MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

BRUNO DOS SANTOS ROCHA

IDENTIFICAÇÃO DE CRIPTOSSISTEMAS PÓS-QUÂNTICOS POR MEIO DE APRENDIZADO DE MÁQUINA.

RIO DE JANEIRO
2023

BRUNO DOS SANTOS ROCHA

IDENTIFICAÇÃO DE CRIPTOSSISTEMAS PÓS-QUÂNTICOS POR MEIO DE APRENDIZADO DE MÁQUINA.

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador(es): José Antonio Moreira Xexéo, D.Sc
Renato Hidaka Torres, D.Sc

Rio de Janeiro
2023

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Rocha, Bruno dos Santos.
Identificação de criptossistemas pós-quânticos por meio de aprendizado de máquina. / Bruno dos Santos Rocha. – Rio de Janeiro, 2023.
157 f.

Orientador(es): José Antonio Moreira Xexéo e Renato Hidaka Torres.

Dissertação (mestrado) – Instituto Militar de Engenharia, Sistemas e Computação, 2023.

1. Criptografia pós-quântica; Identificação de padrões; Ataque de distinção; Aprendizado de máquina; *NIST statistical tests*; *Electronic Codebook*; *Cipher Block Chaining*; Mecanismo de encapsulamento de chaves.i. Xexéo, José Antonio Moreira (orient.) ii. Torres, Renato Hidaka (orient.) iii. Título

BRUNO DOS SANTOS ROCHA

Identificação de criptossistemas pós-quânticos por meio de aprendizado de máquina.

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador(es): José Antonio Moreira Xexéo e Renato Hidaka Torres.

Aprovado em Rio de Janeiro, 11 de dezembro de 2023, pela seguinte banca examinadora:

Prof. TC Rfm **José Antonio Moreira Xexéo** - D.Sc. do IME - Presidente

Documento assinado digitalmente
gov.br RENATO HIDAKA TORRES
Data: 23/01/2024 11:03:01-0300
Verifique em https://validar.iti.gov.br

Prof. **Renato Hidaka Torres** - D.Sc. da UFPA

Prof. **Flavio Luis de Mello** - D.Sc. da UFRJ

Prof. Cel **Julio Cesar Duarte** - D.Sc. do IME

Prof. Cel **Anderson Fernandes Pereira dos Santos** - D.Sc. do IME

Rio de Janeiro
2023

Dedico este trabalho ao meu Deus, à minha família e a todos que acreditaram em mim.

AGRADECIMENTOS

Primeiramente, agradeço ao Senhor Jesus Cristo por cuidar de mim, ser meu amparo nos momentos difíceis e me dar esperança de dias melhores. Obrigado, meu Deus, por tudo!

Agradeço à minha esposa, Roberta Lopes, pelo seu companheirismo e apoio durante esta jornada. Também sou grato ao meu filho, Davi, que, com seus sorrisos inocentes, me incentivou a prosseguir nesta empreitada.

Agradeço também aos meus pais, Margareth e José Carlos, que dedicaram suas vidas aos filhos e sempre nos ofereceram o melhor que podiam.

Por fim, agradeço aos professores José Antônio Moreira Xexéo e Renato Hidaka Torres pela orientação, paciência e dedicação. Também deixo registrado meu apreço aos queridos colegas de curso, Rafael Oliveira e Davi Rocha, pelo apoio na condução desta pesquisa, e a todos os professores do Instituto Militar de Engenharia, onde tive o privilégio de ser aluno.

"Este é meu Filho amado, em quem muito me agrado."(Mateus 3:17)

