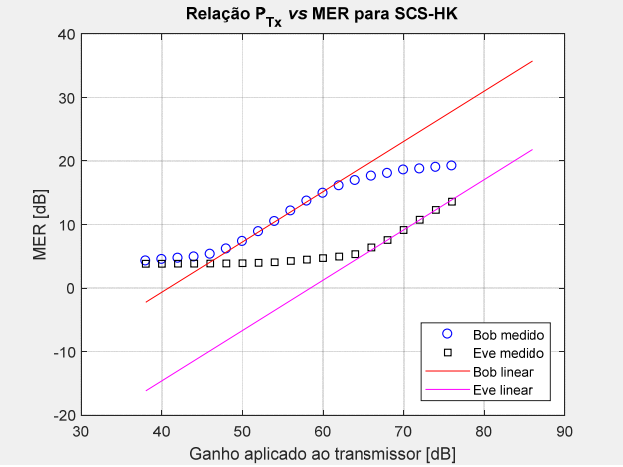

Figura 54 - Relações entre o ganho aplicado ao transmissor (P_{Tx}) e o MER.

Nesta figura, os marcadores representam os valores medidos e as retas as respetivas regressões lineares.

6.4. Resultados

Nesta secção, são apresentados os resultados relativos ao teste, em ambiente real, dos sistemas ICS-HK e SCS-HK implementados em SDR, descritos em 2.5 e 2.6, respetivamente, numa perspetiva teórica e em 5.2.3 e 5.2.4 numa perspetiva prática. Estes testes, realizados com o *hardware* especificado em 5.4 e no cenário referido na secção 5.5, permitiram avaliar o

desempenho desses modelos, em termos de segurança, num cenário real, confirmando os resultados obtidos por simulação teórica em [5].

Como referido, foram analisados apenas resultados considerados válidos pelos procedimentos explicados em 6.1, com recurso às métricas BER e BER-CDF. Em ambas foi utilizado o método de interpolação segmentada de Hermite (PCHIP - *Piecewise Cubic Hermite Interpolating Polynomial*) [59], que preserva a monotonia dos dados e permite uma boa aproximação dos valores obtidos, para obter curvas para os pontos de interesse, representadas por linhas a cheio em todos os gráficos práticos, com exceção da Figura 56. Os pontos medidos foram representados por marcadores não ligados entre si.

6.4.1. ICS-HK e SCS-HK

Nesta secção, são apresentados os resultados obtidos para os esquemas ICS-HK e SCS-HK, bem como para o respetivo modelo de referência, apresentados na secção 5.2. A Figura 55 apresenta os histogramas que indicam a frequência de testes inválidos para cada esquema, em cada recetor e para cada ganho de transmissão testado.

