UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
CÂMPUS DE SOROCABA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IVAN EDUARDO LAGE RODRIGUES

# Desenvolvimento de Software Modular para Criptografia de Sinais em Banda Base

Sorocaba
2022

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
CÂMPUS DE SOROCABA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Ivan Eduardo Lage Rodrigues

# Desenvolvimento de Software Modular para Criptografia de Sinais em Banda Base

Dissertação apresentada à Universidade Estadual Paulista "Júlio de Mesquita Filho" - Campus de Sorocaba como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica pelo Programa de Pós-graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Marcelo Luís Francisco Abbade

Coorientador: Prof. Dr. Luiz Henrique Bonani do Nascimento

Sorocaba
2022

R696d

Rodrigues, Ivan
Desenvolvimento de software modular para criptografia de sinais em banda base / Ivan Rodrigues. -- Sorocaba, 2022
66 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Marcelo Luís Francisco Abbade
Coorientador: Luiz Henrique Bonani de Nascimento

1. Sistemas de Telecomunicação. 2. Criptografia. 3. Processamento de sinais - Técnicas Digitais. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

**Câmpus de São João da Boa Vista**

## CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Desenvolvimento de Software Modular para Criptografia de Sinais em Banda Base

**AUTOR: IVAN EDUARDO LAGE RODRIGUES**
**ORIENTADOR: MARCELO LUÍS FRANCISCO ABBADE**

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica, área: Sistemas Eletrônicos pela Comissão Examinadora:

Prof. Dr. MARCELO LUÍS FRANCISCO ABBADE (Participaçao Virtual)
Coordenadoria de Curso de Engenharia Eletronica e de Telecomunicacoes / Faculdade de Engenharia de Sao Joao da Boa Vista UNESP

Prof. Dr. IVAN ARITZ ALDAYA GARDE (Participaçao Virtual)
Coordenadoria de Curso de Engenharia Eletronica e de Telecomunicacoes / Faculdade de Engenharia de Sao Joao da Boa Vista UNESP

Prof. Dr. JORGE DIEGO MARCONI (Participaçao Virtual)
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas. / Universidade Federal do ABC

São João da Boa Vista, 03 de novembro de 2022

Priscila de Oliveira dos Santos
Supervisora Técnica de Seção - Substituta
Seção Técnica de Graduação e Pós-Graduação

Dedico este trabalho a minha família, meus amigos e minha esposa, que tanto me deram todo o suporte na minha jornada em busca de uma carreira científica.

# AGRADECIMENTOS

Agradeço aos meus pais, Ana Lúcia e Imar Eduardo, por terem me dado todo o suporte para correr atrás dos meus sonhos, assim como todo o amor que me deram por toda a minha vida.

Agradeço também a toda o resto da minha família e meus amigos. Não há melhor prova de que é impossível viver sozinho do que a companhia de boas pessoas, que te motivam a ser uma pessoa melhor todo dia.

À minha esposa, Stefany, que também foi fundamental pelo seu apoio emocional e afeto durante esta jornada.

Ao meu orientador Marcelo Luis Francisco Abbade e meu coorientador Luiz Henrique Bonani por todo o apoio e ensinamentos, os quais deram fruto a este trabalho. Também agradeço a todos os outros professores que tive prazer de cursar suas disciplinas.

Ao meu colega de mestrado, Pedro Pareto, por todo o seu apoio e a amizade que criamos durante esses anos.

E por fim, a toda a instituição da Universidade Estadual Paulista "Júlio de Mesquita Filho" pela oportunidade de contribuir com a pesquisa e ciência doBrasil.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“Informação é a resolução da incerteza.”
— Claude Elwood Shannon

# RESUMO

A pesquisa para criação de técnicas de criptografia seguras mostra-se tornando vital para impedir tentativas de interceptação e quebra do sigilo das mensagens. Nas telecomunicações, o modelo de referência de Sistemas Abertos de Interconexão (*Open Systems Interconnection*, OSI) é contemplado por técnicas de criptografias comercialmente viaveis em todas as suas camadas, exceto a Camada Física. Esta vulnerabilidade é preocupante. Especialmente considerando que é relativamente fácil espiar o conteúdo desta camada utilizando dobras na fibra ou captura de pacotes emsistemas sem fio, por exemplo. Durante décadas foram estudadas métodos de criptografia de sinais para solucionar este problema, e mais recentemente, temos o advento da criptografia espectral. Porém, o uso desta técnica em sinais modulados mostra-se complexa e custosa. Uma abordagem recente que vemsendo estudada é usar o processamento digital de sinais (*digital signal processing*,DSP) para implementar a criptografia espectral em sinais em banda-base. Desta forma, surgiu uma nova abordagem de criptografia de sinais, que chamamos de criptografia de amostragem espectral. Torna-se, então, cada vez mais necessário a existência de softwares de simulação que sejam flexíveis ao surgimento de novas estratégias de encriptação e decriptação de sinais em banda-base. Para isto, é necessária a escolha de uma arquitetura de software versatil em relação às múltiplas estratégias de criptografia de sinais. Neste trabalho, propomos um software, chamado KryptoSJPy, utilizando uma arquitetura híbrida de multicamadas e modelo-visão-controlador (MVC) para atender as necessidade de simulações específicas de criptografia de amostragem espectral. Escolhemos a codificação espectral de fase (*Spectral Phase Encoding*,SPE) e o embaralhamento intracanal (*Spectral Shuffling*, Scr) para testar as primeiras versões do software. Apresentamos também algumas simulações e a comparação com resultados obtidos em códigos anteriores realizados em Matlab.



**Palavras–chave:** criptografia de sinais; criptografia de amostragem espectral; processamento digital de sinais; arquitetura de software.

# ABSTRACT

The research to create secure cryptography techniques have shown to be vital in the prevention of message interceptions and breach of secrecy. In telecommunications, the Open System for Interconnections (OSI) reference model is contemplated by commercially viable cryptography techniques in every layer, except the Physical Layer. This vulnerability is alarming. Especially considering that spying the contents of the Physical Layer is relatively easy with the aid of fiber bending or packet capture in wireless systems, for instance. For decades, methods of signal cryptography have been studies to solve this problem, and recently, there has been the breakthrough of spectral cryptography. However, using this technique in modulated signals turned out to be difficult and costly. A very recent approach that is being researched is using digital signal processing (DSP) to implement spectral cryptography in sampled base-band signals. As such, a new approach to signal cryptography has emerged, which we call Spectral Sampling Cryptography. Thus, it becomes necessary the existence of simulation softwares which are flexible to the advent of new strategies of encryption and decryption of base-band signals. In this dissertation, we propose a software, named KryptoSJPy, using a hybrid multi- layer and model-view-controller architecture to address the necessecity of specific simulations of Spectral Sampling Cryptography. We choose to test the Spectral Phase Encoding (SPE) and Spectral Shuffling (Scr) in the early versions of the software. We also present the analysis of some simulations and the comparation with results obtained by other codes made in Matlab.



**Palavras–chave:** signal cryptography; spectral sampling cryptography; digital signal processing; software architecture.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE ABREVIAÇÕES E SIGLAS

| | |
|---|---|
| AWGN | *Additive white Gaussian noise* |
| BER | *Bit error ratio* (Taxa de erro de bit) |
| BPSK | *Binary phase shift keying* (Modulação por deslocamento de fase binária) |
| DSP | *Digital signal processing* (Processamento digital de sinais) |
| FEC | *Forward error correction* (Correção de erros antecipada) |
| FFT | *Fast Fourier transform* (Transformada rápida de Fourier) |
| IDE | *Integrated development environment* (Ambiente de Desenvolvimento Integrado) |
| iFFT | *Inverse fast Fourier transform* (Transformada rápida inversa de Fourier) |
| MVC | Modelo-visão-controlador |
| OSI | Open Systems Interconnection |
| PEP | *Python Enhancement Proposals* (Propostas de Enriquecimento do Python) |
| PRBS | *Pseudo-Random Bit Sequence* (Sequência de bits pseudoaleatória) |
| PSF | *Python Software Foundation* (Fundação de Software de Python) |
| QAM | *Quadrature amplitude modulation* (Modulação de amplitude em quadratura) |
| QKD | *Quantum Key Distribution* (Distribuição de Chave Quântica) |
| QPSK | *Quadrature phase shift keying* (Modulação por deslocamento de fase em quadratura) |
| Scr | *Scrambling* (Embaralhamento Intracanal) |
| SLM | *Spatial light modulator* (Modulador espacial de luz) |
| SNR | *Signal-noise ratio* (Razão sinal-ruído) |
| SPE | *Spectral Phase Encoding* (Codificação espectral por fase) |

# LISTA DE SÍMBOLOS

| | |
|---|---|
| $n_{aps}$ | Número de amostras por símbolo |
| $n$ | Numero de amostras do sinal |
| $N_s$ | Número de símbolos |
| $s[n]$ | Sinal digital complexo |
| $B$ | Banda do sinal |
| $R_s$ | Taxa de símbolos |
| $r$ | Fator de roll-off |
| $P_{noise}$ | Potência do ruído |
| $\sigma_n^2$ | Variância do ruído |
| $\sigma_{n_{re}}^2$ | Variância do ruído que ocorre na componente real do sinal |
| $\sigma_{n_{im}}^2$ | Variância do ruído que ocorre na componente imaginária do sinal |
| $P_{signal}$ | Potência do sinal |
| $\sigma_{s[k]}^2$ | Variância do sinal |
| $SNR_{dB}$ | Razão sinal-ruído em dB |
| $\Phi_{\mathrm{n}}$ | Fase aleatória da n-ésima amostra do sinal |
| $C[n]$ | Sinal criptografado |
| $k_f$ | Chave do sistema encriptado por SPE |
| $k_s$ | Chave do sistema encriptado por Scr |

# SUMÁRIO

# 1 INTRODUÇÃO

## 1.1 Conceitos fundamentais

Em um mundo em que informações, especialmente dados privados, estão cada vez mais digitalizados, vemos uma urgência em evoluir as medidas de segurança destes dados. A criptografia moderna é definida na literatura como o estudo de técnicas de codificação de dados que garantam sua segurança contra espionagem (KATZ; LINDELL, 2020). Estas técnicas permitem a comunicação entre pares de forma confidencial. A necessidade de boas técnicas de codificação existe desde a Antiguidade, e seguiu influente por outros períodos da história. Temos como exemplo famoso a decifração das máquinas Enigma, durante a Segunda Guerra Mundial (ALEXANDER, 1945). A partir do final dos anos 70, a popularização de computadores pessoais estimulou a implantação de novas técnicas de criptografia de dados, baseadas nos trabalhos seminais de Shannon e de outros grandes pesquisadores da área (SHANNON, 1949). Estes avanços nos estudos da criptografia a elevaram como área do conhecimento, levando-a aos patamares atuais de rigor matemático e robustez da segurança de seus algoritmosmais famosos.

Quando usamos a palavra “criptografia” atualmente é comum pensar em criptografia de dados, devido a seu uso ubíquo na Internet. Seja na encriptação de senhas de autenticação de usuário, de mensagens privadas ou de arquivos inteiros, esta ferramenta mostra-se indispensável para garantirmos a segurança e privacidade de nossos dados. Se considerarmos nas telecomunicações o modelo de referência de Sistemas Abertos de Interconexão (Open Systems Interconnection, OSI), atualmente todas as suas camadas podem criptografar informações, com exceção da Camada Física, que converte dados em sinais. Esta falta de criptografia na Camada Física a torna um ponto vulnerável das telecomunicações modernas. Esta vulnerabilidade da Camada Física inspirou a criação técnicas de encriptação de sinais, que neste trabalho chamaremos de criptografia de sinais.

Neste trabalho desenvolvemos um software flexível à simulação e análise de múltiplas estratégias de criptografia chamado KryptoSJPy. Mais especificamente, será descrita a criação de um software que serve para

testar diversas técnicas de criptografia de sinais, que possibilite uma versatilidade na implantação dos códigos das novas técnicas de criptografia de sinais que estão surgindo. Discutiremos então os resultados das simulações feitas a partir deste software, e comparação dos resultados de outros trabalhos do nosso grupo de estudos. Também será analisada a precisão em relação a resultados esperados pela teoria e a literatura estudada sobre o assunto até o momento.

Este software foi especificamente escrito na linguagem Python. A linguagem Python foi criada no final da década de 80 por Guido van Rossum nos Países Baixos. Atualmente, seu desenvolvimento é feito continuamente pela Fundação de Software de Python (*Python Software Foundation*, PSF). A linguagem é de código aberto (*open source*), o que significa que a linguagem é gratuita e todo o seu código fonte é público. A licença *open source* também dita que todo trabalho derivado da linguagem Python deve ser *open source*. Outra vantagem do Python é a vasta quantidade de bibliotecas (ou módulos) internas e externas que adicionam mais funções, tipos de dados e outros recursos uteis aos nossos objetivos.

## 1.2 **Objetivos do trabalho**

Temos como objetivo criar um software de criptografia desenvolvido em Python, que seja flexível a múltiplas estratégias de codificação. Mais especificamente, gostaríamos de demonstrar a possibilidade de criação de um software que sirva para testes de diversas técnicas de criptografia de sinais. Nomeamos este software “KryptoSJPy”, uma amálgama das palavras *cryptography* (Criptografia em inglês), São João da Boa Vista e Python.

Focaremos a princípio na descrição da arquitetura do KryptoSJPy em Python, utilizando um modelo híbrido de multicamadas e modelo-visão-controlador (MVC). Em seguida, analisaremos os resultados obtidos por algumas simulações do software, comparando-as com resultados de simulações idênticas feitas no Matlab por outros membros do nosso grupo de estudos.

As simulações deste trabalho serão referentes a comunicação de um ou múltiplos sinais entre um transmissor e um receptor pelo uso de sinais criptografados.

A criptografia é realizada em sinais complexos em banda base, que se

originam em sinais com modulação por deslocamento de fase binária (*binary phase shift keying*, BPSK), modulação por deslocamento de fase em quadratura (*quadrature phase shift keying*, QPSK) e modulação de amplitude em quadratura (*quadrature amplitude modulation*, QAM) com 16 símbolos distintos (16-QAM).

# 2 PRINCÍPIOS DA CRIPTOGRAFIA

Neste capítulo serão definidos alguns conceitos centrais de criptografia. A Seção 2.1 abordará os conceitos básicos de criptografia de dados. Na Seção 2.2 utilizaremos estes conceitos para definir o que é a criptografia de sinais. Nas Seções 2.3 e 2.4 discutiremos sobre dois métodos de criptografia que podem ser utilizados na Camada Física, as criptografias quântica e caótica. Por fim, na Seção

2.5 e 2.6 descreveremos a criptografia espectral e sua versão digital, a criptografia de amostragem espectral.

## 2.1 Criptografia de dados

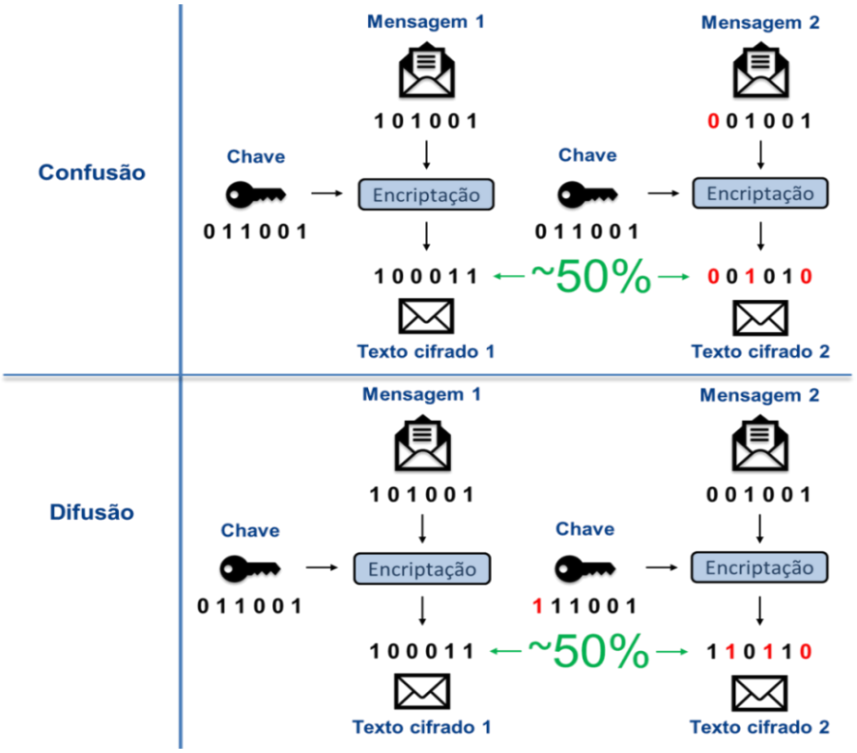
Um dos princípios fundamentais da criptografia é que uma mensagem é transformada em um texto inteligível a terceiros. Chamaremos o resultado desta transformação de texto encriptado ou texto cifrado. O processo desta transformação é chamado de encriptação, ou cifra. Este texto cifrado é trocado entre os participantes da comunicação. A encriptação depende de um conjunto secreto de dados, chamado de chave de encriptação. Esta chave é usada para converter esta cifra de volta à mensagem original, um procedimento que chamamos de desencriptação. Quando a mesma chave é usada nos processos de encriptação e de desencriptação, diz-se que a chave é simétrica. Por outro lado, quando uma chave é usada para encriptar os dados e outra é usada para desencriptá-los, diz-se que as chaves são assimétricas. Neste trabalho utilizaremos apenas o cenário de chave simétrica.