RESUMO

O avanço da computação quântica ameaça a criptografia contemporânea ao quebrar criptossistemas baseados na dificuldade computacional da fatoração de números inteiros com fatores primos grandes. Em resposta, instituições de segurança desenvolveram algoritmos pós-quânticos, os quais, assim como seus antecessores, estão sujeitos a ataques criptoanalíticos. *Ciphertext only attack* é um tipo de ataque utilizado na criptoanálise e é considerado bem-sucedido se qualquer informação do texto cifrado ou da chave puder ser deduzida. Uma das tarefas mais importantes na criptoanálise consiste na identificação do algoritmo de criptografia utilizado. A revisão da literatura revela que poucos trabalhos analisaram criptogramas exclusivamente provenientes de cifras assimétricas, e nenhum deles avaliou especificamente criptossistemas pós-quânticos. Portanto, em um cenário *only ciphertext*, no qual apenas amostras criptografadas estão disponíveis ao pesquisador, este trabalho de dissertação propõe uma metodologia baseada nos testes estatísticos da NIST STS e nos algoritmos de aprendizado de máquina *Support Vector Machine*, *Random Forest*, *Naive Bayes* e *K-Nearest Neighbor* para identificar padrões nos criptogramas e nas chaves de sessão encapsuladas provenientes dos criptossistemas pós-quânticos Frodo, Crystals Kyber, NTRU e Saber, nos modos de operação ECB e CBC, a fim de identificá-los posteriormente. Como contribuições desta pesquisa, destaca-se a detecção de padrões em criptossistemas pós-quânticos, à luz do estado da arte, bem como o desenvolvimento de um método para identificar cifras operando no modo CBC. A pesquisa realizou uma análise cuidadosa da influência do tamanho do criptograma nas medidas de desempenho dos classificadores, contribuindo para o avanço do conhecimento nessa área específica. Os resultados obtidos revelaram a presença de padrões únicos tanto nos criptogramas quanto nas chaves de sessão gerados pelos criptossistemas analisados, evidenciando a presença de "assinaturas"nesses arquivos. O melhor desempenho foi obtido pelo classificador KNN, que em cenários específicos identificou os algoritmos analisados com uma acurácia que atingiu até 100%.

Palavras-chave: Criptografia pós-quântica; Identificação de padrões; Ataque de distinção; Aprendizado de máquina; *NIST statistical tests*; *Electronic Codebook*; *Cipher Block Chaining*; Mecanismo de encapsulamento de chaves.

ABSTRACT

The development of quantum computing poses a threat to contemporary cryptography, given the capability of these computers to break cryptographic systems based on the practical difficulty of factoring large prime numbers and solving the discrete logarithm problem in finite fields. In response, security institutions have developed post-quantum algorithms, which, like their predecessors, are susceptible to cryptanalytic attacks. Ciphertext-only attack is a type of attack used in cryptanalysis and is considered successful if any information from the ciphertext or the key can be deduced. One of the most important tasks in cryptanalysis is the identification of the encryption algorithm used. The literature review reveals that few studies have exclusively analyzed cryptograms originating from asymmetric ciphers, and none of them have specifically evaluated post-quantum cryptosystems. Therefore, in a ciphertext-only scenario, where only encrypted samples are available to the researcher, this dissertation proposes a methodology based on statistical tests from NIST STS and machine learning algorithms such as Support Vector Machine, Random Forest, Naive Bayes, and K-Nearest Neighbor to identify patterns in the cryptograms and encapsulated session keys from post-quantum cryptosystems Frodo, Crystals Kyber, NTRU, and Saber, in ECB and CBC operation modes, for subsequent identification. As contributions of this research, the detection of patterns in post-quantum cryptosystems is highlighted, considering the state of the art, as well as the development of a method to identify ciphers operating in CBC mode. The research conducted a careful analysis of the influence of the cryptogram size on the performance measures of classifiers, contributing to the advancement of knowledge in this specific area. The results obtained revealed the presence of unique patterns in both the cryptograms and session keys generated by the analyzed cryptosystems, demonstrating the existence of "signatures" in these files. The best performance was achieved by the KNN classifier, which in specific scenarios identified the analyzed algorithms with an accuracy reaching up to 100%.