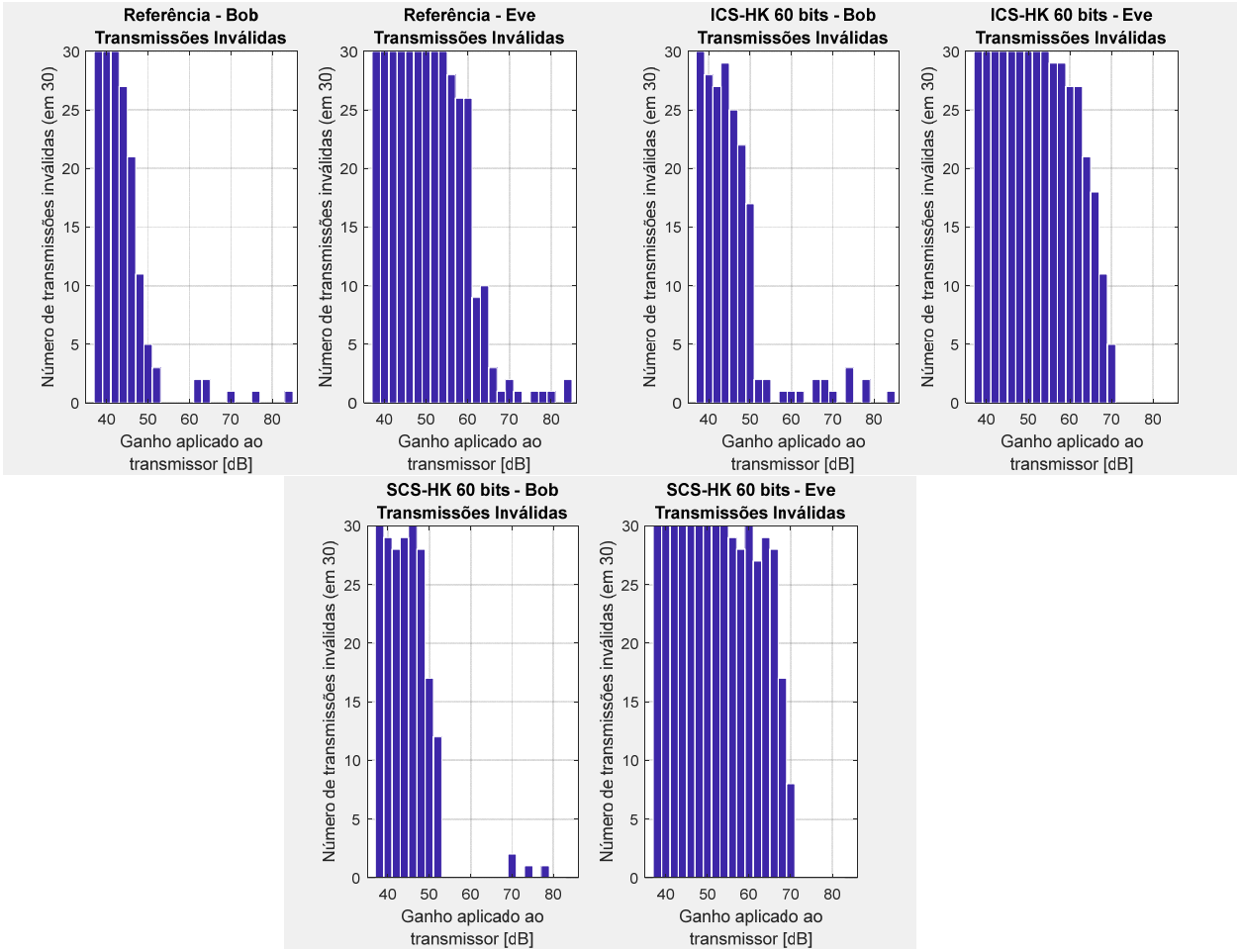

Figura 55 - Frequência de invalidação de transmissões.

Nesta figura, é clara a tendência, expectável, de que o número de transmissões inválidas diminui com o aumento da potência de transmissão. Note-se que, no caso das transmissões

realizadas em potências baixas, as poucas transmissões validadas serão, provavelmente, falsos positivos. No entanto, quer um falso positivo, quer uma transmissão inválida, tendem a apresentar uma probabilidade de erro de 0.5, pelo que estes casos não influenciam os resultados finais.

Antes de interpretar os resultados, importa analisar a consistência dos mesmos. Por isso, a Figura 56 apresenta o BER para todos os esquemas, em função do ganho no transmissor, com o intuito de avaliar os intervalos de confiança, representados com barras verticais, em cada ponto.

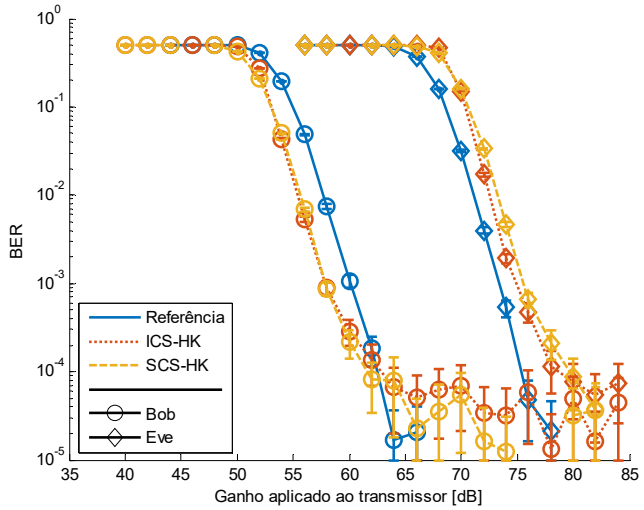

Figura 56 - Sobreposição de todos os BER, para todos os esquemas.

Nela, pode-se constatar que os intervalos de confiança apresentam uma reduzida amplitude até à barreira de BER $= 10^{-4}$. Isso demonstra a consistência dos resultados, pelo que os intervalos de confiança serão desprezados na análise até esse valor.