O objetivo fundamental da encriptação é dificultar, e, idealmente, impedir, a compreensão do conteúdo das mensagens trocadas pelas partes que se identificam e se reconhecem explicita e reciprocamente como legítimas na comunicação em questão. Chamaremos de espiões ou *eavesdroppers* todos aqueles que tentem acessar o conteúdo dessas mensagens clandestinamente, ou seja, sem avisar explicitamente às partes legítimas sobre sua participação naquelasessão de comunicação. Para os fins deste trabalho, consideramos que a análise da segurança de um sistema de encriptação deve basear-se na suposição de que os espiões conhecem todas as informações sobre este sistema, exceto a chave criptográfica usada nas comunicações. Essa suposição

é chamada de Princípio de Kerckhoff, em homenagem ao criptógrafo holandês Auguste Kerckhoff que a formulou no século XIX.

Existem várias propriedades que precisam ser satisfeitas para que uma técnica de criptografia seja segura. Dentre elas, destacamos as propriedades de difusão e confusão que foram definidas por Shannon (SHANNON, 1949). Na difusão, uma pequena mudança na mensagem deve causar uma grande alteração no texto cifrado. Enquanto na confusão, temos que ao causar uma pequena alteração na chave, há uma grande alteração no texto cifrado. Por exemplo, a confusão pode ser analisada avaliando-se a diferença entre os bits de textos cifrados gerados a partir de chaves que difiram entre si por apenas um bit. Idealmente, essa diferença deve gerar textos cifrados que difiram entre si em 50% dos bits. Se esse número fosse maior, um inversor poderia ser usado para reduzi- lo. Na propriedade de difusão, são analisados os textos cifrados entre mensagens que tenham um bit de diferença entre suas mensagens. Novamente, espera-se que essa diferença produza textos cifrados que difiram entre si em 50% dos bits. Os métodos de confusão e difusão são ilustrados na Figura 1.

Figura 1 - Propriedades de confusão e difusão.

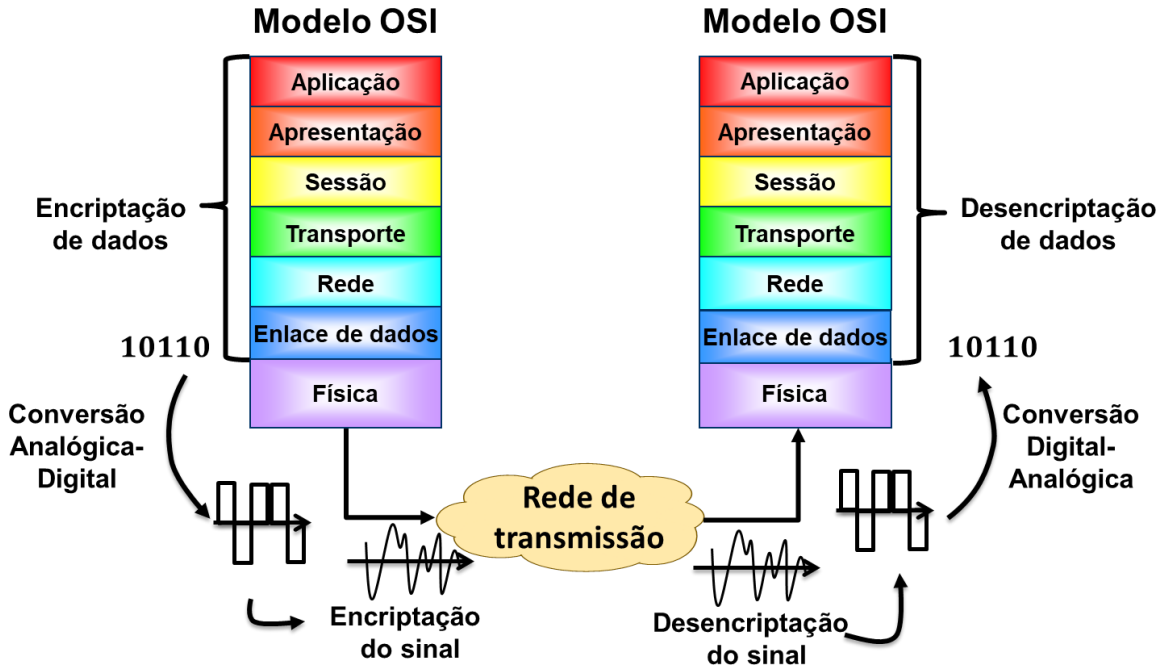



Fonte: Elaborado pelo autor.

## 2.2 Criptografia de sinais

Como dito anteriormente, a vulnerabilidade na camada física tornou a criação de uma criptografia para um sinal na Camada Física um assunto relevante. Mesmo em comunicações ópticas, existem métodos não intrusivos, baseados em acopladores ópticos de grampo (*clip-on couplers*) para espionar os sinais que trafegam pelas fibras. Um destes métodos procura dobrar a fibra (*fiber bending*) para induzir o vazamento da luz através da casca da fibra. Caso o espião seja proficiente e não danifique a fibra, ele pode adquirir informações do sinal com perdas mínimas (ZAFAR; FATHALLAH; BELHADJ, 2010). Além disto, é difícil detectar ataques de espionagem na fibra, e a criação de métodos de prevenção ainda é uma área em desenvolvimento (DAHAN; MAHLAB, 2017; SHUAI, 2019; FURDEK *et al.* 2021). Uma solução viável para alguns desses problemas, e indicada na literatura, é encriptar os sinais enviados pela fibra, de forma que o espião receba um sinal ininteligível.

Figura 2 - Aplicação da criptografia de sinais ao modelo de referência OSI.




Fonte: Elaborado pelo autor.

Definiremos a criptografia de sinais como a união de métodos de criptografia com telecomunicação de sinais. Ou seja, a criptografia de sinais é a criptografia da Camada Física que encripta e desencripta a sequência de bits que é transmitida para o meio físico. Na Figura 2 temos um exemplo da implementação da criptografia de sinais no modelo OSI.

Dentre os métodos de criptografia na Camada Física que estão sendo estudadas recentemente destacam-se, para os fins deste trabalho, a criptografia quântica e a caótica. Nas seções seguintes, discutiremos brevemente sobre estas cifras. Também existe a criptografia que é aplicada diretamente ao espectro da frequência do sinal a ser enviado, chamada de criptografia espectral. Porém, neste trabalho focaremos na criptografia espectral que utiliza métodos de processamento digital de sinais (*Digital signal processing*, DSP) para a realização dessa criptografia de sinais em banda-base. Chamamos este método de criptografia de amostras espectrais.

## 2.3 **Criptografia quântica**

Definimos a criptografia quântica como o uso de propriedades quânticas na comunicação digital, com o propósito de elevar sua segurança (BENNETT; BRASSARD, 1984). O método mais famoso de criptografia quântica é sua aplicação para a transmissão de chaves criptográficas, consagrada por Charles H. Bennett e Gilles Brassard em meados dos anos 80. Seu protocolo, nomeado BB84, tornou-se o pilar para o conceito contemporâneo de distribuição de chave quântica (*Quantum Key Distribution*, QKD). Na QKD a encriptação é feita para permitir a troca de chaves entre transmissor e receptor. Note que a QKD encripta o sinal da chave da transmissão da mensagem, não o sinal da mensagem em si. Em geral, esta chave produzida por QKD é chamada de “chave secreta”, e sua informação é codificada em fótons de polarizações binárias ou qubits. Qubits estão sempre em superposições de seus estados quânticos, tornando difusa sua informação. Ao detectar e medir o qubit, colapsamos seu estado quântico e validamos sua informação (SCHUMACHER, 1995). Graças ao comportamento quântico dos qubits enunciado pelo teorema da não clonagem, a espionagem de sua informação é impossível sem sua interferência. O teorema enuncia que dado dois estados quânticos distintos, não é possível copiar a superposição de seus estados com acurácia (WOTTERS; ZUREK, 1982).

Os detalhes da transmissão da QKD variam de protocolo para protocolo, por isto vamos descrever o protocolo exemplar: BB84. Neste protocolo, a chave é enviada por um usuário, tradicionalmente chamado de “Alice”, como uma sequência aleatória de bits por um canal quântico na forma de fótons que

são codificadas em polarizações com bases ortogonais de $0°$ ou $90°$, ou bases diagonais de $45°$ ou $135°$. Estas bases podem ser escolhidas por Alice de forma predeterminada ou de forma aleatória. Os fótons são recebidos pelo receptor, usualmente chamado de “Bob”, os quais seus medidores podem ter bases aleatórias ou predeterminadas para detectar os fótons.

Devido à natureza quântica dos fótons, sua informação é desconhecida até que seja medido. Caso o fóton seja medido por uma base paralela a sua polarização, ele é transmitido/recebido com sucesso, ou seja, seu valor é absoluto. Caso a base do medidor esteja perpendicular à polarização do fóton, ele é absorvido e seu valor é nulo. Portanto, consideraremos que na polarização ortogonal, o bit 0 é codificado por um fóton com polarização de $0°$, e o bit 1 por um fóton com polarização de $90°$. Na polarização diagonal o bit 0 é codificado por um fóton de polarização $45°$, e o bit 1 por um fóton com polarização $135°$. Seria se um fóton de polarização perpendicular e é decodificado por uma base diagonal, o valor do bit gerado tem 50% de chances de ser 1 ou 0, devido a superposição do estado quântico do fóton. O mesmo ocorre caso o fóton de polarização diagonal é polarizado por uma base perpendicular.

Após a detecção completa dos bits, Alice e Bob devem comunicar entre si, por um canal público, as bases usadas por seus transmissores e receptores. Os bits que Alice e Bob acertaram entre si são considerados como a chave da transmissão. Os bits recebidos por Bob que vieram de polarização perpendicular e decodificado por uma base diagonal, ou vice-versa, são descartados. Este protocolo está ilustrado na Figura 3.

Figura 3 - Exemplo de uma transmissão segundo o protocolo BB84.

| Transmissor da Alice | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Bits enviados | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| Base | ⊕ | ⊗ | ⊗ | ⊗ | ⊕ | ⊗ | ⊕ | ⊗ | ⊕ | ⊕ |
| Transmissão no canal quântico | | | | | | | | | | |
| Polarização dos fótons | → | ↖ | ↖ | ↗ | → | ↗ | ↑ | ↖ | ↑ | → |
| Receptor do Bob | | | | | | | | | | |
| Base | ⊕ | ⊗ | ⊕ | ⊕ | ⊗ | ⊗ | ⊕ | ⊗ | ⊗ | ⊕ |
| Polarização dos fótons | → | ↖ | → ou ↑ | → ou ↑ | ↗ ou ↖ | ↗ | ↑ | ↖ | ↗ ou ↖ | → |
| Bits recebidos | 0 | 1 | 0 ou 1 | 0 ou 1 | 0 ou 1 | 0 | 1 | 1 | 0 ou 1 | 0 |
| Transmissão no canal público | | | | | | | | | | |
| Chave pública | 0 | 1 | | | | 0 | 1 | 1 | | 0 |

Fonte: Elaborado pelo autor.

O Teorema da Não Clonagem garante segurança à chave pública, pois um espião que tente medir os fótons com qualquer base irá colapsar seu estado quântico, evitando determinar seu valor binário. Tentativas de espionagem podem ser detectadas por Alice e Bob comparando um número aleatório de bits detectados com sucesso por suas bases. Se houver discordância em relação à distância e à taxa de transmissão de bits, houve interferência e possivelmente espionagem. Ou seja, podemos detectar falhas nesta comunicação com qualquer razão de erro de bits (*Bit error ratio*, BER) abaixo de 100% (BENNETT; BRASSARD, 1984).

Apesar de sua segurança ser forte, a criptografia quântica enfrenta problemas em relação a sua distancia de transmissão e sua taxa de

transmissão de bits (YIN *et al.*, 2016). Limites teóricos mostram taxas de transmissão de bits abaixo de $10\ {}^{kb}/_{s}$ para distancias de 250 km de fibra (LUCAMARINI, 2018). Estes valores estão muito abaixo se comparados com valores de transmissões em

sistemas comerciais de comunicação por fibra óptica. Além disto, o teorema da não clonagem impede o uso de amplificadores clássicos para aumentar a distancia da transmissão. Entretanto, soluções estão sendo estudadas para mitigar os problemas presentes na transmissão a partir do uso, por exemplo, de amplificadores quânticos (BRIEGEL *et al.*, 1998; SANGOUARD *et al.*, 2009), satélites (LIAO *et al.*, 2017), e esquemas de geração de fótons gêmeos de distribuição de chave quântica (*Twin-field Quantum Key Distribution*, TF-QKD) (LUCAMARINI, 2018). Uma análise profunda das soluções atuais da criptografia quântica foge do escopo deste trabalho, entretanto seu campo de estudo mostra- se bastante ativo até o momento desta escrita (XU *et al.*, 2020; ZHANG *et al.* 2022). As buscas sucessivas por métodos de mitigar os problemas deste método de criptografia trarão inovações à segurança de envio de chaves de criptografia na camada física.

## 2.4 **Criptografia caótica**

Podemos considerar “criptografia caótica” como um termo que unifica vários métodos de criptografia que utilizam sistemas caóticos multidimensionais, como base para a encriptação de uma mensagem (ALVAREZ, 2006). A literatura separa sistemas de criptografia caóticos em duas bases: sincronização caótica e sistemas digitais.

Nos sistemas baseados em sincronização caótica, são utilizados métodos analógicos de sincronização de dois ou mais sistemas caóticos em relação a um fenômeno físico, como um sinal de acoplamento (PECOLA; CARROLL, 1990), a retroalimentação de entrada dos sistemas (RULKOV et al., 1995) ou a sincronização das fases de sistemas rotatórios (ROSENBLUM; PIKOVSKY; KURTHS, 1996). Os tipos de sincronizações caóticas e seus detalhes matemáticos são um tema que foge do escopo deste trabalho. A sincronização caótica permite utilizar sinais gerados a partir de sistemas caóticos, como portadoras para modular a onda do sinal da mensagem. A desencriptação do

sinal é feita após a sincronização entre o transmissor e o receptor, o que permite a separação do sinal da portadora e da mensagem. Até recentemente, sistemas de sincronização caótica continuam sendo estudados e testados para implementar em redes ópticas (KE *et al.*, 2016; YI *et al.*, 2018) e até mesmo combinando com QKD (JIANG *et al.* 2020).

Já os sistemas de criptografia caóticos digitais, ou cifras caóticas digitais, utilizam algoritmos computacionais que calculam resultados de equações de sistemas caóticos para criptografar mensagens. Nem toda cifra caótica digital é exclusivamente uma criptografia de sinais. Existem aplicações de cifras caóticas digitais para cifra de bloco, cifras de fluxo, geradores de números pseudoaleatórios, funções de *hash*, marcas d'água de imagens, encriptação de vídeos, assinaturas digitais e várias outras aplicações que fogem do escopo deste trabalho (TEH *et al.*, 2020). Porém focaremos nos sistemas de criptografia caótica digital que utilizam DSP para encriptar mensagens de sinais na camada física. Podemos generalizar abordagens de cifras caóticas digitais pelo uso de equações diferenciais multidimensionais para realizar a encriptação de múltiplos blocos de processamento do sinal. Em geral, estas equações são compostas por múltiplas constantes e múltiplas variáveis. Para iniciar sua resolução, é necessária a escolha de condições iniciais. A resolução da equação a partir das condições iniciais torna-se a primeira chave do primeiro bloco de processamento do sinal. Estes valores podem ser usados para realizar modulações no sinal, como, por exemplo, deslocamentos em fase, permutação no tempo e quadratura (YANG *et al.*, 2016; SULTAN *et al.* 2018). Os valores da chave são considerados como as condições iniciais para o segundo bloco de processamento do sinal, e assim por diante até que todos os blocos sejam encriptados. A desencriptação é feita utilizando os mesmos parâmetros para realizar demoduçaões no sinal.

Sistemas caóticos parecem aleatórios ou imprevisíveis pela volatilidade de seus resultados com diferentes condições iniciais. Entretanto, devemos lembrar que estes sistemas são determinísticos, o que trás uma grande falha a sua segurança. Além disto, seus esquemas podem ser considerados altamente complexos, muito específicos e/ou carentes de rigor analítico-criptográfico. Após décadas de críticas aos métodos de sistemas de criptografia caóticos, diretrizes e boas práticas vem sendo delimitadas para a criação de sistemas criptográficos robustos (TEH *et al.*, 2020).