Keywords: Post-quantum cryptography; Pattern recognition; Distinguishing attack; Machine learning; NIST statistical tests; Electronic Codebook; Cipher Block Chaining; Key encapsulation mechanism.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3DES Triple DES

AES Advanced Encryption Standard

ARM Advanced RISC Machine

Ba Bagging

CBC Cipher Block Chaining

CFB Cipher Feedback

DT Decision Tree

DES Data Encryption Standard

ECB Electronic Code Book

IBL Instance based learning

KNN K-Nearest Neighbors

LDA Linear Discriminant Analysis

LR Logistic Regression

MLP Multi-Layer Perceptron

NB Naive Bayes

NN Neural network

OFB Output Feedback

RF Random Forest

RoFo Rotation Forest

SVM Support Vector Machine

SUMÁRIO

1 INTRODUÇÃO

A criptografia, uma ciência antiga que envolve a escrita codificada para garantir o sigilo de informações sensíveis durante o armazenamento ou transmissão, é usada para preservar a autenticidade, confidencialidade e integridade dos dados (1). Segundo Diffie e Hellman(13), esse processo é realizado por meio de um algoritmo que transforma a mensagem original em uma mensagem cifrada, permitindo que somente o remetente e o destinatário legítimo possam decifrá-la e recuperar a mensagem original. Essa medida de segurança assegura que mesmo que a mensagem seja interceptada por terceiros mal-intencionados durante a transmissão, ela permanecerá confidencial e não poderá ser compreendida sem a utilização da chave criptográfica adequada.

Ao longo do tempo, essa técnica foi aperfeiçoada e processos mecânicos e eletro-mecânicos foram amplamente empregados durante as duas Guerras Mundiais, incluindo as famosas máquinas de cifra Enigma utilizadas pelo exército alemão. Com o avanço da computação, surgiram cifras mais fortes, como o DES (*Data Encryption Standard*) e o AES (*Advanced Encryption Standard*), que oferecem maior segurança para as informações.

O surgimento da criptografia de chave pública na década de 70 representou um marco importante na história da criptografia. Um artigo que aborda este tema é *New Directions in Cryptography*, Diffie e Hellman(13), publicado na revista *IEEE Transactions on Information Theory* em 1976. O artigo apresenta o conceito de criptografia de chave pública, que permite que duas pessoas se comuniquem de forma segura sem a necessidade de compartilhar previamente uma chave secreta. Diffie e Hellman(14) também propõem um método para troca de chaves seguras, conhecido como *Diffie-Hellman key exchange*, que se tornou uma das bases da criptografia moderna. A abordagem inovadora da criptografia de chave pública proporcionou o aperfeiçoamento das assinaturas digitais, que oferecem maior segurança e flexibilidade aos sistemas criptográficos, com base na chave secreta do emissor (15).

Conforme Schneier(16), a criptoanálise é uma ciência dedicada à quebra de códigos e obtenção de informações secretas a partir de mensagens cifradas, sem prévio conhecimento dos elementos usados na criptografia. Segundo Tan e Ji(17), o trabalho criptoanalítico ocorre em diferentes cenários, explicados na Tabela 1 e apresentados a seguir em ordem crescente de dificuldade: *Chosen plaintext and ciphertext*, *Chosen ciphertext*, *Chosen plaintext*, *Known plaintext* e *Only ciphertext*. Dileep e Sekhar(18) destacam a identificação do algoritmo de criptografia e da chave como tarefas essenciais na criptoanálise, especialmente quando apenas textos cifrados estão disponíveis para o pesquisador, como no cenário mais desafiador, o *Only ciphertext*. Na prática, um criptoanalista possui poucas informações e desconhece o algoritmo usado para cifrar o texto claro. Portanto, identificar o criptossistema

é uma das atividades fundamentais para reduzir o esforço criptoanalítico e pode contribuir significativamente para decodificar a mensagem cifrada (19).

Tabela 1 – Meios de Ataque e Informações Disponíveis para um Criptoanalista

Meios de Ataque	**Informações Disponíveis para um Criptoanalista**
Only ciphertext	Algoritmo criptográfico Texto cifrado a ser decifrado
Known plaintext	Algoritmo criptográfico Texto cifrado a ser decifrado Pares de texto em claro e texto cifrado
Chosen plaintext	Algoritmo criptográfico Texto cifrado a ser decifrado Texto em claro escolhido por um criptoanalista e o texto cifrado correspondente
Chosen ciphertext	Algoritmo criptográfico Texto cifrado a ser decifrado Texto cifrado escolhido por um criptoanalista e o texto em claro correspondente
Chosen plaintext and ciphertext	Algoritmo criptográfico Texto cifrado a ser decifrado Texto em claro escolhido por um criptoanalista e o texto cifrado correspondente Texto cifrado escolhido por um criptoanalista e o texto em claro correspondente