Por fim, para efetuar uma correta comparação dos resultados obtidos, convém ter, ainda, em consideração os resultados das simulações teóricas, efetuadas em [5], que previam, para o esquema ICS-HK, com uma chave de 60 bits e condições próximas das usadas nas medições práticas um BER e BER-CDF dados pela Figura 57.

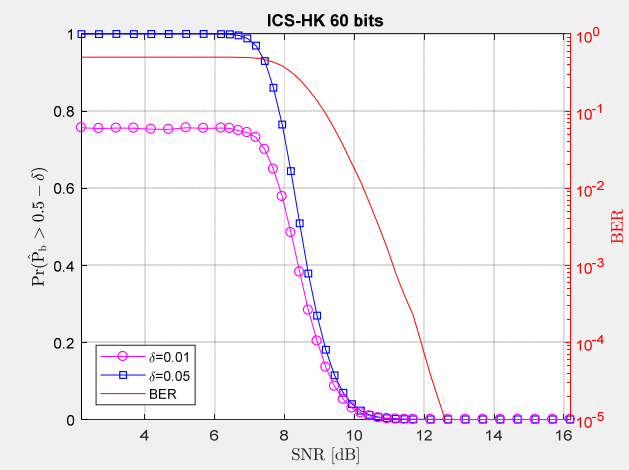

Figura 57 - BER e BER-CDF teóricos para ICS-HK com chave de 60 bits e LDPC (1536,1280).

Em [5], uma transmissão foi considerada segura quando a Eve apresentava uma $\Pr(\hat{P}_b > 0.45) \geq 0.999$, onde $\hat{P}_b$ é a taxa de erros nos bits de informação, e fidedigna se o BER

fosse inferior a 10^{-5}. Nestas condições, para o código LDPC $(1536,1280)$ usado, o valor máximo de SNR a que a Eve poderia operar $(SNR_{E,max})$ seria próximo dos 6.9 dB, enquanto o Bob deveria operar, no mínimo, a $SNR_{B,\min} \cong 12.6\ dB$. Isto resultaria na necessidade de o Bob operar com, pelo menos, $12.6 - 6.9 = 5.7\ dB$ de vantagem sobre a Eve.

Ao traçar as mesmas curvas com base nos resultados práticos de teste nas USRP do esquema ICS-HK, foi obtida a Figura 58.

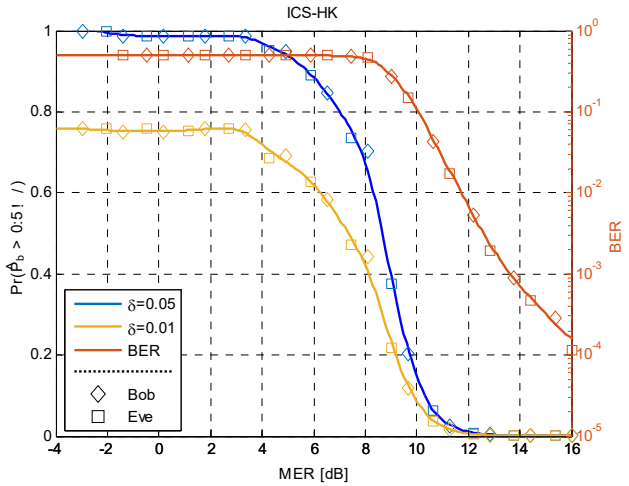

Figura 58 - BER e BER-CDF medidos para ICS-HK com chave de 60 bits e LDPC (1536,1280).

Nesta figura, numa primeira análise, são claras as semelhanças com as curvas teóricas, com ambas a apresentaram o mesmo andamento. Há algumas flutuações, também elas expectáveis, por o canal real não ser AWGN, devido, nomeadamente, às componentes multipercurso e outras interferências presentes aquando da realização das experiências. Este pormenor influencia, também, o grau de semelhança entre o MER e a SNR.

Numa observação mais cuidada, torna-se claro que os limites estipulados para que uma transmissão se considere segura são demasiado exigentes, uma vez que, ao realizar testes em ambiente real, se introduz alguma variabilidade nos resultados, tornando-se muito difícil alcançar a meta de $\Pr(\hat{P}_b > 0.45) \geq 0.999$. Da mesma forma, como, devido ao equilíbrio exigido entre a resolução dos valores de BER medidos e o tempo de processamento e esforço computacional, apenas foram transmitidos 10^5 bits, só se devem considerar adequadas as medições de BER até 10^{-4}, não sendo possível utilizar o mesmo limite de fiabilidade.