Pesquisadores que trabalham com criptografia caótica estão aprimorando as técnicas estudadas para superar as críticas encontradas na literatura. No entanto, hoje, no melhor de nosso conhecimento, não há nenhum sistema que use acriptografia caótica comercialmente.

## 2.5 **Criptografia espectral**

A criptografia espectral é o método de criptografia de sinais no qual a encriptação é realizada no espectro do sinal. Este espectro é dividido em fatias espectrais, que são divisões do sinal óptico a partir do uso, por exemplo, de demultiplexadores ópticos. A primeira técnica a ser desenvolvida neste meio foi a codificação espectral de fase (*Spectral Phase Encoding, SPE*) (CORNEJO; TOCNAYE, 2008). Na SPE, é empregado um arranjo de encriptação 4-F para a criação de fatias espectrais. Neste arranjo duas grades de difração são separadas por duas lentes intercaladas por um modulador espacial de luz (*light spatial modulator*, SLM). Cada componente esta distante entre si pela distância focal $f$ da lente. Neste arranjo, o sinal optico que inside na grade de difração é separado em varias fatias. Essa estratégia permite a geração de mais de 200 fatias. As fatias são convergidas para o SLM ao passarem pela lente. Ao incidirem nos pixels do SLM, as fases de cada fatia são alteradas aleatoriamente por uma corrente elétrica. O conjunto das alterações das fatias formam a chave do sistema. Por fim, as fatias são convergidas por outra lente a uma grade de difração que as junta em um unico sinal novamente. A segurança da SPE pode ser incrementada pelo uso de aplicação de atrasos as fatias (ABBADE *et al.*, 2015; ABBADE *et al.*, 2017), técnica nomeada como codificação espectral de fase com atraso (*Spectral Phase and Delay Encoding*, SPDE). E por fim existe a técnica de embaralhamento entre de fatias de múltiplos canais, ou embaralhamento intercanal (*Shuffling*) (ABBADE *et al.,* 2019; ABBADE *et al.,* 2020).

Existem grandes vantagens na criptografia espectral, como a banda do sinal criptografado se manter idêntica ao sinal original, assim como sua taxa de transmissão manter-se próxima a das taxas utilizadas em redes comerciais de telecomunicação. Entretanto, encriptação de sinais modulados requer o uso de hardware específico, como o SLM, que introduz um custo adicional ao sistema de

comunicação. Além disso, a utilização desse hardware, normalmente, expõe o sinal a certos tipos de degradação que comprometem o desempenho da criptografia de sinal. Por exemplo, existe uma região opaca entre os píxels do SLM, que ao ser atingida por uma fatia deteriora o sinal criptografado. Além disso, também no esquema 4-f, flutuações térmicas no laser podem desalinhar a frequência central da portadora ótica. Este desalinhamento pode alterar as fatias consideradas no receptor em relação às usadas no transmissor, comprometendo a qualidade do sinal descriptografado. De fato, à medida que o número de fatias aumenta, essa degradação torna-se mais severa. Isso é, obviamente, inconveniente para a segurança da criptografia, que aumenta proporcionalmente ao número de fatias utilizado.

Esses problemas podem ser resolvidos se os sinais forem criptografados em banda-base no domínio elétrico e, posteriormente, modulados para serem transmitidos no domínio óptico. No lado do receptor, o sinal pode ser demodulado e descriptografado em banda-base. De fato, nessa abordagem, a criptografia pode ser realizada por DSPs. Chamaremos esta abordagem de criptografia de amostragem espectral (*Spectral sampling cryptography*, SSC).

## 2.6 **Criptografia de amostragem espectral**

A criptografia de amostragem espectral tem como estratégia utilizar processamento digital de sinais para criptografar o sinal em banda-base. Esta abordagem contorna os problemas associados à criptografia espectral por realizar a encriptação no domínio elétrico ao invés do domínio óptico. A DSP já faz parte de sistemas de comunicação de alto desempenho. Além disso, a utilização de DSPs é extremamente flexível, permitindo a realização de operações matemáticas por meio de uma programação adequada. Outra vantagem importante é que, em condições térmicas típicas, o desempenho dos DSPs do transmissor e do receptor será o mesmo, independentemente da temperatura. Todas essas características mostram que a utilização de DSPs para criptografar sinais em banda-base que podem, posteriormente, serem modulados, é extremamente promissora.

O estudo de um esquema otimizado para esta criptografia ainda está em andamento, por este motivo este trabalho vai considerar um esquema

generalizado de criptografia de amostragem espectral. Esperamos que a criação do KryptoSJPy intensifique o andamento destes estudos. Consideramos uma mensagem composta por bits que é mapeada a um sinal digital real ou complexo com $N_s$ simbolos. Cada símbolo é composto por um numero $n_{aps}$ de amostras, resultando em um sinal de $n$ amostras tal que:

$$n = N_s * n_{aps}. \quad (1)$$

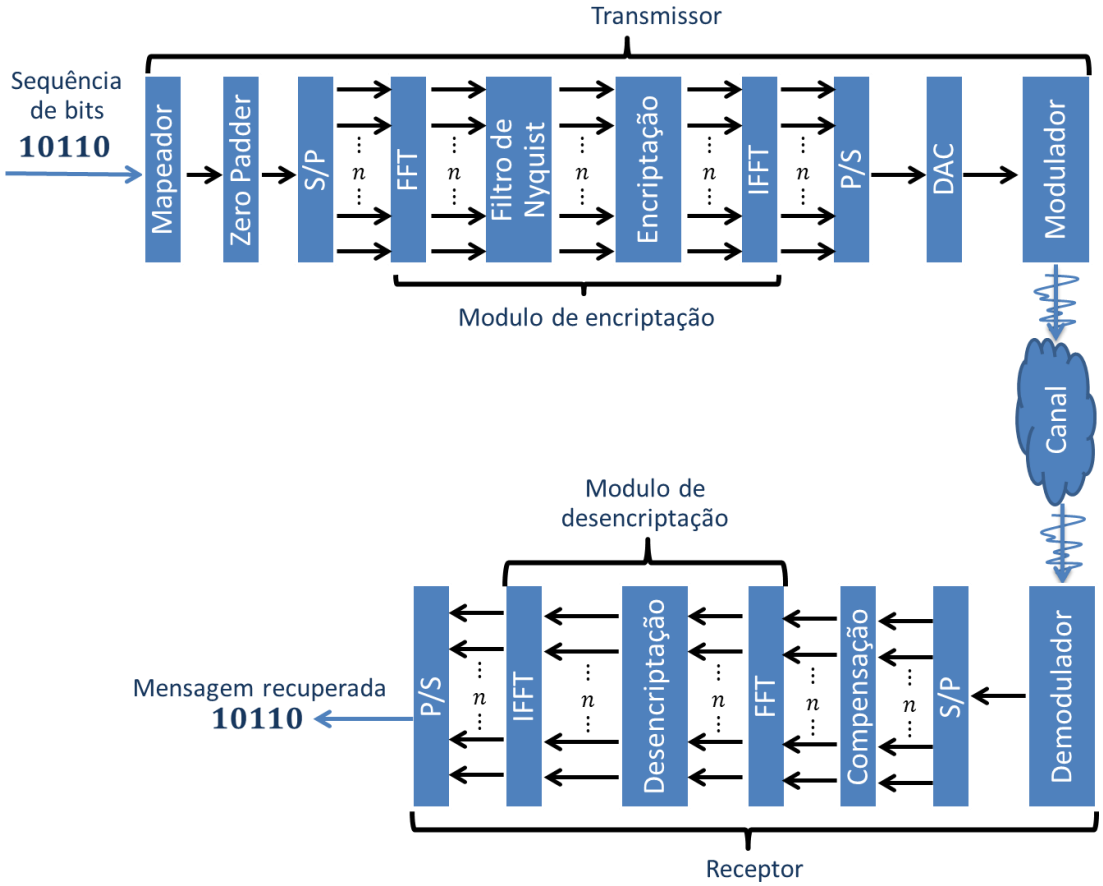
Para combatermos interferência intersimbólica (*intersymbol interference*, ISI), todas as mostras de cada símbolo menos a primeira são zeradas. Este processo é chamado de *zero padding* é realizado por um bloco de mesmo nome. As $n$ amostras são paralelizadas por um conversor série-paralelo (S/P). Esta mensagem passará por um processador digital de sinais customizado que inclui o hardware necessário para encriptar ou desencriptar a mensagem. Para simplificar, chamaremos este processador digital de sinais customizado de "módulo de encriptação". Este módulo inclui um algoritmo de transformada rápida (*fast Fourier transform*, FFT), um filtro de Nyquist, o operador de encriptação e um algoritmo de FFT inversa (IFFT). O filtro de Nyqust é usado para tornar plano o espectro do encriptado, garantindo que não haja componente espectral vantajosa para ser espionada. Como consequência secundária, o filtro também compensa a ISI (LATHI; DING, 2012). Neste trabalho focaremos nas simulações de duas operações de encriptação para criptografia de amostragem espectral: a codificação espectral de fase por DSP (SANTOS, 2020), e o embaralhamento intracanal por DSP (*Spectral Scrambling*, Scr) (SANTOS, 2020). Estas criptografias, assim como o sinal criptografado, serão detalhadas no Capítulo 5.

Após a mensagem ser criptografada, as $n$ amostras são serializadas novamente por um conversor paralelo/série. Logo após ocorre a conversão dosinal para um sinal analógico utilizando um conversor digital-analógico (*Digital-analog converter*, DAC), e então ele será enviado pela rede de transmissão.

A desencriptação do sinal é amostrado novamente, e degradações causadas pela transmissão são compensadas por um bloco dedicado a isto. Em seguida o sinal é submetido ao "módulo de desencriptação", que é análogo ao módulo de encriptação descrito anteriormente, porém sem o filtro de Nyqust. A operação de desencriptação análoga a encriptação realizada na transmissão do sinal. Após o sinal ser serializado novamente, a mensagem é recuperada. Todo

este processo é ilustrado na Figura 4.

Figura 4 - Diagrama do módulo de encriptação e desencriptação.




Fonte: Elaborado pelo autor.

É importante que este sinal seja resistente a ruídos, assim como atinja os requisitos de confusão e difusão de Shannon. Podemos analisar os bits ou símbolos (conjunto de bits) dos sinais simulados para verificar a obtenção da confusão e difusão nos processos de criptografia.

# 3 ARQUITETURA DE SOFTWARE

A arquitetura de software é o conjunto de abstrações e fundamentos que descrevem o software. Sua documentação auxilia na organização do software, permitindo-nos vê-lo como um sistema de funções e componentes inter-relacionados. Quando uma arquitetura for detalhadamente analisada e documentada podemos chama-la de padrão de arquitetura ou modelo de arquitetura. Ao aplicar este padrão de forma mais abrangente ao projeto de software, ou a outros tipos de projetos, podemos chama-lo de padrão de projeto ou *design patterns*. Estes padrões trazem boas práticas de desenvolvimento paraa organização de um projeto (SOMMERVILLE, 2011).

Para desenvolver o software deste projeto, foi necessário decidir qual padrão de arquitetura do software seria usado como base. Existem vários padrões de arquitetura descritos na literatura. Entre esses, destacam-se no âmbito deste trabalho, os seguintes padrões:

1. Duto e filtro;
2. Orientado a objetos;
3. Baseado em eventos;
4. Repositório;
5. Multicamadas;
6. Modelo-visão-controlador (MVC).

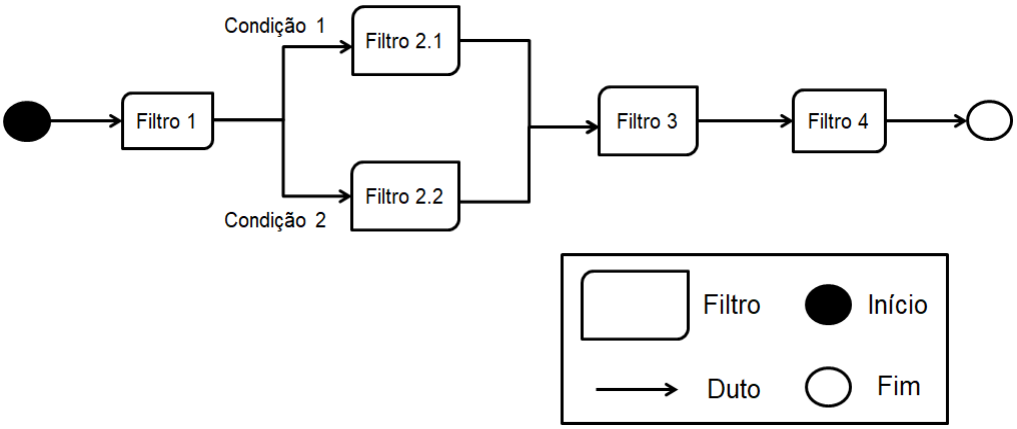
Neste capítulo, estes modelos serão descritos e discutiremos sobre suas vantagens e desvantagens, que guiaram a escolha da arquitetura do software deste trabalho (SHAW; GARLAN, 1996; SOMMERVILLE, 2011; GUTMANN, 2004). Algo comum em todos estes modelos é o uso de componentes abstratos, como dutos, objetos, eventos, dentre outros. Estes componentes são representações de funções ou classes do software que tem tarefas específicas e serão explicadas no decorrer deste capítulo. A distinção entre cada padrão está nas especificações de seus componentes e suas relações.

## 3.1 **Padrão duto e filtro.**

Neste padrão, todo componente transformador de dados do sistema recebe uma entrada e produz uma saída. O nome vem da transmissão de

entradas e saídas, chamadas dutos, serem transformadas pelos componentes, denominados filtros. A Figura 5 representa um padrão de duto e filtro simples, em que um processo qualquer passar por um conjunto de filtros. É importante notar a presença de estruturas condicionais na figura, que são perfeitamente compatíveis a este padrão. Dutos e filtros é um padrão simples de ser implementado em sistemas que tem um comportamento sequencial ou concorrente, além de ser fácil de ser entendido. A natureza independente dos filtros também torna a criação e manutenção de novos filtros uma tarefa mais fácil. Porém, ele é restritivo, pois os dados enviados entre os componentes devem sempre ser do mesmo tipo. Isto significa que se um componente altera o tipo de dado no duto, o mesmo componente não pode ser reutilizado em outra parte dos dutos sem alterar novamente o dado. Outro problema é que todo filtro deve ser transformativo, o que pode tornar sua implementação complexa e ineficiente em certos casos. Um exemplo é o caso em que é desejável ter um componente que apenas exiba os dados de um objeto em um instante.

Figura 5 – Padrão de duto e filtro.



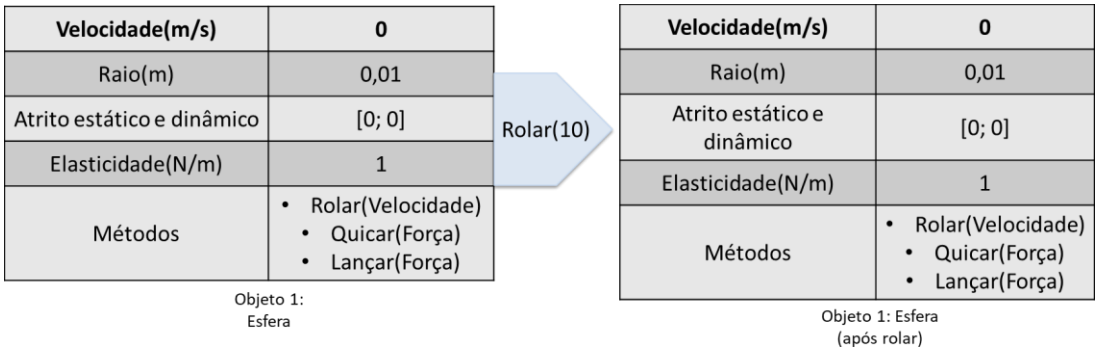

Fonte: Elaborado pelo autor

## 3.2 **Padrão orientado a objetos.**

Neste padrão, os dados do sistema fazem parte de objetos, os quais se comunicam com outros objetos usando mensagens alocadas em suas propriedades. Os objetos podem ter uma ou mais funções, também chamadas de métodos, o que os diferencia dos filtros da arquitetura anteriormente citada, sendo mais flexíveis. A Figura 6 mostra um objeto (abstraído por uma tabela de dados) antes e depois de utilizar um de seus métodos. Objetos podem

guardar informações, assim como mostrar seus dados de forma dinâmica. Eles também são conhecidos pelo seu sistema de “herança”, em que um objeto pode herdar propriedades de outros objetos. Isto torna a criação de novos objetos semelhantes uma tarefa fácil. Em linguagens modernas de computação, objetos são criados a partir do uso de classes. Contudo, a implementação de deste modelo pode ser uma tarefa difícil dependendo do sistema. A escolha de quais elementos do sistema devem se tornar objetos é um dos aspectos mais importantes do modelo (SOMMERVILLE, 2011). Além disto, podem ocorrer problemas ao alterar as propriedades de um objeto, obrigando o programador a checar todos os objetos que dependam dele. Um adendo importante sobre este padrão é que o uso de poucos objetos em certa arquitetura não necessariamente significa usar o modelo orientado a objetos. Ou seja, uma arquitetura é guiada por um modelo orientado a objetos quando o uso de objetos é indispensável para o entendimento de sua organização.