Knudsen e Meier(20) foram pioneiros ao cunhar o termo "ataque de distinção"para se referir ao processo de identificação de algoritmos de criptografia. Inicialmente, esse conceito consistia em diferenciar um algoritmo criptográfico de uma sequência aleatória gerada por um gerador de números aleatórios. Com o tempo, essa abordagem foi expandida para distinguir criptogramas com o objetivo de identificar algoritmos com base no texto cifrado. Essa área tem sido alvo de muitas pesquisas, que buscam detectar padrões e classificar criptogramas gerados por diferentes criptossistemas. Segundo Nagireddy(21), ataques de identificação ou de distinção também podem ser aplicados para avaliar a força das cifras, além de reduzir o espaço da chave ou descriptografar mensagens codificadas. Identificar o algoritmo criptográfico e o modo de operação utilizados é a primeira tarefa de um criptoanalista em sua pesquisa, e recentemente, o uso de classificadores de aprendizado de máquina tem ganhado destaque nessa tarefa.

No artigo *Post-Quantum Cryptography* de Ding, Gower e Stinson(22), os autores discutem os desafios que a computação quântica representa para a criptografia contemporânea devido ao fato de tais computadores possuírem capacidade para quebrar criptossistemas baseados na dificuldade prática de fatoração do produto de grandes números primos e na resolução do problema do logaritmo discreto em corpos finitos.

Contrapondo essa ameaça, diversas instituições públicas e privadas relacionadas à segurança da informação desenvolveram algoritmos resistentes a ataques perpetrados

por computadores quânticos, processo que originou a criptografia pós-quântica. Desde 2016, o *National Institute of Standards and Technology* (NIST) tem realizado um concurso para atualizar os padrões criptográficos americanos, por meio da seleção de um ou mais algoritmos pós-quânticos (23). Outras agências governamentais e organizações em várias regiões do mundo, incluindo a União Europeia (24) e a China (25), também estão conduzindo processos de padronização e adoção de algoritmos pós-quânticos semelhantes aos do NIST. No Brasil, a Agência Brasileira de Inteligência (ABIN) lançou a biblioteca criptográfica *libharpia*, que suporta os algoritmos pós-quânticos Crystals Dilithium para assinatura digital e Crystals Kyber para encapsulamento de chaves. Ela foi apresentada no XXII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (26) e utilizada nas eleições de outubro e novembro de 2022.

A criptografia pós-quântica naturalmente sucederá a criptografia contemporânea na medida que a capacidade computacional dos computadores quânticos aumentar e é preciso conduzir pesquisas de identificação de padrões ou ataques de distinção em criptogramas gerados por criptossistemas pós-quânticos, à luz das pesquisas conduzidas anteriormente, que analisaram e identificaram algoritmos clássicos, como o DES, o Blowfish e o AES, entre outros.

Sobre este tema, vale destacar que o Instituto Militar de Engenharia (IME) tem um histórico significativo de trabalhos realizados desde 2006 sobre ataques de distinção, o que revela a importância e o comprometimento dessa instituição em relação a esta área. Assim, contribuir para este legado e para o aprimoramento dessa especialidade do IME também é um fator motivacional para o autor desta dissertação.

1.1 Motivação

Devido à possibilidade de que o avanço dos computadores quânticos comprometa a criptografia assimétrica contemporânea, diversos algoritmos de chave pública e híbridos pós-quânticos estão sendo desenvolvidos. Segundo Mosca(27), esses algoritmos são fundamentais para garantir a segurança a longo prazo da comunicação digital e provavelmente serão adotados como o novo padrão criptográfico, substituindo naturalmente as cifras utilizadas atualmente. A relevância deste fato motivou o NIST a estabelecer a categoria *Public-key Encryption and Key-establishment Algorithms* no seu atual concurso seletivo (23).