Sendo desejada uma probabilidade de erro da Eve sempre próxima de 0.5 para uma transmissão ser considerada segura, um relaxamento que permita uma taxa de erro maior que 0.45 em 98,5% dos casos foi considerada como aceitável, atendendo ao facto de as experiências em ambiente real apresentarem uma maior variabilidade do que uma simulação. Da mesma forma, atendendo ao tamanho do tamanho do ficheiro enviado, considerar fidedigna uma transmissão com BER inferior a 10^{-4} também se assemelha adequado.

Atendendo a estas novas restrições, para o caso prático de ICS-HK, resulta da análise da Figura 58 que $SNR_{E,max} \cong 3.2\ dB$ e $SNR_{B,min} \cong 17.3\ dB$, o que se traduz num *gap* de $17.3 - 3.2 = 14.1\ dB$. Esta pode parecer uma vantagem demasiado grande, quando comparada com os resultados teóricos; no entanto, o principal termo de referência deverá ser uma transmissão nas mesmas condições, mas sem o uso de um *interleaver*, considerando apenas o código LDPC. As experiências realizadas nessas condições conduziram aos resultados apresentados na Figura 59.

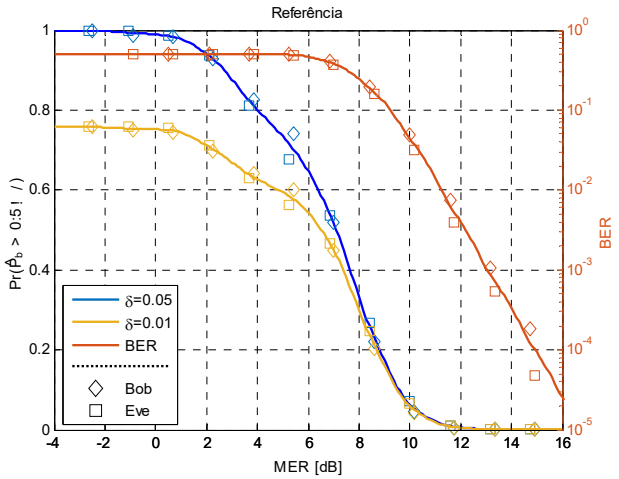

Figura 59 - BER e BER-CDF medidos para o modelo de referência com LDPC.

Aqui, mantendo os mesmos requisitos de segurança e fiabilidade, obtém-se $SNR_{E,max} \cong 0.5\ dB$, $SNR_{B,min} \cong 15\ dB$ e um intervalo de $15 - 0.5 = 14.5\ dB$. Perante estes resultados, confirma-se que o esquema ICS-HK exige uma vantagem ligeiramente menor do Bob sobre a Eve, para o mesmo nível de segurança, embora requeira maiores potências de transmissão.

Já no que respeita à nova técnica, SCS-HK, proposta neste trabalho, que faz uso de uma chave de *scrambling*, ao invés de uma chave de *interleaving*, para as mesmas condições testadas que o ICS-HK, o mesmo código e o mesmo tamanho de chave, obtiveram-se os resultados da Figura 60.

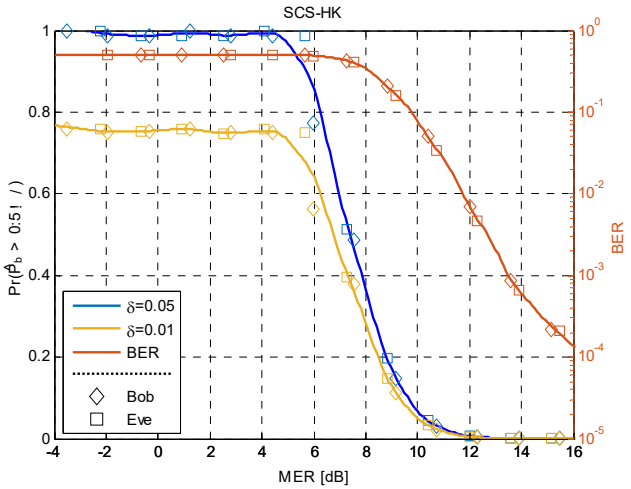

Figura 60 - BER e BER-CDF medidos para SCS-HK com chave de 60 bits e LDPC (1536,1280).