Figura 6 – Padrão orientado a objetos



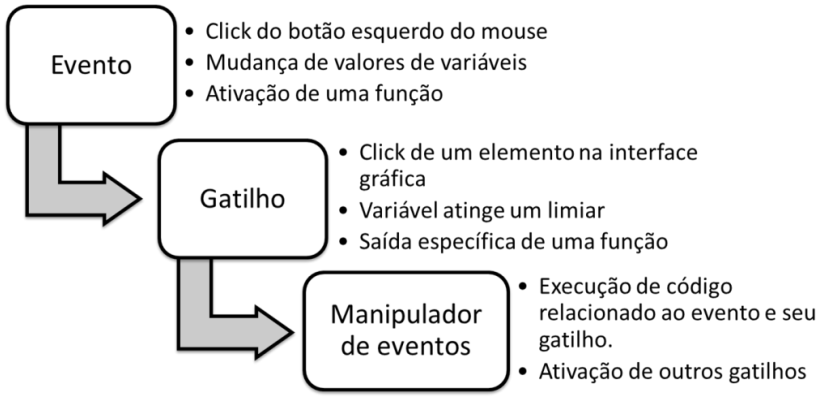

Fonte: Elaborado pelo autor

## 3.3 **Padrão baseado em eventos**

No padrão baseado em eventos, os componentes do sistema chamados de gatilhos reagem a eventos. Estes eventos são quaisquer fenômenos que aconteçam no programa, como mudanças em valores de dados ou o pressionar de botões. Os eventos são mediados por um componente, por um manipulador de eventos (*event handler*). A Figura 7 representa uma arquitetura simples com padrão baseado em eventos com exemplos de eventos e gatilhos. Por exemplo, o evento de clicar com o mouse ativa certos gatilhos como um botão em uma interface gráfica. O manipulador de dados vai então ser acionado

e vai executar o código relevante ao evento. Como vantagem, os componentes reagem de forma implícita, sem a necessidade de interagir entre si. Isto também significa que a criação de novos componentes se torna mais simples, eles apenas precisam ser capazes de reagir aos eventos. Contudo, este modelo pode ser bastante complexode ser implementado em certos casos, exigindo alta abstração do sistema. Outro ponto negativo é que há casos em que componentes distintos podem reagir ao mesmo evento simultaneamente de forma inesperada. Um exemplo deste cenário, no qual múltiplos gatilhos estão ligados ao evento de ativar uma função, e se não houver cuidado no modo como estes gatilhos respondem ao manipulador de eventos podem ocorrer erros na execução do código. Isto dificulta a depuração do código e pode atrapalhar na manipulação de grandes quantidades de dados.

Figura 7 – Padrão baseado em eventos e exemplos de eventos e gatilhos.



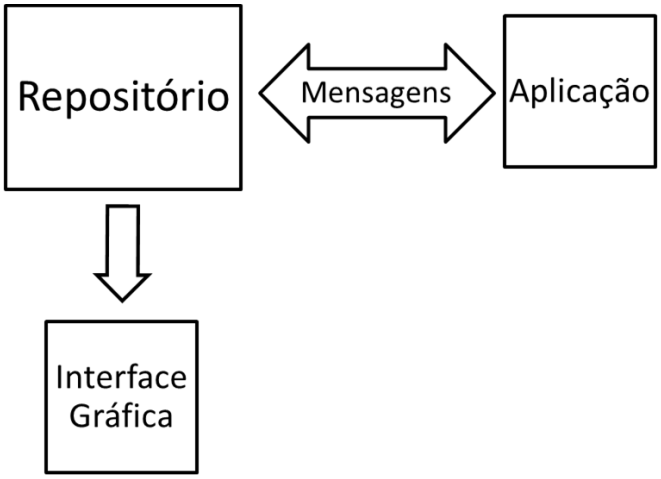

Fonte: Elaborado pelo autor

## 3.4 **Padrão de repositório**

No padrão de repositório temos dois tipos de componentes principais, o banco central de dados (ou repositório), e os componentes que operam os dados do banco. Os dados são gerados e/ou guardados no repositório, e são transformados pelos componentes ligados a ele. Os componentes em torno do repositório podem trocar mensagens entre eles ou não. Na Figura 8 temos o exemplo de um modelo de repositório simples. Neste exemplo, o repositório troca mensagens com uma aplicação, e seus dados são exibidos por uma interface gráfica. Este modelo é ideal para casos em que uma grande quantidade de dados precisa ser manipulada. Porém, também restringe os componentes a fazerem apenas transformações pertinentes ao tipo de dado necessário para o

repositório, semelhantemente ao caso do modelo de dutos e filtros. Outra desvantagem é a dependência ao repositório, o que significa que erros no repositório perpetuam a todos os seus elementos operantes.

Figura 8 – Padrão de repositório.



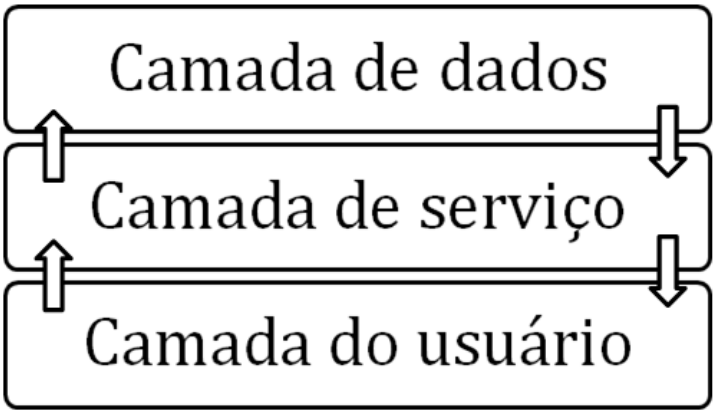

Fonte: Elaborado pelo autor

## 3.5 Padrão de multicamadas

No padrão de multicamadas, cada camada atende um serviço da camada superior, e opera como cliente da camada inferior (GUTMANN, 2004). Isto nos permite modelar um problema complexo em uma série de passos simples. Estes passos serão mediados pela ligação entre as camadas, o número de camadas e suas funções. A ligação entre as camadas é, geralmente, um protocolo, algoritmo ou função que determina como as camadas interagem. Este modelo tem a vantagem de suas camadas serem independentemente modificáveis, desde que o código principal seja mantido (SOMMERVILLE, 2011). Na Figura 9, temos o exemplo de uma arquitetura de três camadas, em que cada uma executa uma tarefa específica de um sistema qualquer. Se for necessário futuramente, durante o desenvolvimento do *software*, uma camada inteira pode ser substituída por outra com função equivalente. Entretanto, a separação entre as camadas pode se tornar muito abstrata e até mesmo difícil de ser mantida. Por exemplo, podem ocorrer casos em que funções poderiam pertencer a mais de uma camada, ou nenhuma delas, conceitualmente, tornando difícil encaixa-la em uma só. Também há o caso no qual há mais de um módulo ou biblioteca de funções em uma mesma camada, tornando importante que estes módulos não se comuniquem, para não criar acidentalmente “subcamadas” desnecessariamente.

A ligação entre as camadas deve ser considerada de forma que não haja problemas no desempenho e implementação do modelo.

Figura 9 – Modelo simples de três camadas.

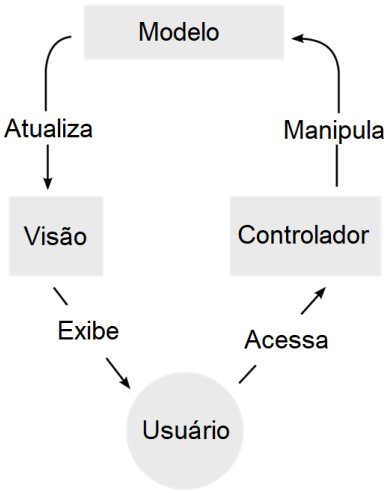



Fonte: Elaborado pelo autor

## 3.6 **Padrão modelo-visão-controlador**

O padrão modelo-visão-controlador (MVC) é descrito como uma separação entre a apresentação visual, ou *view,* e os processos lógicos, ou *model,* de um *software*. Esta separação é mediada por uma camada de integração da interface ou *controller*. Este padrão tem como vantagem guiar o trabalho de forma clara em estruturas separadas bem definidas (CORTEZ; VAZHENIN, 2015). Na Figura 10 temos uma representação do padrão MVC. Assim como o modelo de camadas, podemos considera-lo um modelo focado na organização abstrata do software. Isto permite que futuros contribuintes do código possam adaptar suas próprias funções lógicas isoladamente a uma camada. Os dados no software podem ser apresentados de maneiras diferentes, seja pelos resultados do *model,* ou a exibição no *view*, ajudando em sua interpretação. Este modelo pode ser desnecessariamente complexo para casos mais simples em que um ou mais elementos do MVC sejam abstratos demais para o problema.

Figura 10 – Modelo MVC.




Fonte: Adaptado de Regis (2010).

## 3.7 **Comparação de Padrões**

Podemos resumir as vantagens e desvantagens dos padrões citados pelo Quadro 1:

Quadro 1 - Vantagens e desvantagens dos principais padrões considerados.

| Padrão | Vantagens | Desvantagens |
|---|---|---|
| Pipe-and-filter | • Implementação simples;<br>• Fácil de compreender;<br>• Fácil de modificar. | • Restritivo por seu processolinear;<br>• Todo filtro deve ser transformativo. |
| Orientado aobjetos | • Flexibilidade de seus componentes em relação a suas funções;<br>• Fácil de criar novos objetos. | • Implementação complexa devido a necessidade de abstrair as tarefas do códigopara objetos<br>• Herança pode ocasionar problemas na depuração. |
| Baseado em eventos | • Troca de mensagens/dados implícita entre componentes; | • Implementação complexa, especialmente em relação ao manipulador de evento e |

| | | |
|---|---|---|
| | • Facilidade de implementar novos componentes. | seus gatilhos;<br>• Reações de eventos entre múltiplos componentes pode dificultar a depuração. |
| Repositório | • Ideal para manipulação de grandes quantidades de dados. | • Restritivo por seu processo centralizado ao repositório;<br>• Erros no componente do repositório se perpetuam emoutros componentes. |
| Multicamadas | • Fácil de modificar;<br>• Divide uma tarefa complexa em múltiplas tarefas simples. | • Desnecessário em casos que podem ser resolvidos com uma ou duas camadas;<br>• Ligações entre as camadas devem ser implementadas cuidadosamente. |
| MVC | • Pouca abstração para representar;<br>• Fácil de modificar componentes individuais;<br>• Múltiplas formas de representar dados. | • Desnecessário quando não há foco na apresentação dos dados |

Fonte: Elaborado pelo autor

## 3.8 **Outros padrões**

Existem mais modelos na literatura como *Forward/Receiver* e processo distribuído, porém ambos utilizam múltiplos softwares. Como desejamos criar um único *software* para a simulação da criptografia de sinais, estes fogem do nosso escopo. Outros foram considerados muito específicos em suas aplicações, como

padrões de arquiteturas de aplicação para linguagens ou dados bancários. Há também a possibilidade de combinar dois ou mais modelos, criando uma arquitetura híbrida. Também existem padrões de arquiteturas mais complexas como padrões de transição de estados e padrões de sub-rotinas que não foram considerados por fugirem do escopo do trabalho. Outra possibilidade é arquitetura híbrida (ou heterogênea) em que se misturam vários padrões diferentes conhecidos. Seu uso é bastante comum nos *softwares* modernos (SHAW; GARLAN, 1996; AKTAS, 2018; MIYAZAKI, 2018; YU, 2019), pois permitem aliviar as desvantagens de cada arquitetura e projetar o software de forma mais flexível.

## 3.9 Arquitetura adotada

Para os propósitos do KryptoSJPy, foi inicialmente considerado o uso da arquitetura de multicamadas. Usamos também conceitos do padrão MVC devido sua organização clara e pouca abstração, resultando assim em uma arquitetura híbrida.

Um modelo muito abstrato pode dificultar sua compreensão por futuros desenvolvedores do trabalho, assim como tornar mais difícil de segui-lo de forma consistente. No KryptoSJPy, duas camadas realizam os trabalhos de *model* e *view*, e elas serão mediados por um código principal, o *controller*. Detalharemos esta arquitetura na seção seguinte.

Este modelo híbrido foi escolhido devido às vantagens citadas dos padrões de multicamadas e MVC, especialmente em relação à independência entre as camadas. Também consideramos o padrão MVC como intuitivo para ditar a relação e separação entre as camadas. Outra vantagem desta arquitetura é a sua compatibilidade com o padrão de programação orientada a objetos.

# 4 DESCRIÇÃO DO SOFTWARE

Para desenvolver o KryptoSJPy, utilizamos a arquitetura híbrida descrita no capítulo anterior, junto ao conhecimento técnico que adquirimos na área para criar um programa de simulação da criptografia de sinais. Partes do *software* implementado neste trabalho foram originalmente baseadas em códigos anteriores do nosso grupo de pesquisa, escritos em Matlab. Isto acelerou o processo de implementação de equações e outros modelos matemáticos para o código. Esta conversão entre as linguagens foi possível devido à semelhança entre a sintaxe das linguagens Python e Matlab. Utilizamos as bibliotecas externas ao Python *numpy* (HARRIS et al., 2020), *scipy* (VIRTANEN et al., 2020), *matplotlib* (CASWELL et al., 2022) e *pandas* (THE PANDAS DEVELOPMENT TEAM, 2022), que são bibliotecas de funções matemáticas, equações científicas, manipulação de gráficos e manipulação de dados, respectivamente. Estes módulos são distribuídos pelo pacote Anaconda, que é uma distribuição de Python comumente utilizada para pesquisas científicas e acadêmicas. Além disto, utilizamos as bibliotecas *configparser* (para manipulação de arquivos de configuração) e *timeit* (para funções referentes a medições de tempo), que são nativas do Python.

A arquitetura do *software* é descrita por três camadas, separadas respectivamente em códigos e pastas distintas. As camadas são ilustradas na Figura 11 e descritas como:

1. A **camada de dados**, que inclui os dados selecionados pelo usuário, assim como parâmetros fixos inatos (como constantes matemáticas) ou variáveis que também podem ser alteradas pelo usuário. Os dados desta camada são exportados para um arquivo .ini externo ao código, que é configurado pelas funções da biblioteca *configparser*;
2. A **camada lógica**, que contém as funções relativas a cálculos matemáticos e manipulações dos dados da camada anterior. Estas funções são chamadas no código principal, sendo configuradas pelo usuário. Por exemplo, antes de o código ser executado, o usuário pode escolher adicionar, ou não, ruído ao sistema. Sua escolha será então considerada como um dado na

camada de dados, que ativará a função de geração de ruído no canal, ou não. Esta camada também descreve o objeto de “sinal” do código, que detalharemos em outro capítulo;

3. A **camada gráfica**, que engloba todas as funções pertinentes a geração de gráficos e outras figuras escolhidas pelo usuário, baseada nos dados emitidos pela camada lógica. Portanto, a camada gráfica prioritariamente não deverá retornar dados numéricos, apenas imagens.

Figura 11 – Arquitetura de software do projeto.

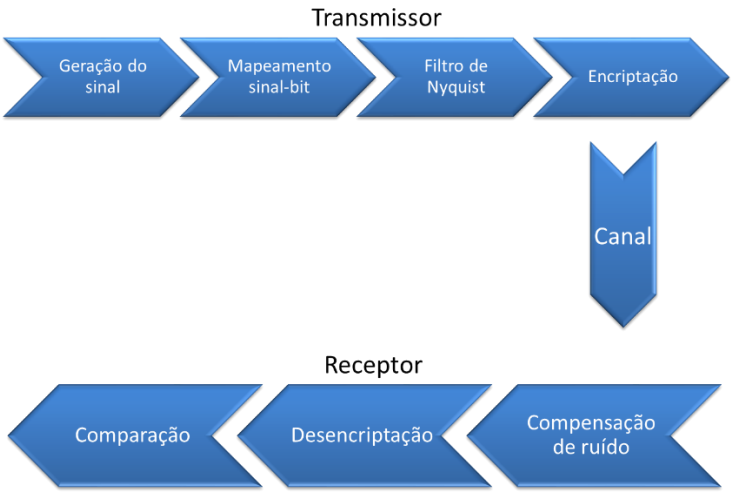

Fonte: Elaborado pelo autor

As camadas são mediadas por um código que chamamos de código principal ou código *main*. Este código principal utiliza funções das camadas anteriores, assim como as funções da camada lógica utilizam dados das camadas de dados para executar as simulações. O código *main* não precisa ser fixo, podendo ser alterado para servir as necessidades de simulação.