Na literatura, várias abordagens têm sido usadas para ataques de distinção, incluindo o modelo *bag-of-words* e a distribuição qui-quadrado. No entanto, o atual estado da arte, representado pelas recentes pesquisas de Xia, Li e Chen(28), Yuan et al.(11), Yuan et al.(12) e Yu e Shi(29), indica a utilização de algoritmos de aprendizado de máquina e valores p dos testes estatísticos da suíte NIST SP 800-22. Essa suíte, desenvolvida pelo NIST na década de 2000, foi projetada para validar geradores de números aleatórios e

pseudoaleatórios em aplicações criptográficas (30). Esses autores limitaram-se a analisar algoritmos clássicos com exploração limitada dos recursos disponíveis na suíte do NIST e no conjunto utilizado de classificadores. Seus experimentos não alcançaram 100% de acurácia na identificação dos algoritmos analisados. Essa abordagem talvez seja mais eficaz otimizando tanto o emprego dos recursos da suíte quanto os hiperparâmetros dos classificadores.

A presente dissertação procura eliminar a lacuna existente na literatura sobre pesquisas de identificação, que se restringiram a criptossistemas clássicos, e, à luz do estado da arte, emprega todos os recursos disponíveis na suíte NIST SP 800-22 e ferramentas de aprendizado de máquina para identificar algoritmos pós-quânticos participantes do atual concurso seletivo do NIST por meio de seus criptogramas em um cenário *only ciphertext*.

1.2 Caracterização do Problema

Em novembro de 2022, a *International Business Machines* (IBM) apresentou o *Osprey*, o maior computador quântico produzido até então, com 433 qubits, mais que o triplo da capacidade do *Eagle*, seu antecessor com 127 qubits (31). Recentemente, Yan et al.(32) apresentaram um algoritmo destinado a fatoração do produtos de grandes números primos e demonstraram experimentalmente sua eficácia em fatorar inteiros de 48 bits. Além disso, estimaram que um computador quântico com 372 qubits seria capaz de comprometer a criptografia RSA-2048.

O avanço dos computadores quânticos está diretamente relacionado com o desenvolvimento da criptografia pós-quântica, o que levou ao surgimento de diversas propostas avaliadas na categoria de *Public-key Encryption and Key-establishment Algorithms* durante a segunda e terceira rodada do concurso seletivo do NIST. A proposta Classic McEliece é baseada em códigos de Goppa, uma classe de códigos corretores de erros que se fundamenta na teoria dos códigos, enquanto as propostas Frodo (reticulados LWE - *Learning With Errors*), Crystals Kyber (reticulados MLWE - *Module Learning With Errors*), NTRU (reticulados NTRU) e Saber (reticulados MLWR - *Module Learning With Rounding*) são baseadas em reticulados. Todos os candidatos apresentaram abordagens distintas e foram projetados para proporcionar altos níveis de segurança baseados na dificuldade de quebrar variantes do AES e do SHA3.

A metodologia de ataque de distinção, conforme ilustrada na Figura 1, consiste em um sistema capaz de determinar qual algoritmo criptográfico foi empregado apenas com base no criptograma fornecido. Nesse método, um criptograma C é submetido ao sistema de identificação, que, ao analisar suas características, determina o algoritmo criptográfico E utilizado para cifrar o texto original. Segundo Kölbl(33), é fundamental que os algoritmos criptográficos sejam projetados para resistir a ataques de distinção, a fim de assegurar

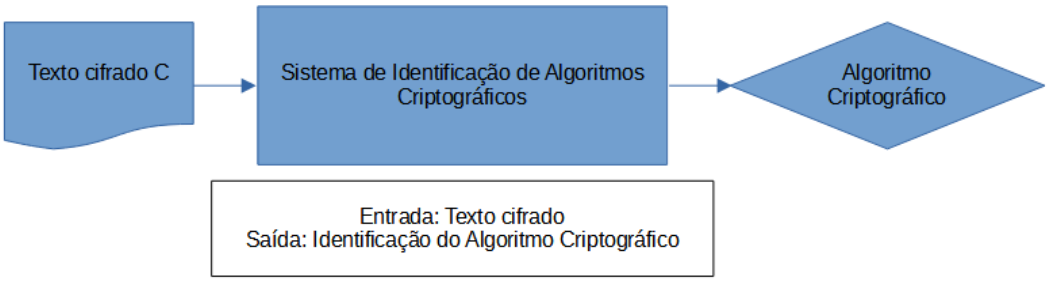


Figura 1 – Princípio de identificação de algoritmo de criptografia

níveis adequados de segurança em ambientes de comunicação segura.