Neste caso, pode-se verificar um $SNR_{E,max} \cong 4.7\ dB$ e um $SNR_{B,min} \cong 16.6\ dB$, o que resulta numa vantagem necessária do Bob sobre a Eve de apenas $16.6 - 4.7 = 11.9\ dB$, para a

mesma segurança. Com este esquema, consegue-se uma redução significativa da vantagem necessária sobre a Eve em relação ao esquema de referência, bem como em relação ao ICS-HK. Outra vantagem em relação ao ICS-HK prende-se com a obtenção de uma transmissão fidedigna com, sensivelmente, menos 0.7 dB de MER, o que significa uma menor potência de transmissão.

É, ainda, possível fazer um resumo com a sobreposição de BER e BER-CDF com $\delta = 0.05$, em separado, para todos os modelos, respetivamente na Figura 61 e Figura 62.

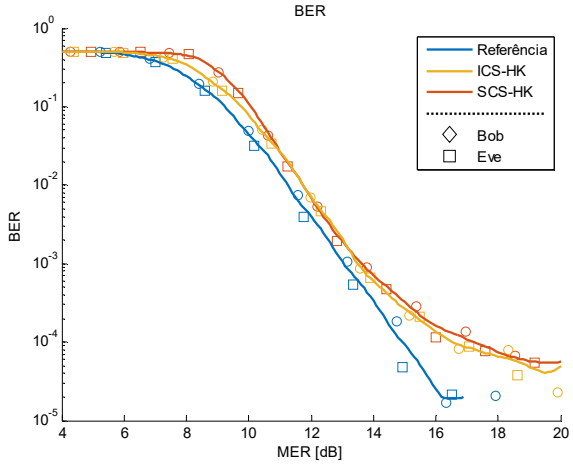

Figura 61 - BER para modelos de referência, ICS-HK e SCS-HK em função do MER.

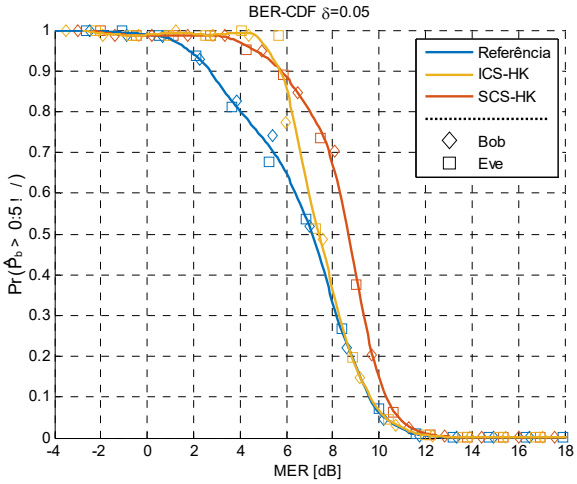

Figura 62 - BER-CDF para modelos de referência, ICS-HK e SCS-HK em função do MER.

Nestas figuras, pode-se ver que, em termos de fiabilidade (medida em BER), o custo de usar um dos esquemas (ICS-HK ou SCS-HK) não é muito elevado, se comparado com o benefício que trazem em segurança (medida em BER-CDF). Em concreto, se, por um lado, o uso de um destes esquemas pode resultar na necessidade de aumentar a SNR em $1.5{\sim}2\ dB$ para que o Bob receba os dados corretamente, por outro, poderá aumentar a segurança em relação à Eve em $2.5{\sim}4\ dB$. Por outro lado, analisando apenas o BER-CDF, vê-se que o SCS-HK mantém probabilidades de erro elevadas até maiores SNR, o que é benéfico em termos de segurança, conseguindo desempenhos semelhantes aos do modelo de referência em SNR altas.

7. Conclusão

O projeto em que está inserido este trabalho procura encontrar novas formas de garantir a segurança das comunicações entre dispositivos, num mundo cada vez mais dominado por estes. Para isso, explora a camada física, que tem vindo a ser ignorada neste contexto, para garantir um complemento de qualidade à criptografia utilizada nas camadas superiores.

Concretamente, nesta dissertação, pretendia-se a implementação dos esquemas ICS e ICS-HK num cenário de testes real, de modo a complementar as simulações teóricas já efetuadas e validar o seu uso num contexto mais prático. Esta foi feita com recurso aos SDR USRP B210 e ao MATLAB/Simulink.

No decorrer da implementação, foi proposta uma variação aos esquemas propostos, consistindo na substituição do *interleaver* por um *scrambler*, usando as condições iniciais dos seus registos como chave.