Um dos códigos *main* utilizado neste trabalho, por exemplo, está escrito de forma a simular um sistema de transmissão e recepção de sinais. Então, uma sequência aleatória de bits é gerada, convertida na forma de sinal, criptografada, transmitida por um canal, sendo o sinal finalmente recebido e convertido de volta para bits. Desta forma, o *software* tem o intuito de simular como o sinal criptografado se comporta na transmissão de dados em relação a um sinal não criptografado. Esta comparação pode ser observada nos resultados que o *software* apresenta no final da simulação. Este processo está descrito na Figura 12.

Outro código *main* pode descrever apenas parte deste processo para analisar a taxa de confusão e difusão dos sinais encriptados. Portanto, as funções do *software* podem ser utilizadas para analisar a criptografia de sinais de múltiplasformas.

Figura 12 – Esboço do modelo de transmissor e receptor do software




Fonte: Elaborado pelo autor

Neste *software*, o sinal é um objeto, mais especificamente, no Python é chamado de classe. O uso de classes para representar o sinal trouxe maior legibilidade ao código e conveniência para simular múltiplos sinais e simulações. O uso de objetos também possibilitou a identificação de sinais individualmente. Podemos atribuir aos objetos informações como a chave de sua encriptação, seu ruído específico, sua encriptação própria e quaisquer outras informações desejáveis. Esta individualização do sinal facilitou a depuração do código e as comparações de sinais diferentes em uma mesma simulação. Apesar de usarmos um objeto, não consideramos nossa arquitetura como orientada a objetos no sentido tradicional, pois não há comunicação entre objetos.

Como este *software* será usado por outros membros do nosso grupo de estudos, ativos ou futuros, houve uma grande ênfase em legibilidade e padronização do código. Isto não só influenciou na escolha da arquitetura, mas também na escolha do nome das variáveis, funções e objetos do código. A convenção comumente utilizada foi publicada em Propostas de Enriquecimento do Python (*Python Enhancement Proposals*, PEP) pela PSF. As PEPs são escritas com o objetivo de explicar recursos do Python implementados, em

desenvolvimento ou propostos, e recomendações ideais em suas utilizações. Dentre estas propostas, existem diretrizes de boas práticas de formatação do código-fonte. Porém, os autores recomendam que as convenções não sejam seguidas se sacrificarem a legibilidade do código, ou se vão contra convenções do contexto do código (ROSSUM; WARSAW; COGHLAN, 2001). Por exemplo, o uso das letras *i* e *q* maiúsculas são convencionais em DSP para nomear os eixos reais e complexos de quadratura de sinais digitais. Estas letras foram escritas maiúsculas em variáveis desses eixos, mesmo em casos que a convenção de uma PEP utilize letras minúsculas.

Dentre as diretrizes das PEPs consideramos as seguintes as mais relevantes para o nosso trabalho, levando em conta os recursos do Python que usamos:

1. Não colocar nomes ofensivos ou que divergem muito do contexto do código.
2. Usar variáveis nomeadas por uma letra única para casos específicos:
3. i, j, k, n, m, etc. para contadores ou iteradores;
4. f para chamadas de dados externos (*f = open("demofile.txt")*).
5. Não usar traços (-) em nomes de módulos.
6. Não usar múltiplos sublinhados antes ou depois do nome de uma variável (*_my_var_*).
7. Colocar o tipo da variável em seu nome pode ser redundante, visto que existe a função *type()*.
8. Sempre usar sublinhado se quiser separar palavras. Se não, simplesmente não as separe.
9. Todo tipo de dado deve ser nomeado em minúsculo, com excessão de classes e variáveis globais. Classes serão designadas em maiúsculo nas iniciais e variáveis globais em maiúsculo por completo.

No momento, o KryptoSJPy não tem interface própria, portanto é executado por um ambiente de desenvolvimento integrado (*Integrated development environment*, IDE). Um IDE é um software usado para programação em que o código pode ser editado, depurado e executado. Portanto, resultados numéricos e gráficos são visualizados no IDE e/ou arquivados no computador. O

IDE usado foio *Spyder*, que é distribuído dentro do pacote *Anaconda*.

No capítulo seguinte será detalhada a teoria físico-matemática usada paraas simulações básicas implementadas no software até o momento. Também serão descritas as criptografias de sinais que estão programadas no código na ultima versão.

# 5 TÉCNICAS DE CRIPTOGRAFIA IMPLEMENTADAS NO KRYPTOSJPY

Neste capítulo abordaremos detalhes teóricos sobre a simulação feita pelo KryptoSJPy. Na Seção 5.1 descreveremos a geração do sinal e o processo de transmissão e recepção do mesmo, independente de criptografias. Nas Seções 5.2 e 5.3 descreveremos as criptografias SPE e de embaralhamento intracanal, respectivamente, assim como sua simulação.

## 5.1 Descrição do sinal simulado

Até o momento a simulação mais realizada pelo simulador envolve a transmissão e recepção de sinais em banda base, sendo todas as operações realizadas considerando estes sinais somente no domínio elétrico. Na fase inicial de execução do programa, são considerados parâmetros customizáveis pelo usuário na camada de dados, como o número de bits ou símbolos a serem simulados, os números de amostras do sinal, a taxa de símbolos escolhida, e outros mais. A partir destes parâmetros criamos uma ou mais sequências de bits gerados pseudo-aleatoriamente (*Pseudo-random Bit Sequence,* PRBS). Também é possível usar uma sequência preestabelecida como dado de entrada do usuário. Estas sequências são mapeadas por uma função que irá gerar um sinal digital complexo $S[n]$ a partir dos bits por sinalização polar. O mapeamento dos bits é realizado tal que grupos de bits sejam convertidos em símbolos.

Nesta dissertação, consideramos as situações em que esses sinais digitais são o equivalente em banda-base de sinais BPSK, QPSK ou 16-QAM. Portanto, vamos nos referir a esses equivalentes em banda-base de sinais digitais, simplificadamente, como BPSK, QPSK ou 16-QAM. Chamaremos o número desímbolos do sinal de $N_s$.

A seguir, este sinal passa por um filtro de Nyquist, que neste simulador éum filtro de cosseno levantado. Esse filtro limita a banda do sinal, $B$, a:

$$B = \frac{R_S}{2}(1 + r) \qquad (1)$$

em que $R_s$ é a taxa de símbolos e $r$ é o fator de *roll-off* do filtro, que expressa a porcentagem do excesso de banda resultante do filtro em relação à

banda mínima prevista pelo Teorema de Amostragem de Nyquist (LATHI, 2012). Por fim, temos a encriptação do sinal. Discutiremos sobre as técnicas nas seções seguintes. Por enquanto, apenas assumiremos que o sinal foi encriptado por alguma técnica.

Em seguida realizamos a simulação de um canal, que pode ou não conter ruído. Neste trabalho, simularemos um canal ruidoso com ruído branco gaussiano aditivo (*Additive white Gaussian noise*, AWGN). A potência do ruído, $P_{noise}$, em um sinal complexo, é avaliada por meio de (LATHI, 2012):

$$P_{noise} = \frac{\sigma_{n_{re}}^2 + \sigma_{n_{im}}^2}{2} \qquad (2)$$

sendo que $\sigma_n^2$ é a variância do ruído. $\sigma_{n_{re}}^2$ e $\sigma_{n_{im}}^2$ são, respectivamente, as variâncias do ruído na componente real e imaginária do sinal. Neste projeto é suposto que as variâncias destas componentes são independentes entre si.

No receptor, o sinal passa por um filtro retangular passa-baixa de banda $B$ para reduzir o ruído que está fora da banda do sinal. Depois, a desencriptação em si é feita com a mesma chave da encriptação, processada para desencriptar o sinal. O sinal desencriptado é convertido novamente para forma de bits. Isso permite que o software compare os sinais encriptados e desencriptados, de modo a analisar as consequências da estratégia utilizada em relação à BER. Também é analisada a razão sinal-ruído (*signal-noise ratio,* SNR) do sinal, dada por:

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right), \qquad (3)$$

em que $P_{signal}$ é a potência do sinal, calculada por:

$$P_{signal} = \frac{Re\left(\sigma_{s[k]}^2\right) + Im(\sigma_{s[k]}^2)}{2}, \qquad (4)$$

tal que $\sigma_{s[k]}^2$ é a variância do sinal. A SNR é uma medida importante para avaliarmos o comportamento do sinal para diferentes quantidades de ruído no canal, e é calculada a partir da razão entre a potência do sinal transmitido com a potência do ruído. No nosso software não atribuímos nenhuma unidade em especial para a amplitude do sinal. Por esta razão consideramos a amplitude do

sinal e do ruído, assim como seus respectivos desvios padrões, como adimensionais. As potências do sinal e ruído, assim como a SNR, portanto, também serão adimensionais como consequência das equações (2), (3) e (4). Caso a unidade da amplitude fosse medida em volts ($V$), as potências e SNR seriam medidas em $V^2$.

Durante todos estes processos, que serão manipulados pela Camada Lógica, podemos gerar gráficos. Estes gráficos são desenhados utilizando a biblioteca matplotlib. As funções que geram gráficos recorrentes na simulação são servidas pela camada gráfica.

## 5.2 Criptografia de codificação espectral de fase por processamento digital de sinais

A SPE consiste no uso de modulação de fases aleatórias para modular fatias espectrais de um sinal complexo, pois o sinal considerado esta modulado opticamente. Porém, em sua versão digital (DSP-SPE), que é realizada no domínio elétrico exclusivamente, consideraremos que o numero de fatias é igual ao numero de $n$ amostras do sinal. Inicialmente consideramos um conjunto das fases $\phi_n$ $(n = 1, 2, \ldots, n)$ geradas pseudoaleatoriamente entre $-\pi$ e $\pi$. O conjunto de fases $\phi_n$ é considerado a chave $k_f$ do sistema. As $n$ amostras do sinal $S[n]$ são moduladas em fase conforme (SANTOS, 2020):

$$C[n] = S[n]e^{j\phi_n} \quad (5)$$

resultando no sinal encriptado $C[n]$. O sinal é então convertido ao domínio do tempo através de uma IFFT.

Para realizarmos a desencriptação, as amostras são multiplicadas pelo complexo conjugado da chave (SANTOS, 2020), novamente no domínio da frequência. Ou seja, o sinal é modulado tal que:

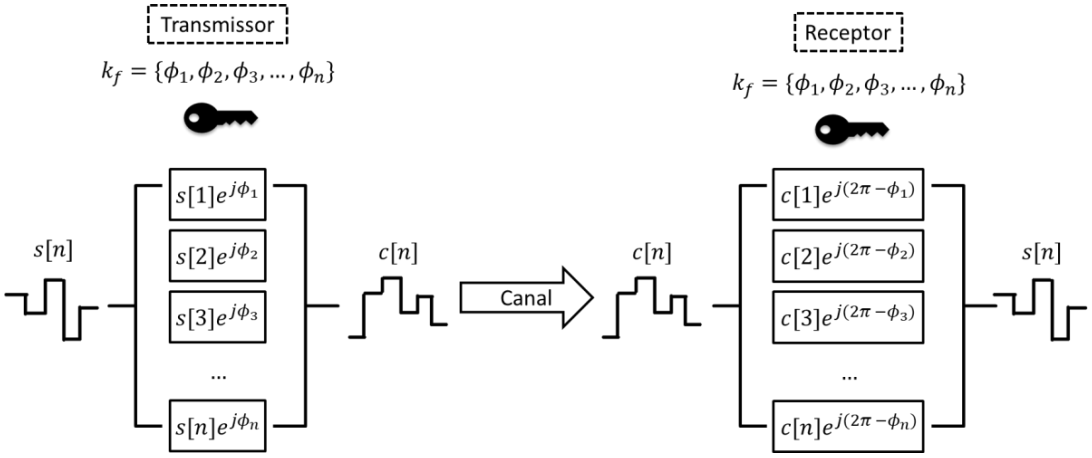
$$S[n] = C[n]e^{j\phi_n}. \quad (6)$$

Para finalizar o processo, o sinal é convertido para o domínio do tempo. A encriptação é considerada satisfatória se a BER do sinal encriptado for igual a 0,5. Na SPE, originalmente, as propriedades de confusão e difusão de Shannon não são satisfeitas. Isto ocorre pelo fato de que operações nas fatias de um sinal

de entrada não propagam para outras fatias após a encriptação (SANTOS, 2020). Neste trabalho, tentaremos verificar se o mesmo ocorre na DSP-SPE.

Todo esse processo é ilustrado na Figura 13. Nota-se que as FFTs e IFFTs citadas anteriormente estão omitidas na figura. Consideraremos que a FFT e IFFT sempre estão embutidas no modulo de encriptação. Esta omissão também ocorrerá para o embaralhamento intracanal por DSP (DSP-Scr) que será descrito na seção seguinte.

Figura 13 – Esboço do processo da DSP-SPE.




Fonte: Elaborado pelo autor

## 5.3 Criptografia de embaralhamento intracanal por processamento digitalde sinais

A criptografia de embaralhamento intracanal é baseada em misturar as posições das fatias espectrais de um sinal de forma aleatória. Em sua versão digital (DPS-Scr) vamos considerar, novamente, que o número de fatias é igual ao número de amostras do sinal $n$. Para embaralhar as amostras, é criada uma lista, nomeada $k_s$, tal que:

$$k_s = [1,2,3, \dots, n]. \qquad (7)$$

Esta lista é então embaralhada por um algoritmo. Neste caso usamos o algoritmo *random.shuffle* da biblioteca *numpy* em Python. O *random.shuffle* usa uma versão moderna do embaralhamento de Fisher-Yates, também conhecido como embaralhamento de Durstenfeld, para permutar os elementos de um arranjo ou lista (DURSTENFELD, 1964). Este algoritmo funciona a partir da

seguintesequência de passos:

É iniciado um loop *for* de $i - 1$ a $0$, sendo $i$ o número de elementos do arranjo;

1. Enumera-se $j$, o valor iterado atual no loop;
2. Escolhe-se um número $k$ tal que $0 \leq k \leq j$;
3. As posições de $k$ e $j$ são trocadas, se $k \neq j$. Caso o contrário, passe;
4. Repetem-se os passos II à IV até que $j = 0$.

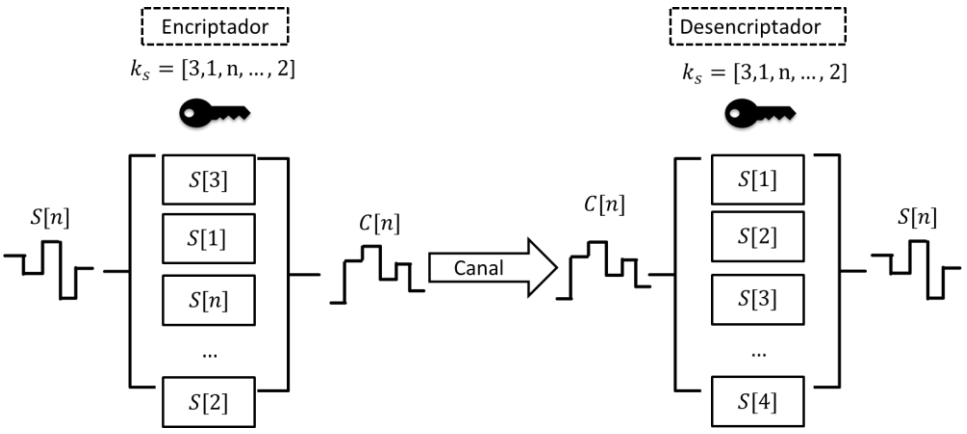
Este algoritmo é considerado bastante eficiente, dado sua simplicidade. E sua complexidade de tempo é linear, ou seja, $O(n)$.

Voltando ao DSP-Scr, após a lista $k_s$ ser embaralhada, podemos considerá- la como a chave desta criptografia. A encriptação então ocorre permutando as posições das amostras do sinal $S[n]$ utilizando a chave $k_s$ resultando no sinal encriptado $C[n]$ tal que:

$$C[n] = [s[k_s[1]], s[k_s[2], s[k_s[3]], \dots, s[k_s[n]]] \qquad (8)$$

A desencripação ocorre retornando as amostras para suas posições originais. Nota-se que a chave deve ser considerada uma lista indexada para que isto ocorra. Caso contrário, é necessário enviar uma chave composta por duas listas, sendo uma a lista $k_s$ embaralhada, e outra a lista $k_s$ não embaralhada. O Python utiliza listas indexadas no qual o índice inical é 0, contornando a necessidade do uso de enviar duas listas. Todo esse processo é ilustrado na figura 14.