Existem diversos estudos na literatura que abordam a identificação de cifras contemporâneas e modos de operação por meio de seus criptogramas. Segundo Nagireddy(21), a identificação da cifra e do modo de operação pode ser tratada como um problema de identificação e classificação de padrões, e, portanto, algoritmos de aprendizado de máquina podem ser uma abordagem útil. O objetivo é extrair informações e características dos criptogramas que permitam a construção de uma função supervisionada por algoritmos de aprendizado de máquina, que minimize os erros de classificação e possa classificar com precisão um novo criptograma.

Dentro do contexto apresentado, surge uma questão de pesquisa relevante no campo da criptografia pós-quântica: no cenário *only ciphertext*, é possível identificar padrões nos criptogramas e nas chaves de sessão gerados pelos algoritmos pós-quânticos Frodo, Crystals Kyber, NTRU e Saber e reconhecê-los posteriormente por meio de algoritmos de aprendizado de máquina?

1.3 Hipóteses

1. Por meio de análises estatísticas, é possível obter informações relevantes de textos cifrados gerados por algoritmos criptográficos pós-quânticos nos modos de operação *Electronic Codebook* (ECB) e *Cipher Block Chaining* (CBC); e

2. Com base nessas informações relevantes, é possível utilizar classificadores de aprendizado de máquina para realizar ataques de identificação ou distinção em criptogramas e nas chaves de sessão gerados e encapsulados, respectivamente, pelos esquemas de encriptação de chave pública e mecanismos de encapsulamento pós-quânticos.

As hipóteses apresentadas acima são justificadas pelo fato de que estudos recentes ((11), (12), (28), (29)) utilizaram testes estatísticos e algoritmos de aprendizado de máquina para identificar os algoritmos criptográficos clássicos AES, 3DES, Blowfish, CAST e RC2 nos modos ECB e CBC, com base em criptogramas. No entanto, nenhum desses estudos

alcançou acurácia máxima (100%) nem utilizou todos os recursos disponíveis na NIST STS SP 800-22.

1.4 Objetivos

Em um cenário de *only ciphertext*, o objetivo geral desta pesquisa consiste em propor uma metodologia de ataque de distinção e identificar padrões nos arquivos provenientes dos criptossistemas pós-quânticos Frodo, Crystals Kyber, NTRU e Saber.

Os objetivos específicos desta são:

1. Propor um método supervisionado que utilize algoritmos de aprendizado de máquina e todas as ferramentas estatísticas disponíveis na bateria do NIST e que seja aplicável aos modos de operação ECB e CBC;
2. Comparar os resultados obtidos com o estado da arte, a fim de avaliar a eficácia do método proposto e sua contribuição para o avanço da criptografia pós-quântica; e
3. Determinar se a abordagem proposta é capaz de aumentar as taxas de identificação de algoritmos criptográficos clássicos, em comparação com pesquisas anteriormente realizadas.

Seguem abaixo as contribuições esperadas por esta pesquisa:

1. Identificar padrões nos criptogramas e nas chaves de sessão encapsuladas provenientes dos algoritmos pós-quânticos Frodo, Crystals Kyber, NTRU e Saber;
2. Avaliar se a abordagem de identificação proposta é mais eficaz que os esquemas de identificação presentes na literatura especializada; e
3. Contribuir para a Estratégia Nacional de Segurança Cibernética (E-Ciber) por meio da metodologia de ataque de distinção proposta, fortalecendo a governança cibernética e seguindo as diretrizes do Governo Federal diante da crescente preocupação do Estado Brasileiro nesse campo.