Para avaliar a utilidade dos modelos propostos, foram realizados testes num cenário em que o Bob possuía uma vantagem (uma parede) em relação à Eve, para ICS-HK e SCS-HK. Neste cenário, não se quis apenas testar para uma vantagem aleatória, mas, especificamente, avaliar o caso de uma transmissão numa sala fechada, garantindo que, na pior das hipóteses, a Eve estaria à escuta do lado de fora da mesma. Infelizmente, o material disponível não permitiu avaliar os modelos ICS e SCS, embora estes tenham sido desenvolvidos.

Como esperado, o modelo proposto inicialmente (ICS-HK) provou ser vantajoso em relação à ausência do mesmo, introduzindo segurança na transmissão. Além deste, o modelo alterado com o *scrambler* (SCS-HK) mostrou-se ainda mais eficaz, com um aumento da segurança para uma chave de iguais dimensões.

Com estas experiências, prova-se que esta abordagem é válida e que pode ser proveitoso explorá-la melhor para, num futuro próximo, se conseguirem comunicações sem fios mais seguras e, assim, contribuir para o avanço tecnológico, nomeadamente através do crescimento da Internet das Coisas, por exemplo.

7.1. Trabalho Futuro

Uma vez que as bases desta linha de investigação estão lançadas, quer em termos teóricos, quer em termos práticos e auguram um bom futuro, o trabalho a fazer será mais num sentido de escolha de parâmetros e integração com plataformas usadas em áreas onde possa ser útil.

Assim, e porque os débitos conseguidos foram baixos, atendendo, principalmente, à complexidade introduzida pelos códigos LDPC e ao processamento efetuado num único *core*, seria importante a realização de um trabalho de otimização. Este poderia passar pelo desenvolvimento de *software* específico para implementação em FPGA e/ou paralelização de parte do processamento, bem como o uso de funções mais eficientes.

Por outro lado, a integração destes modelos em processadores desenvolvidos para dispositivos móveis (como, por exemplo, Snapdragon), embora possa não se traduzir num aumento de desempenho, pode avaliar o comportamento, consumo, limitações e usabilidade desta tecnologia neste tipo de dispositivos.

Seria, também, interessante testar estes modelos noutros cenários que possam ser de interesse, como, por exemplo, usar um raio de segurança como vantagem ou testar o efeito que espessuras e/ou materiais diferentes empregues na construção das paredes de uma sala pudessem ter nos resultados.

Por fim, mediante o cenário e a capacidade do dispositivo onde se pretendesse utilizar estes esquemas, outro caminho de desenvolvimento passaria por testar diferentes códigos e tamanhos de chaves, para que se obtenha um compromisso válido entre segurança e desempenho.

8. Referências

[1] J. P. Vilela, M. Bloch, J. Barros e S. W. McLaughlin, "Wireless Secrecy Regions With Friendly Jamming," *IEEE Transactions on Information Forensics and Security,* vol. 6, nº 2, pp. 256-266, 2011.

[2] A. D. Wyner, "The Wire-Tap Channel," *The Bell System Technical Journal,* vol. 54, nº 8, pp. 1355-1387, 1975.

[3] A. Mukherjee, S. A. A. Fakoorian, J. Huang e A. L. Swindlehurst, "Principles of Physical Layer Security in Multiuser Wireless Networks: A Survey," *IEEE Communications Surveys & Tutorials,* vol. 16, nº 3, pp. 1550-1573, 2014.

[4] J. P. Vilela, M. Gomes, W. K. Harrison, D. Sarmento e F. Dias, "Interleaved Concatenated Coding for Secrecy in the Finite Blocklength Regime," *IEEE Signal Processing Letters,* vol. 23, nº 3, pp. 356-360, Março 2016.

[5] D. Sarmento, J. P. Vilela, W. K. Harrison e M. Gomes, "Interleaved Coding for Secrecy with a Hidden Key," em *IEEE Globecom Workshop on Trusted Communications with Physical Layer Security*, San Diego, CA, USA, 2015.

[6] W. K. Harrison, D. Sarmento, J. P. Vilela e M. Gomes, "Analysis of short blocklength codes for secrecy," *Em Revisão,* 2015.

[7] D. Luciano e G. Prichett, "Cryptology: From Caesar Ciphers to Public-Key Cryptosystems," *The College Mathematics Journal,* vol. 18, nº 1, pp. 2-17, 1987.

[8] W. K. Harrison, J. Almeida, M. R. Bloch, S. W. McLaughlin e J. Barros, "Coding for Secrecy: An overview of error-control coding techniques for physical-layer security," *IEEE Signal Processing Magazine,* pp. 41-50, setembro 2013.