Figura 14 – Esboço do processo da DSP-Scr




Fonte: Elaborado pelo autor

# 6 SIMULAÇÕES ELABORADAS E RESULTADOS

Primeiramente, é importante ressaltar de que todas as simulações elaboradas neste capítulo foram executadas no computador do autor deste trabalho. O computador utilizado não é considerado como especializado para simulações de grandes quantidades de dados, e também é de uso pessoal do autor, portanto, contendo programas e executáveis que podem consumir a memória durante a execução das simulações. O computador tem as seguintes especificações de hardware: processador Intel® Core™ i5-4570 CPU @ 3.20GHz, 8GB de memória RAM, 1TB de armazenamento interno, e uma placa de vídeo NVIDIA GeForce GTX 1050 Ti. Além disto, o código não está escrito de forma otimizada, visto que este tópico foge do escopo deste trabalho. Portanto, dados referentes ao tempo de execução das simulações serão exibidos por motivos de documentação de resultados, mas não devem ser usados como parâmetros de desempenho real do simulador.

Para demonstrar algumas das capacidades básicas da Camada Gráfica e a biblioteca *matplotlib*, elaboramos uma simulação de um sinal BPSK, que foi transmitido e recebido sem encriptação, sem filtro de Nyquist, e com ruído gaussiano (AWGN). O ruído do sinal será compensado pelo filtro retangular passa-baixa descrito na Seção 5.1. Os bits que compõem o sinal foram gerados como PRBS. A seguir listamos os parâmetros de entrada da simulação:

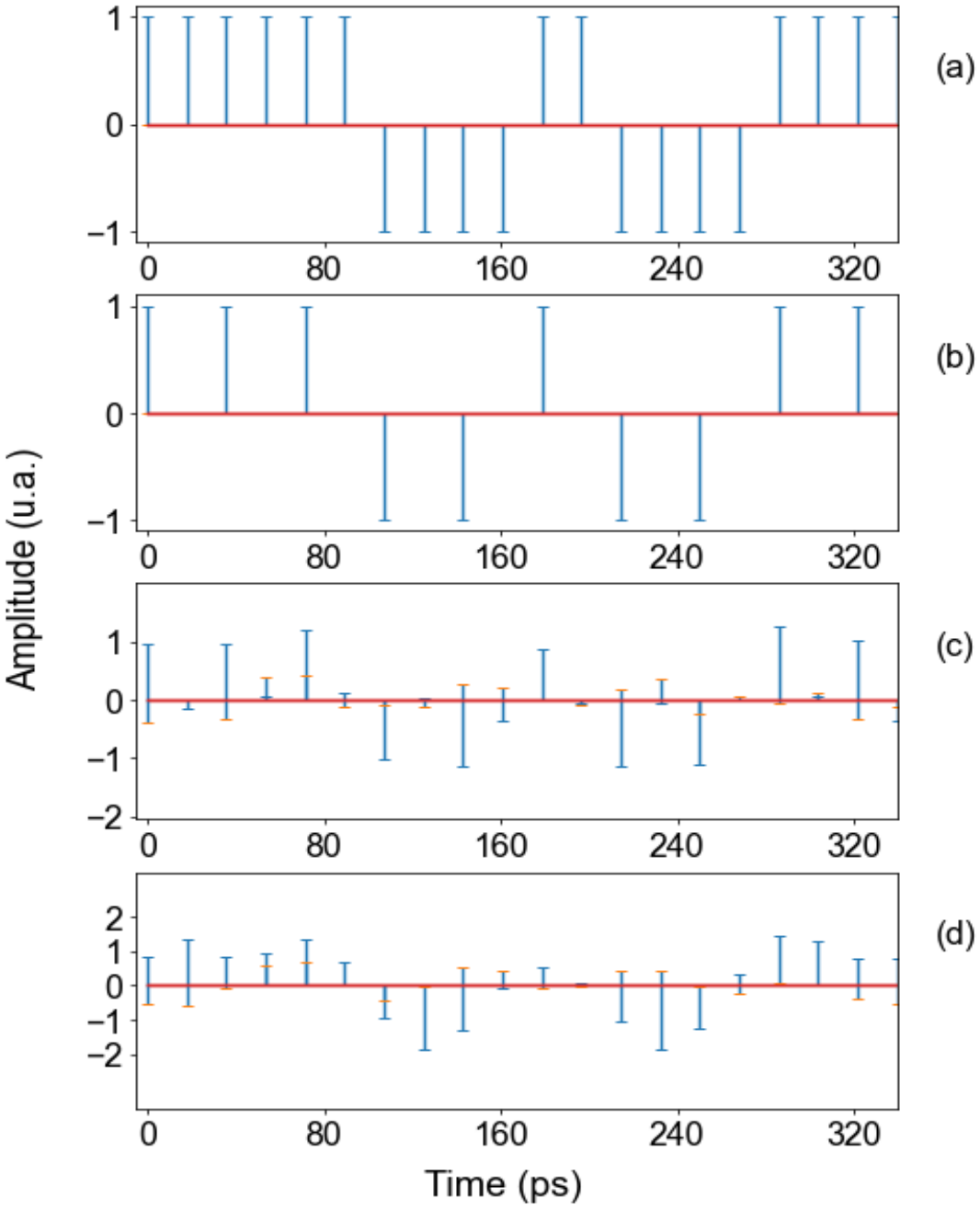
- Numero de símbolos $N_s = 16384$;
- Taxa de símbolos $R_s = 28GBaud$ ;
- Número de amostras por símbolo $n_{aps} = 2$;
- Desvio padrão do ruído gaussiano do canal $\sigma_n = 0{,}23$.

Os valores da $R_s$ e $\sigma_n$ foram escolhidos por motivos arbitrários. Já os valores de $N_s$ e $n_{aps}$ foram escolhidos para ser o suficiente para detalhar os gráficos, porém sem impactarem no desempenho das simulações. Também é importante deixar claro que comercialmente é comum utilizar $n_{aps} = 2$.

Nas Figuras 15 temos os gráficos do sinal no domínio do tempo do sinal simulado. Já na Figura 16 temos o espectro de amplitude, e o diagrama de constelação deste sinal. As figuras são divididas em gráficos subfiguras (a), (b), (c)e (d). Elas exibem o comportamento do sinal na entrada, após o *zero padding*,

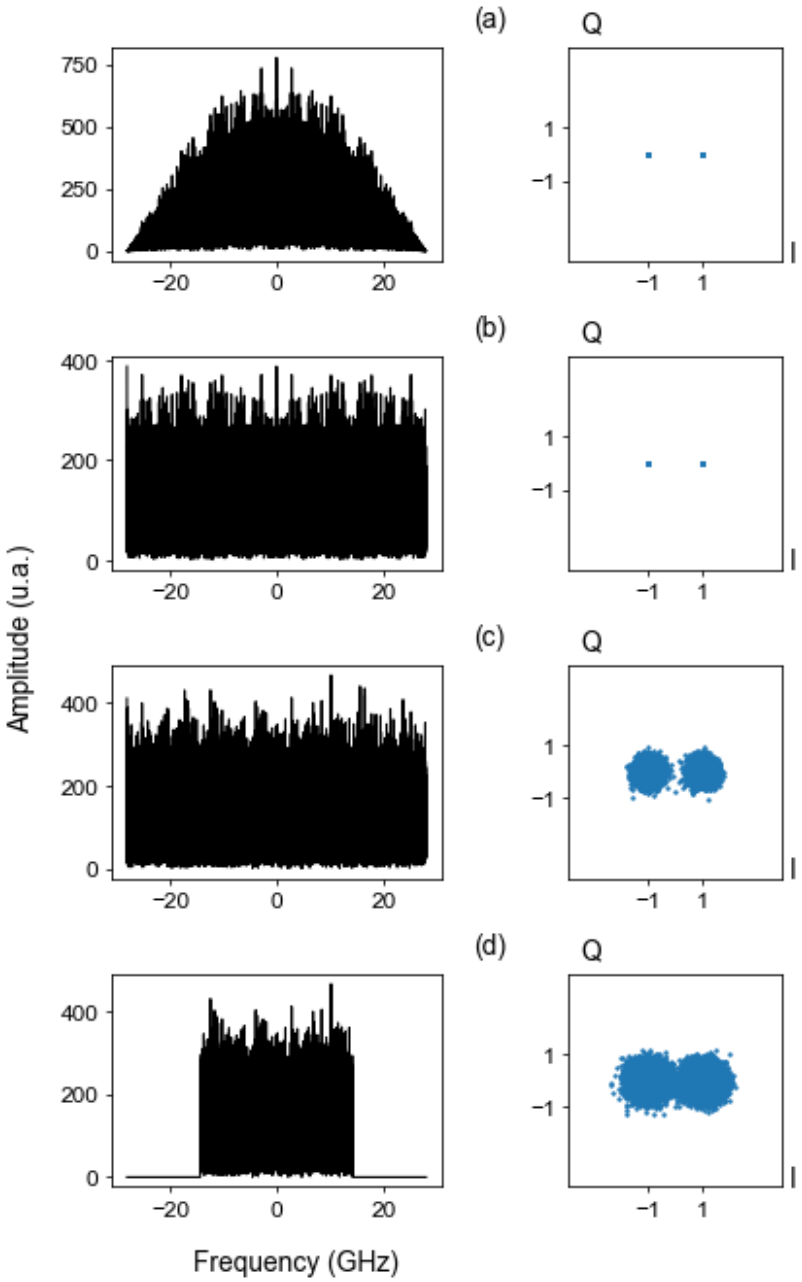
no canal ruidoso e após a compensação do ruído. A Figura 15 mostra apenas os 10 primeiros bits do sinal por questão de melhorar a visualização do gráfico. É possível notar que existe ISI. Podemos atribuir isto a falta do filtro de Nyquist.

Figura 15 – Gráficos no domínio do tempo dos 50 primeiros bits de um sinal BPSK. (a) Na entrada do transmissor; (b) Após o zero padding; (c) No canal ruídoso; (d) Após a compensação do ruído.




Fonte: Elaborado pelo autor

Figura 16 – Espectro de amplitude e diagrama de constelação de um sinal BPSK. (a) Na entrada do transmissor; (b) Após o zero padding; (c) No canal ruídoso; (d) Após a compensação do ruído.



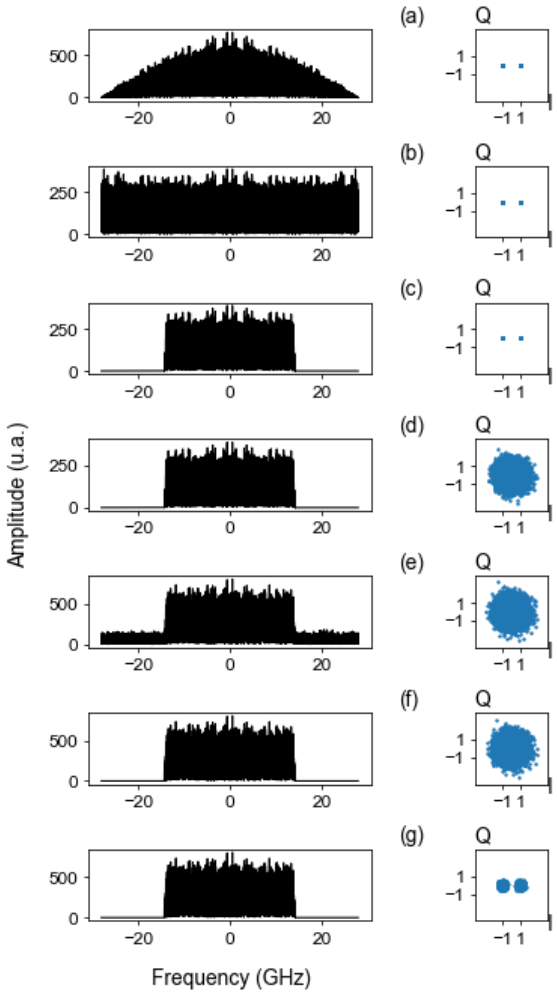

Fonte: Elaborado pelo autor

Para testar a DSP-SPE, temos uma simulação de um sinal BPSK, desta vezcom 16384 símbolos. Como descrito na Seção 5.1, o sinal passa por um filtro de Nyquist antes de ser encriptado. O fator de *roll-off* utilizado foi de $0{,}02$. Os valores da $N_s, R_s, n_{aps}$ e $\sigma_n$ foram iguais aos da simulação anterior.

Nas Figuras 17 apresentamos, os gráficos do espectro de amplitude e o diagrama de constelação durante todo o processo de transmissão e recepção. É possível notar na subfigura de espectro de amplitude da Figura 17(c) que a banda

do sinal não aumenta durante a encriptação. Nota-se também que mesmo com o ruído, o sinal desencriptado manteve sua constelação próxima dos bits originais. Para termos certeza estatística, simulamos novamente um total de 100 sinais com esta mesma configuração, resultando em $1638400$ símbolos simulados. A BER média resultante foi de aproximadamente $9{,}1 \cdot 10^{-10}$, com uma SNR média de $15{,}89\ dB$ de e um tempo de execução de 4 minutos e 48 segundos. Estes 1000 sinais foram simulados sem gerar gráficos, visto que a criação de imagens pelo *matplotlib*, especialmente de alta qualidade, impactam no tempo de execução. Nas subfigura de constelações das Figuras 17(d), 17(e) e 17(f) podemos notar a aleatoriedade da distribuição dos pontos, indicando sucesso na encriptação. Durante estes momentos da comunicação a BER é de 0,5.

Figura 17 – Espectro de amplitude de um sinal BPSK com DSP-SPE. (a) Na entrada do transmissor; (b) Após o zero padding; (c) Após o filtro de Nyquist; d) Após a encriptação; e) Após o canal AWGN; f) Após a compensação do ruído; g) Após a desencriptação.

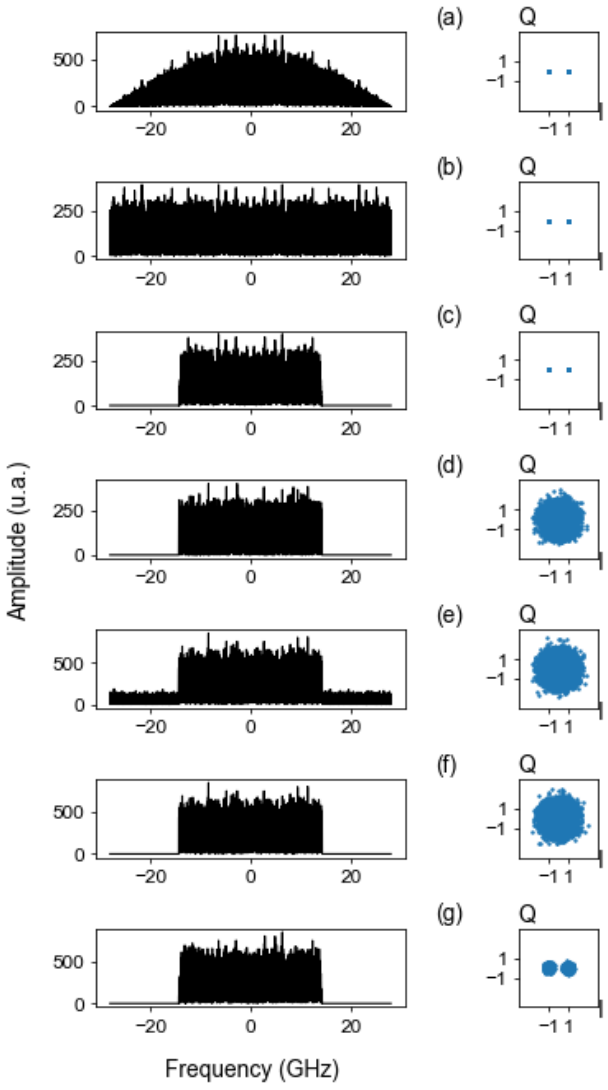



Fonte: Elaborado pelo autor

Também realizamos uma simulação com parâmetros idênticos aos da

simulação anterior para a criptografia de embaralhamento intracanal. Seus gráficos de espectro de amplitude e diagrama de constelação são exibidos nas Figuras 18. Os resultados também são semelhantes, com uma recuperação dos dados mesmo após o ruído. Ao gerar 100 sinais BPSK com DSP-Scr para confirmar os resultados da BER e da SNR, obtemos valores médios de aproximadamente $7{,}25 \cdot 10^{-8}$ e $15{,}81\ dB$ respectivamente. O tempo de execução foi de 4 minutos e 31 segundos. Assim como visto anteriormente, temos uma distribuição aleatória do sinal em torno de uma região circular nas Figuras 18(d), 18(e) e 18(f), indicando o processo de encriptação.

Figura 18 – Espectro de amplitude de um sinal BPSK com DSP-Scr. (a) Na entrada do transmissor; (b) Após o zero padding; (c) Após o filtro de Nyquist; d) Após a encriptação; e) Após o canal AWGN; f) Após a compensação do ruído; g) Após a desencriptação.