[9] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal,* vol. 28, nº 4, pp. 656-715, 1949.

[10] M. Bloch e J. Barros, Physical-Layer Security: From Information Theory to Security Engineering, Cambridge University Press, 2011.

[11] U. M. Maurer, "The Strong Secret Key Rate of Discrete Random Triples," em *Communication and Cryptography: Two Sides of One Tapestry*, Kluwer Academic Publishers, 1994, pp. 271-285.

[12] D. Klinc, J. Ha, S. W. McLaughlin, J. Barros e B.-J. Kwak, "LDPC Codes for the Gaussian Wiretap Channel," *IEEE Transactions on Information Forensics and Security,* vol. 6, nº 3, pp. 532-540, 2011.

[13] L. Lai e H. E. Gamal, "The Relay-Eavesdropper Channel: Cooperation for Secrecy," *IEEE Transactions on Information Theory,* vol. 54, nº 9, pp. 4005-4019, 2008.

[14] Y. Oohama, "Capacity Theorems for Relay Channels with Confidential Messages," em *IEEE International Symposium on Information Theory*, Nice, 2007.

[15] S. Goel e R. Negi, "Guaranteeing Secrecy using Artificial Noise," *IEEE Transactions on Wireless Communications,* vol. 7, nº 6, pp. 2180-2189, 2008.

[16] E. Tekin e A. Yener, “The General Gaussian Multiple-Access and Two-Way Wiretap Channels: Achievable Rates and Cooperative Jamming,” *IEEE Transactions on Information Theory,* vol. 54, nº 6, pp. 2735-2751, 2008.

[17] A. Thangaraj, S. Dihidar e S. W. McLaughlin, “Applications of LDPC Codes to the Wiretap Channel,” *IEEE Transactions on Information Theory,* vol. 53, nº 8, pp. 2933-2945, 2007.

[18] M. Baldi, M. Bianchi e F. Chiaraluce, “Coding With Scrambling, Concatenation, and HARQ for the AWGN Wire-Tap Channel: A Security Gap Analysis,” *IEEE Transactions on Information Forensics and Security,* vol. 7, nº 3, pp. 883-894, 2012.

[19] J. Almeida e J. Barros, “Random Puncturing for Secrecy,” em *Asilomar*, Pacific Grove, CA, USA, 2013.

[20] H. Mahdavifar e A. Vardy, “Achieving the Secrecy Capacity of Wiretap Channels Using Polar Codes,” *IEEE Transactions on Information Theory,* vol. 57, nº 10, pp. 6428 - 6443, 2011.

[21] M. Yusuf e H. Arslan, “Enhancing Physical-Layer Security in Wireless Communications Using Signal Space Diversity,” em *Military Communications Conference, MILCOM 2016 - 2016 IEEE*, Baltimore, MD, USA, 2016.

[22] J. Mitola III, “Software Radios: Survey, Critical Evaluation and Future Directions,” *IEEE Aerospace and Electronic Systems Magazine,* Abril 1993.

[23] D. Pu e A. M. Wyglinski, Digital Communication Systems Engineering with Software-Defined Radio, Artech House, 2013.

[24] Ettus Research, “USRP N210 Software Defined Radio (SDR) - Ettus Research,” [Online]. Available: https://www.ettus.com/product/details/UN210-KIT.

[25] NooElec, “NooElec - NooElec NESDR SMArt - Premium RTL-SDR w/ Aluminum Enclosure, 0.5PPM TCXO, SMA Input. RTL2832U & R820T2-Based. - SDR Receivers,” [Online]. Available: http://www.nooelec.com/store/sdr/sdr-receivers/nesdr-smart-sdr.html.

[26] J. H. Reed, Software Radio: A Modern Approach to Radio Engineering, Prentice Hall PTR, 2002.

[27] Clearbox, “Spectre,” [Online]. Available: http://www.clearboxsystems.com.au/spectrum-management/spectre/.

[28] Ettus Research, “USRP B210 USB Software Defined Radio (SDR) - Ettus Research,” [Online]. Available: https://www.ettus.com/product/details/UB210-KIT.

[29] Ettus Research, “USRP X310 High Performance Software Defined Radio (SDR),” [Online]. Available: https://www.ettus.com/product/details/X310-KIT.

[30] The GNU Radio Foundation, Inc., “GNU Radio,” [Online]. Available: https://www.gnuradio.org/.

[31] National Instruments, “LabVIEW - National Instruments,” [Online]. Available: http://www.ni.com/pt-pt/shop/labview.html.

[32] Ettus Research, “Ettus Research - The leader in Software Defined Radio (SDR),” [Online]. Available: https://www.ettus.com/.

[33] The Mathworks, Inc., “MATLAB - MathWorks,” [Online]. Available: https://www.mathworks.com/products/matlab.html.

[34] The Mathworks, Inc., “Communications System Toolbox - MATLAB & Simulink,” [Online]. Available: https://www.mathworks.com/products/communications.html.

[35] The MathWorks, Inc., “USRP® Support from Communications System Toolbox - Hardware Support - MATLAB & Simulink,” [Online]. Available: https://www.mathworks.com/hardware-support/usrp.html.

[36] The Mathworks, Inc., “DSP System Toolbox - MATLAB & Simulink,” [Online]. Available: https://www.mathworks.com/products/dsp-system.html.

[37] S. Lin e D. J. Costello, Error Control Coding, Prentice Hall, 2004.

[38] J. G. Proakis, Digital Communications, McGraw-Hill, 2001.

[39] A. B. Carlson, P. B. Crilly e J. C. Rutledge, Communication Systems, McGraw-Hill, 2002.

[40] S. Haykin, Communication Systems, John Wiley & Sons, Inc., 2001.

[41] D. J. C. Mackay e R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *Electronics Letters,* vol. 32, nº 18, pp. 1645-1646, 1996.

[42] R. G. Gallager, “Low-Density Party-Check Codes,” *IRE Transactions on Information Theory,* pp. 21-28, janeiro 1962.

[43] W. Ryan, *An Introduction to Low-Density Parity-Check Codes,* Universidade do Arizona, 2001.

[44] D. J. C. MacKay, “Good Error-Correcting Codes Based on Very Sparse Matrices,” *IEEE Transactions on Information Theory,* vol. 45, nº 2, pp. 399-431, 1999.

[45] *IEEE Standard for Local and metropolitan area networks,* IEEE Standard 802.16e, 2005.

[46] R. M. Tanner, “A Recursive Approach to Low Complexity Codes,” *IEEE Transactions on Information Theory,* Vols. %1 de %2IT-27, nº 5, pp. 533-547, setembro 1981.

[47] Mathworks, “Decode binary low-density parity-check code specified by parity-check matrix - Simulink,” [Online]. Available: https://www.mathworks.com/help/comm/ref/ldpcdecoder.html. [Acedido em 6 janeiro 2017].

[48] R. G. Lyons, Understanding Digital Signal Processing, New Jersey: Prentice Hall, 2004.

[49] M. Rice, Digital Communications: A Discrete-Time Approach, New Jersey: Prentice Hall, 2009.

[50] Mathworks, “Adaptively adjust gain for constant signal-level output - MATLAB,” [Online]. Available: https://www.mathworks.com/help/comm/ref/comm.agc-class.html. [Acedido em 08 02 2017].

[51] R. E. Best, Phase-Locked Loops: Design, Simulation, and Applications, McGraw-Hill, 2003.

[52] F. M. Gardner, “Interpolation in Digital Modems - Part I: Fundamentals,” *IEEE Transactions on Comunications,* vol. 41, nº 3, pp. 501-507, 1993.

[53] P. Borwein e M. J. Mossinghoff, “Barker Sequences and Flat Polynomials,” em *Number Theory and Polynomials*, Cambridge, Cambridge University Press, 2008, pp. 71-88.

[54] Mathworks, “Uniformly distributed pseudorandom integers - MATLAB randi,” [Online]. Available: https://www.mathworks.com/help/matlab/ref/randi.html. [Acedido em janeiro 2017].

[55] I. Widjaja e A. Leon-Garcia, Communication Networks: Fundamental Concepts and Key Architectures, McGraw-Hill College, 2000.

[56] W. Stallings, Operating Systems: internals and design principles, Prentice Hall, 2000.

[57] D. S. Moore, G. P. McCabe e B. A. Craig, Introduction to the Practice of Statistics, New York: W. H. Freeman and Company, 2009.

[58] ETSI DVB Standard ETR 101 290 V1.3.1 (2014-07), “Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems,” 2014.

[59] F. N. Fritsch e R. E. Carlson, “Monotone Piecewise Cubic Interpolation,” *SIAM Journal on Numerical Analysis,* vol. 17, nº 2, pp. 238-246, 1980.