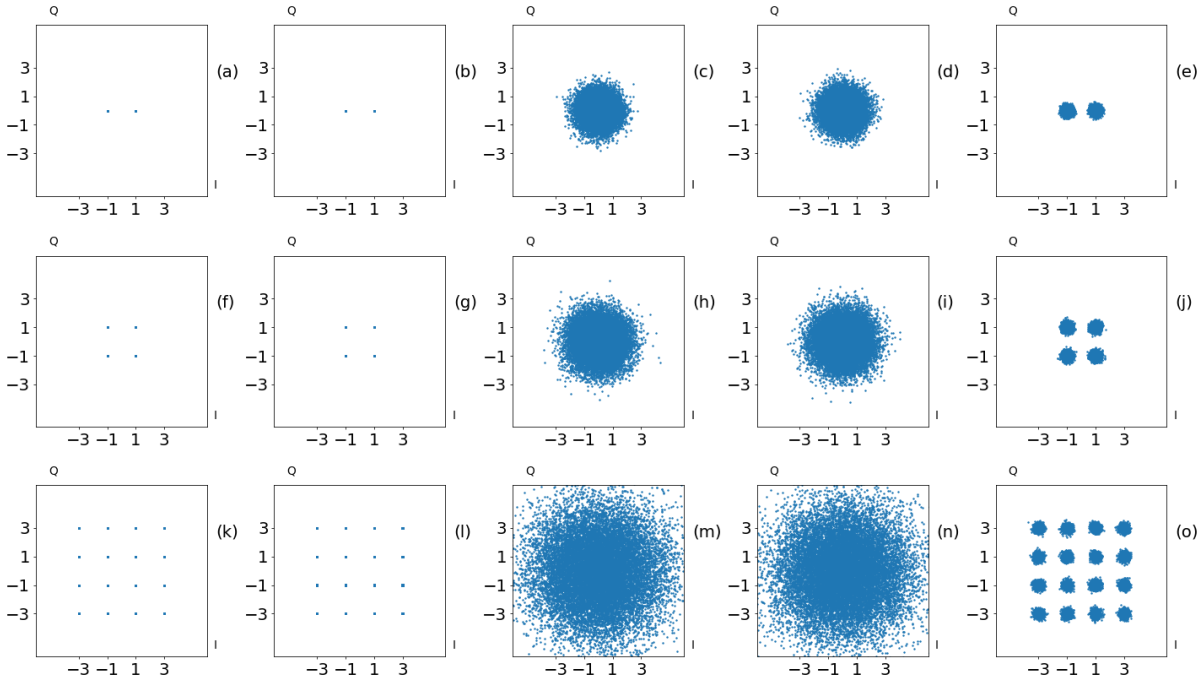



Fonte: Elaborado pelo autor

Por fim podemos testar um esquema de criptografia hibrido de DSP-SPE e

DSP-Scr, que chamaremos de DSP-SPE-Scr. Neste sistema, o nosso modulo de encriptação, após a filtragem, aplica DSP-SPE e em seguida DSP-Scr. A desencriptação é então feita pela DSP-Scr e DSP-SPE, em sucessão, para recuperar a mensagem. Para avaliar este esquema, simulamos um sinal BPSK, QPSK e 16-QAM com os parâmetros anteriores, resultando assim em sinais de $16384$ bits, $32768$ bits e $65536$ bits respectivamente. A figura 19 apresenta o diagrama da constelação de sinal original na entrada do transmissor, após o filtro de Nyquist, após a encriptação DSP-SPE-Scr, após o canal AWGN, e após as desencriptações. A BER calculada destes sinais esta na ordem de $10^{-8}$ou menos.

Figura 19 – Diagrama de constelação BPSK, QPSK e 16-QAM para DSP-SPE-Scr. (a), (f) e (k) Na entrada do transmissor; (b), (g) e (l) Após o filtro de Nyquist; (c), (h) e (m) Após a encriptação SPE-Scr; (d), (i) e (n) Após o canal AWGN; (e), (j) e (o) Após a desencr



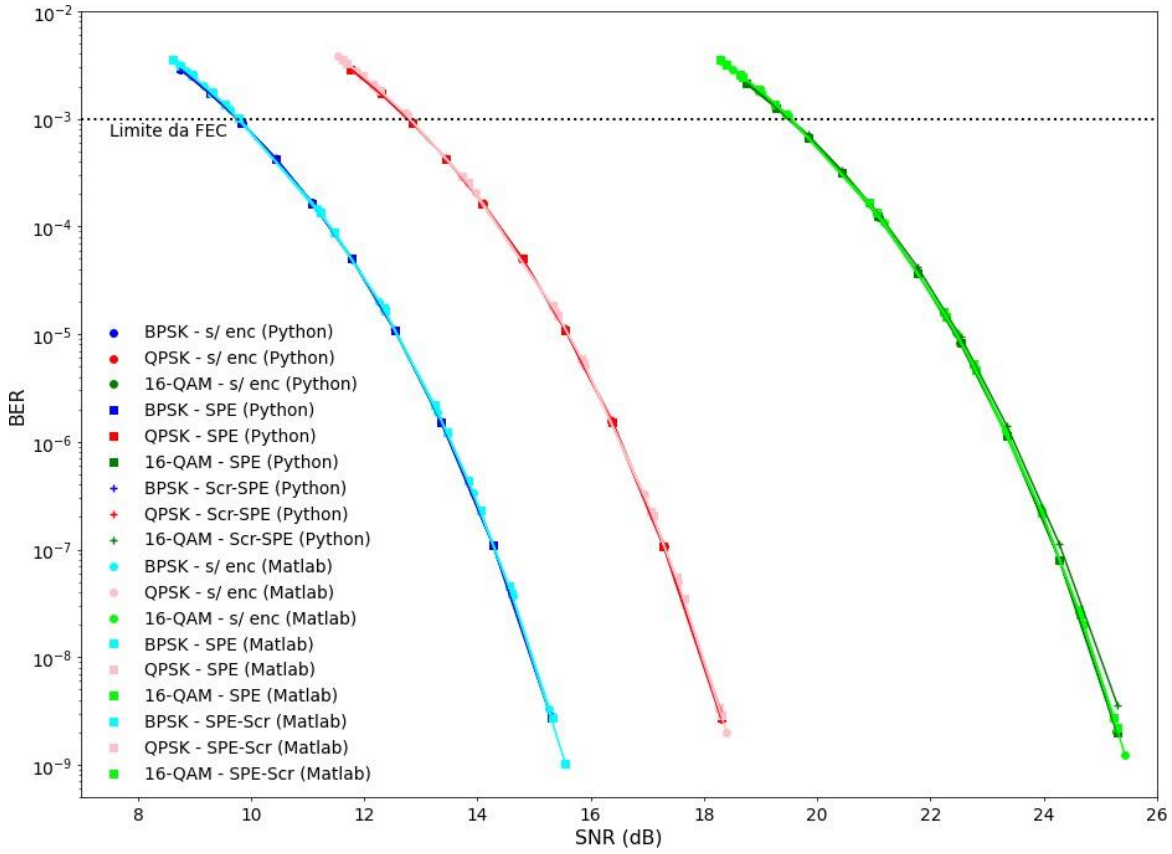

Fonte: Elaborado pelo autor

Para ter certeza dos valores da BER, comparamos os resultados obtidos pelo KryptoSJPy, com os dados obtidos em (SANTOS, 2020), no qual a autora utilizou o software *Matlab* para obter seus resultados. Utilizando um *loop*, aumentamos gradativamente $\sigma_n$ de $0{,}24$ a $0{,}51$, para gerar sinais BPSK, QPSK e 16-QAM sem encriptação, com encriptação DSP-SPE e DSP-SPE-Scr. Cada sinal foi gerado como um conjunto de $50$ sinais simultâneos para aumentar a precisão estatística da BER e da SNR, calculada pela média aritmética destes

conjuntos. Esta comparação de BER *vs.* SNR, assim como a comparação com os dados do Matlab, é apresentada na figura 20. Também utilizamos como parâmetro o limite da correção de erros antecipada (*Forward error correction, FEC*), que é utilizado para recuperar sinais a partir do limite de BER (TYCHOPOULOS et al, 2006). O limite que consideramos é de $10^{-3}$, pela recomendação G.975.1 (2004) do ITU-T.

Observando a figura 20 podemos inferir tanto a DSP-SPE quanto a DSP-SPE-Scr recuperam o sinal com sucesso de forma idêntica a casos em que não há criptografia. Além disto, as curvas do KryptoSJPy são idênticas as geradas pelo Matplotlib. Sobre os pontos individuais da curva, é possível que haja uma diferença por motivos estatísticos, além de possíveis diferenças no calculo da SNR entre os programas.

Figura 20 – Comparação de resultados do KryptoSJPy e Matlab da BER vs. SNR para sinais BPSK, QPSK e 16-QAM transmitidos por um canal AWGN.

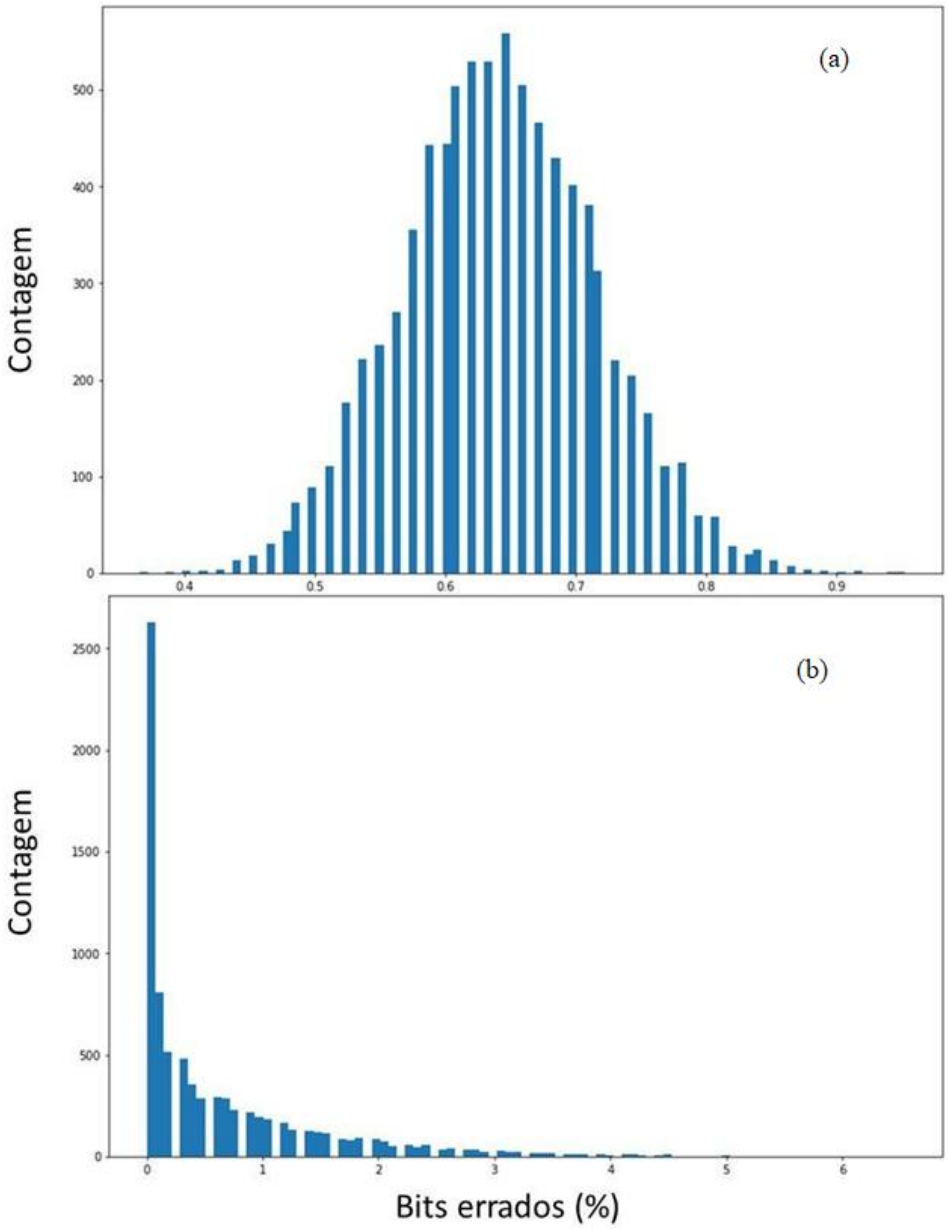



Fonte: Elaborado pelo autor

Por fim, podemos verificar a confusão e difusão da DSP-SPE. Realizamos as simulações de verificação destas propriedades utilizando *loops* para trocar bits da mensagem e da chave, como explicado na Seção 2.1. Entretanto como

mencionado na Seção 2.6, estas propriedades não são contempladas pela SPE. Verificamos estes resultados nas Figuras 21(a) e 21(b). Para que estas propriedades sejam possíveis, uma solução é o uso de chaves dinâmicas (SANTOS, 2020). As chaves dinâmicas, de modo simplificado, é o uso de chaves únicas para cada bloco de encriptação, ou no nosso caso para cada sinal único encriptado. A explicação completa esta em (Ngo *et al.*, 2010).

Figura 21 – Histograma de amplitude dos testes de confusão (a) e difusão (b) para SPE.




Fonte: Elaborado pelo autor

# 7 CONSIDERAÇÕES FINAIS

Os resultados obtidos pelas simulações foram satisfatórios, seguindo a teoria prevista pela literatura, assim como se validando a partir dos resultados de outros trabalhos semelhantes. Além disto, a arquitetura apresentada mostrou-se organizada, ajudando na criação deste software no seu período de aproximadamente 3 anos. Novas técnicas de criptografia podem ser facilmente implementadas apenas criando novas funções na camada lógica e ajustando o código principal para usa-las em simulações pertinentes.

O desenvolvimento do KryptoSJPy esta longe de seu fim. Durante a realização de simulações foi notada lentidão em certas funções, e até mesmo vazamentos de memória que impediram a geração de gráficos antes do tempo da defesa desta tese. Isso significa que existem partes do código que podem ser aperfeiçoadas. Uma solução a ser investigada é o uso de técnicas de multiprocessamento e *multithreading*, para utilizar o processador do computador em sua total potência. A própria arquitetura ainda pode ser melhorada também, talvez implementando outras classes além do sinal, a fim tornar o processo de simulação mais dinâmico.

Desenvolvimentos posteriores do software deverão incluir outros componentes do sistema de telecomunicação, como, por exemplo, os DACs, os moduladores ópticos, outros tipos de filtros, outros tipos de canais (alem dos AWGNs), as chaves dinâmicas, por exemplo. Para uma experiência melhor para usuários, também será necessário a criação de uma interface de usuário. Desta forma, um usuário poderá usar versões estáveis do software sem a necessidadede executar IDEs ou entender de programação em geral.

É esperado que versões mais aprimoradas do KryptoSJPy ajudem a elevar a criptografia de sinais a novos horizontes no campo da telecomunicação. O autor deste trabalho se propõe a continuar a melhorar o código mesmo após a defesa desta dissertação e ajudar aqueles que quiserem contribuir para o mesmo.

# REFERÊNCIAS

ABBADE, M. L. F.; BOBADILLA, L. D. B.; DE CARVALHO, D. O.; FERREIRA, A. A.; BONANI, L. H.; ALDAYA, I. **All-Optical Encryption Using Multi-Channel Spectral Shuffling.** IEEE Photonics Technology Letters. [S. l.]: Institute of Electrical and Electronics Engineers (IEEE), 1 jan. 2019. DOI 10.1109/lpt.2018.2884477. Disponível em: http://dx.doi.org/10.1109/LPT.2018.2884477.

ABBADE, M. L. F.; CVIJETIC, M.; MESSANI, C. A.; ALVES, C. J.; TENENBAUM, S. **All-optical cryptography of M-QAM formats by using two- dimensional spectrally sliced keys. Applied Optics.** [S. l.]: The Optical Society, 5 maio 2015. DOI 10.1364/ao.54.004359. Disponível em: http://dx.doi.org/10.1364/AO.54.004359.

ABBADE, M. L. F.; DIOGO BUENO BOBADILLA, L.; BEATRIZ MARTAO, V.; ORQUIZA DE CARVALHO, D.; FERREIRA, A. A.; BONANI, L. H. **Double-lock strategy applied to optical spectral phase and delay encoding.** 2017 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC). [S. l.]: IEEE, ago. 2017. DOI 10.1109/imoc.2017.8121129. Disponível em: http://dx.doi.org/10.1109/IMOC.2017.8121129.

ABBADE, M. L. F.; NOGUEIRA, M. P.; SANTOS, M. O.; FAGOTTO, E. A. M.; BONANI, L. H.; ALDAYA, I. **DSP-Based Multi-Channel Spectral Shuffling Applied to Optical Networks.** IEEE Photonics Technology Letters. [S. l.]: Institute of Electrical and Electronics Engineers (IEEE), 1 fev. 2020. DOI 10.1109/lpt.2019.2962837. Disponível em: http://dx.doi.org/10.1109/LPT.2019.2962837.

AKTAS, M. S. **Hybrid cloud computing monitoring software architecture.** Concurrency and Computation: Practice and Experience, v. 30, n.21, p. e4694, 25 maio 2018.

ALEXANDER, C. H. O'D. **Cryptographic History of Work on the German Naval**

**Enigma.** Catalogue reference HW 25, nº 1, ca. 1945. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1949.tb00928.xi. Acesso em: 27 jun. 2020.

ALVAREZ, G.; LI, S. **Some basic cryptographic requirements for chaos-based cryptosystems.** International Journal of Bifurcation and Chaos. [S. l.]: World Scientific Pub Co Pte Lt, ago. 2006. DOI 10.1142/s0218127406015970. Disponível em: http://dx.doi.org/10.1142/S0218127406015970.

BENNETT, C. H.; BRASSARD, G. **Quantum cryptography: public key distribution and coin tossing.** IEEE International Conference on Computers, Systems and Signal Processing, v. 175, p. 8, dec. 1984.

BRIEGEL, H.-J.; DÜR, W.; CIRAC, J. I.; ZOLLER, P. **Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication.** Physical Review Letters. [S. l.]: American Physical Society (APS), 28 dez. 1998. DOI 10.1103/physrevlett.81.5932. Disponível em: http://dx.doi.org/10.1103/PhysRevLett.81.5932.

CASWELL, Thomas A; LEE, Antony; DROETTBOOM, Michael; DE ANDRADE, Elliott Sales; HOFFMANN, Tim; KLYMAK, Jody; HUNTER, John; FIRING, Eric; STANSBY, David; VAROQUAUX, Nelle; NIELSEN, Jens Hedegaard; ROOT, Benjamin; MAY, Ryan; ELSON, Phil; SEPPÄNEN, Jouni K.; DALE, Darren; JAE-JOON LEE; GUSTAFSSON, Oscar; MCDOUGALL, Damon; … KNIAZEV, Nikita. **matplotlib/matplotlib**: REL: v3.6.2. [S. l.]: Zenodo, 2022. DOI 10.5281/ZENODO.7275322. Disponível em: https://zenodo.org/record/7275322.

CORNEJO, J.; TOCNAYE, E. L. M. B. **Non-invasive WDM channel scrambling for secure high data rate optical transmissions.** Proc. SPIE: Photon Management III, Strasbourg, France, v. 6994, Abril 2008. DOI 10.1117/12.781036.

CORTEZ, R.; VAZHENIN, A. **Virtual model-view-controller designpattern: Extended MVC for service-oriented architecture.** IEEJ Transactions on

Electrical and Electronic Engineering, vol. 10, no. 4, p. 411–422, 27 Mar. 2015.DOI 10.1002/tee.22101. Disponível em: http://dx.doi.org/10.1002/tee.22101. Acesso em: 19 ago. 2021.

DAHAN, D.; MAHLAB, U. **Security threats and protection procedures for optical networks.** IET Optoelectronics. [S. l.]: Institution of Engineering and Technology (IET), set. 2017. DOI 10.1049/iet-opt.2016.0150. Disponível em: http://dx.doi.org/10.1049/iet-opt.2016.0150.

DURSTENFELD, R. **Algorithm 235: Random permutation.** Communications of the ACM. [S. l.]: Association for Computing Machinery (ACM), jul. 1964. DOI 10.1145/364520.364540. Disponível em: http://dx.doi.org/10.1145/364520.364540.

FURDEK, M.; NATALINO, C.; DI GIGLIO, A.; SCHIANO, M. Optical network security management: requirements, architecture, and efficient machine learning models for detection of evolving threats [Invited]. Journal of Optical Communications and Networking. [S. l.]: Optica Publishing Group, 24 dez. 2020. DOI 10.1364/jocn.402884. Disponível em: http://dx.doi.org/10.1364/JOCN.402884.

GUTMANN, P. **Cryptographic Security Architecture**. 1st. ed. New York, USA: Springer, 2004. 320 p. ISBN 978-0-387-21551-8. Disponível em: https://link.springer.com/book/10.1007/b97264#about. Acesso em: 19 ago. 2021.

HARRIS, Charles R.; MILLMAN, K. Jarrod; VAN DER WALT, Stéfan J.; GOMMERS, Ralf; VIRTANEN, Pauli; COURNAPEAU, David; WIESER, Eric; TAYLOR, Julian; BERG, Sebastian; SMITH, Nathaniel J.; KERN, Robert; PICUS, Matti; HOYER, Stephan; VAN KERKWIJK, Marten H.; BRETT, Matthew; HALDANE, Allan; DEL RÍO, Jaime Fernández; WIEBE, Mark; PETERSON, Pearu; … OLIPHANT, Travis E. **Array programming with NumPy.** Nature. [S. l.]: Springer Science and Business Media LLC, 16 set. 2020. DOI 10.1038/s41586-020-2649-2. Disponível em: http://dx.doi.org/10.1038/s41586-020-2649-2.

HUY-HOANG NGO; XIAN-PING WU; PHU-DUNG LE; CAMPBELL WILSON; BALASUBRAMANIAM SRINIVASAN. **Dynamic Key Cryptography and Applications**. International Journal of Network Security, vol. 10, nº 3, 1 maio 2010.https://doi.org/10.6633/IJNS.201005.10(3).01.

JIANG, N.; ZHAO, X.; ZHAO, A.; WANG, H.; QIU, K.; TANG, J. **High-Rate Secure Key Distribution Based on Private Chaos Synchronization and Alternating Step Algorithms.** International Journal of Bifurcation and Chaos. [S. l.]: World Scientific Pub Co Pte Lt, fev. 2020. DOI 10.1142/s0218127420500273. Disponível em: http://dx.doi.org/10.1142/S0218127420500273.

KATZ, Jonathan; LINDELL, Yehuda. **Introduction to Modern Cryptography**. Third. ed. New York: CRC Press, 2020. 628 p. ISBN 978- 0815354369.

KE, J.; YI, L.; HOU, T.; HU, W. **Key technologies in chaotic optical communications.** Frontiers of Optoelectronics. [S. l.]: Springer Science and Business Media LLC, set. 2016. DOI 10.1007/s12200-016-0570-y. Disponível em: http://dx.doi.org/10.1007/s12200-016-0570-y.

LATHI, B. P.; DING, Z. **Sistemas de Comunicações Analógicos e Digitais Modernos**. 4ª. ed. Rio de Janeiro: LTC, 2012. 837 p. ISBN 978-85-216- 2027-3.

LIAO, S.-K.; CAI, W.-Q.; LIU, W.-Y.; ZHANG, L.; LI, Y.; REN, J.-G.; YIN, J.; SHEN, Q.; CAO, Y.; LI, Z.-P.; LI, F.-Z.; CHEN, X.-W.; SUN, L.-H.; JIA, J.-J.; WU, J.-C.; JIANG, X.-J.; WANG, J.-F.; HUANG, Y.-M.; WANG, Q.; ... PAN, J.-W. **Satellite-to-ground quantum key distribution.** Nature. [S. l.]: Springer Science and Business Media LLC, 9 ago. 2017. DOI 10.1038/nature23655. Disponível em: http://dx.doi.org/10.1038/nature23655.

LUCAMARINI, M.; YUAN, Z. L.; DYNES, J. F.; SHIELDS, A. J. **Overcoming the rate–distance limit of quantum key distribution without quantum repeaters.** Nature. [S. l.]: Springer Science and Business Media LLC, maio 2018. DOI 10.1038/s41586-018-0066-6. Disponível em: http://dx.doi.org/10.1038/s41586-018-0066-6.

MAO, Y.; WANG, B.-X.; ZHAO, C.; WANG, G.; WANG, R.; WANG, H.; ZHOU, F.; NIE, J.; CHEN, Q.; ZHAO, Y.; ZHANG, Q.; ZHANG, J.; CHEN, T.-Y.; PAN, J.-W. **Integrating quantum key distribution with classical communications in backbone fiber network**. Optics Express. [S. l.]: The Optical Society, 27 fev. 2018. DOI 10.1364/oe.26.006010. Disponível em: http://dx.doi.org/10.1364/OE.26.006010.

MIYAZAKI, T. et al. **An OpenCL-based Software Framework for a Heterogeneous Multicore Architecture on Zynq-7000 SoC.** IPSJ Transactions on System LSI Design Methodology, v. 12, n. 0, p. 46–49, 2019.

PECORA, L. M.; CARROLL, T. L. **Synchronization in chaotic systems.** Physical Review Letters. [S. l.]: American Physical Society (APS), 19 fev. 1990. DOI 10.1103/physrevlett.64.821. Disponível em: http://dx.doi.org/10.1103/PhysRevLett.64.821.

ROSENBLUM, M. G.; PIKOVSKY, A. S.; KURTHS, J. **Phase Synchronization of Chaotic Oscillators.** Physical Review Letters. [S. l.]: American Physical Society (APS), 11 mar. 1996. DOI 10.1103/physrevlett.76.1804.Disponível em: http://dx.doi.org/10.1103/PhysRevLett.76.1804.

ROSSUM, G. van; WARSAW , B.; COGHLAN , N. **PEP 8: Style Guide for Python Code.** In: PYTHON-DEV. Python Enhancement Proposals. [S. l.], 5 jul. 2001. Disponível em: https://peps.python.org/pep-0008/. Acesso em: 20 set. 2022.

RULKOV, N. F.; SUSHCHIK, M. M.; TSIMRING, L. S.; ABARBANEL, H. D. I. **Generalized synchronization of chaos in directionally coupled chaotic systems.** Physical Review E. [S. l.]: American Physical Society (APS), 1 fev. 1995. DOI 10.1103/physreve.51.980. Disponível em: http://dx.doi.org/10.1103/PhysRevE.51.980.

SANGOUARD, N.; SIMON, C.; DE RIEDMATTEN, H.; GISIN, N. **Quantum repeaters based on atomic ensembles and linear optics.** arXiv, 2009. DOI 10.48550/ARXIV.0906.2699. Disponível em: https://arxiv.org/abs/0906.2699.

SANTOS, M. de O. **Criptografia na camada física baseada em codificação espectral implantada por meio de DSP e aplicada a redes ópticas.** Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), São João da Boa Vista, 2020.

SCHUMACHER, B. **Quantum coding.** Physical Review A. [S. l.]: American Physical Society (APS), 1 abr. 1995. DOI 10.1103/physreva.51.2738. Disponível em: http://dx.doi.org/10.1103/PhysRevA.51.2738.

SHANNON, C. E. **Communication Theory of Secrecy Systems***. Bell System Technical Journal. [S. l.]: Institute of Electrical and Electronics Engineers (IEEE), out. 1949. DOI 10.1002/j.1538-7305.1949.tb00928.x. Disponível em: http://dx.doi.org/10.1002/j.1538-7305.1949.tb00928.x.

SHAW, M.; GARLAN, D. **Software Architecture: Perspectives on anEmerging Discipline.** Englewood Cliffs, New Jersey: Prentice Hall, 1996.

SHUAI, R.; YINFA, Z.; GUIJIN, X.; XUEJUN, R.; XIAOSHUANG, W. **Research on Security Protection of Key Information Infrastructure in Optical Networks.** 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN). [S. l.]: IEEE, jun. 2019. DOI 10.1109/iccsn.2019.8905305. Disponível em: http://dx.doi.org/10.1109/ICCSN.2019.8905305.

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SULTAN, A.; YANG, X.; HAJOMER, A. A. E.; HU, W.**Chaotic Constellation Mapping for Physical-Layer Data Encryption in OFDM-PON.**IEEE Photonics Technology Letters. [S. l.]: Institute of Electrical and

Electronics Engineers (IEEE), 15 fev. 2018. DOI 10.1109/lpt.2018.2789468. Disponível em: http://dx.doi.org/10.1109/LPT.2018.2789468.

TEH, J. S.; ALAWIDA, M.; SII, Y. C. **Implementation and practical problems of chaos-based cryptography revisited.** Journal of Information Security and Applications. [S. l.]: Elsevier BV, fev. 2020. DOI 10.1016/j.jisa.2019.102421. Disponível em:http://dx.doi.org/10.1016/j.jisa.2019.102421.

THE PANDAS DEVELOPMENT TEAM. **pandas-dev/pandas**: Pandas. [S. l.]: Zenodo, 2022. DOI 10.5281/ZENODO.7223478. Disponível em: https://zenodo.org/record/7223478.

TYCHOPOULOS, A.; KOUFOPAULOU, O.; TOMKOS, I. FEC in optical communications - A tutorial overview on the evolution of architectures and the future prospects of outband and inband FEC for optical communications.IEEE Circuits and Devices Magazine. [S. l.]: Institute of Electrical and Electronics Engineers (IEEE), nov. 2006. DOI 10.1109/mcd.2006.307281. Disponível em: http://dx.doi.org/10.1109/MCD.2006.307281.

VIRTANEN, Pauli; GOMMERS, Ralf; OLIPHANT, Travis E.; HABERLAND, Matt; REDDY, Tyler; COURNAPEAU, David; BUROVSKI, Evgeni; PETERSON, Pearu; WECKESSER, Warren; BRIGHT, Jonathan; VAN DER WALT, Stéfan J.; BRETT, Matthew; WILSON, Joshua; MILLMAN, K. Jarrod; MAYOROV, Nikolay; NELSON, Andrew R. J.; JONES, Eric; KERN, Robert; … LARSON, Eric. **SciPy 1.0**: fundamental algorithms for scientific computing in Python. Nature Methods. [S. l.]: Springer Science and Business Media LLC, 3 fev. 2020. DOI 10.1038/s41592-019-0686-2. Disponível em: http://dx.doi.org/10.1038/s41592-019-0686-2.

WOOTTERS, W. K.; ZUREK, W. H. **A single quantum cannot be cloned.** Nature. [S. l.]: Springer Science and Business Media LLC, out. 1982. DOI 10.1038/299802a0. Disponível em: http://dx.doi.org/10.1038/299802a0.

XU, F.; MA, X.; ZHANG, Q.; LO, H.-K.; PAN, J.-W. **Secure quantum key distribution with realistic devices.** Reviews of Modern Physics. [S. l.]: American

Physical Society (APS), 26 maio 2020. DOI 10.1103/revmodphys.92.025002. Disponível em: http://dx.doi.org/10.1103/RevModPhys.92.025002.

YANG, X.; SHEN, Z.; HU, X.; HU, W. **Chaotic Encryption Algorithm Against Chosen-Plaintext Attacks in Optical OFDM Transmission.** IEEE Photonics Technology Letters. [S. l.]: Institute of Electrical and Electronics Engineers (IEEE), 15 nov. 2016. DOI 10.1109/lpt.2016.2601659. Disponível em: http://dx.doi.org/10.1109/LPT.2016.2601659.

YI, L.; KE, J.; XIA, G.; HU, W. **Phase chaos generation and security enhancement by introducing fine-controllable dispersion.** Journal of Optics. [S. l.]: IOP Publishing, 1 jan. 2018. DOI 10.1088/2040-8986/aa9e1d. Disponívelem: http://dx.doi.org/10.1088/2040-8986/aa9e1d.

YIN, H.-L.; CHEN, T.-Y.; YU, Z.-W.; LIU, H.; YOU, L.-X.; ZHOU, Y.-H.; CHEN, S.-J.; MAO, Y.; HUANG, M.-Q.; ZHANG, W.-J.; CHEN, H.; LI, M. J.; NOLAN, D.; ZHOU, F.; JIANG, X.; WANG, Z.; ZHANG, Q.; WANG, X.-B.; PAN, J.-W. **Measurement-Device-Independent Quantum Key Distribution Over a 404 km Optical Fiber.** Physical Review Letters. [S. l.]: American Physical Society (APS), 2 Nov. 2016. DOI 10.1103/physrevlett.117.190501. Disponivel em: http://dx.doi.org/10.1103/PhysRevLett.117.190501.

YU, C. et al. Onboard system of hybrid underwater robotic vehicles: Integrated software architecture and control algorithm. Ocean Engineering, v. 187, p. 106121, set. 2019.

ZAFAR IQBAL, M.; FATHALLAH, H.; BELHADJ, N. **Optical fiber tapping: Methods and precautions.** 8th International Conference on High-capacity Optical Networks and Emerging Technologies. [S. l.]: IEEE, dez. 2011. DOI 10.1109/honet.2011.6149809. Disponível em: http://dx.doi.org/10.1109/HONET.2011.6149809.

ZHANG, W.; VAN LEENT, T.; REDEKER, K.; GARTHOFF, R.; SCHWONNEK, R.; FERTIG, F.; EPPELT, S.; ROSENFELD, W.; SCARANI, V.; LIM, C. C.-W.;

WEINFURTER, H. **A device-independent quantum key distribution system for distant users.** Nature. [S. l.]: Springer Science and Business Media LLC, 27 jul. 2022. DOI 10.1038/s41586-022-04891-y. Disponível em: http://dx.doi.org/10.1038/s41586-022-04891-y.