1.5 Método de pesquisa

De acordo com Wazlawick(34), o método de pesquisa é composto por uma sequência de passos necessários para alcançar um determinado objetivo. O método empregado nesta pesquisa inclui as seguintes etapas:

1. **Tema e Problema de Pesquisa:** Este estudo aborda ataques de distinção, que se referem à capacidade de distinguir entre um cifrador de bloco e uma sequência aleatória com base em uma saída específica, conforme descrito por Kölbl(33). O problema desta pesquisa consiste em realizar um ataque de distinção em criptogramas gerados por esquemas de criptografia de chave pública dos mecanismos de encapsulamento de chaves pós-quânticos Frodo, Crystals Kyber, NTRU e Saber, nos modos de operação ECB e CBC, no cenário *only ciphertext*.

2. **Estudo dos trabalhos relacionados e identificação da abordagem utilizada:** Na etapa de seleção e análise da literatura, foram identificadas pesquisas que utilizaram algoritmos de aprendizado de máquina para solucionar o problema de identificação do algoritmo criptográfico. Os seguintes critérios foram adotados para análise dos trabalhos relacionados: modo de operação, tipos de chave e vetor de inicialização, tamanho e quantidade de amostras, acurácia registrada pelos classificadores e tipos de algoritmo criptográfico e de aprendizado de máquina. As pesquisas encontradas foram relevantes para orientar a escolha dos algoritmos de aprendizado de máquina utilizados nesta pesquisa, bem como para definir as características das amostras de criptogramas coletadas para os experimentos.

3. **Delimitação do problema e determinação da hipótese e objetivo**: Após identificar a lacuna existente na literatura em relação à realização de ataques de distinção em algoritmos pós-quânticos e delimitar o escopo deste trabalho, foram definidos os problemas a serem resolvidos, as hipóteses a serem testadas, bem como os objetivos geral e específicos da pesquisa, além das contribuições esperadas.

4. **Planejamento e implementação dos experimentos:** Com base nos experimentos realizados pelo estado da arte atual, uma base de textos em língua portuguesa foi criptografada usando diferentes chaves e vetores de inicialização, nos modos ECB e CBC, pelos algoritmos Blowfish, AES, Frodo, Crystals Kyber, NTRU e Saber. Os textos cifrados foram representados por vetores compostos pelos valores p obtidos após submissão dos criptogramas gerados a bateria de testes de aleatoriedade NIST STS SP 800-22. Esses vetores foram classificados de acordo com o algoritmo criptográfico e compuseram a base de entrada dos algoritmos de aprendizado, sendo divididos em conjuntos de treinamento, teste e validação. A confirmação da hipótese desta pesquisa ocorrerá caso os algoritmos consigam encontrar fronteiras de decisão que possibilitem distinguir os criptogramas, com uma acurácia maior do que um acerto aleatório.

5. **Análise dos resultados e comparação com trabalhos relacionados:** Nesta etapa da pesquisa, os resultados obtidos são interpretados e discutidos em relação ao problema de pesquisa e às hipóteses testadas. A comparação com outras pesquisas

existentes na literatura é importante para contextualizar os resultados obtidos e avaliar a contribuição desta pesquisa para o avanço do conhecimento científico.

1.6 Organização da Dissertação

Esta dissertação encontra-se estruturada em cinco Capítulos a fim de atingir aos objetivos acima expostos.

No Capítulo 2 são abordados conceitos relevantes para o desenvolvimento deste trabalho relacionados aos algoritmos criptográficos e de aprendizado de máquina utilizados nesta pesquisa, incluindo a criptografia simétrica e assimétrica, as cifras de bloco e de fluxo, bem como os modos de operação ECB e CBC.

No Capítulo 3 é apresentada a revisão bibliográfica realizada nesta pesquisa, na qual se identificam as abordagens adotadas por diversos autores no tema de identificação de padrões em criptogramas. Além disso, é destacada a lacuna de pesquisa identificada no objetivo e nas hipóteses deste trabalho.

No Capítulo 4, é detalhado o método de pesquisa desenvolvido para ser aplicável em ambos os modos de operação analisados e apresentada a sua implementação, além dos resultados obtidos nos experimentos executados.

Por último, no Capítulo 5, os resultados obtidos nos experimentos realizados no Capítulo anterior são comparados com os trabalhos anteriores identificados durante a revisão da literatura. Conclusões acerca das hipóteses e objetivos também são discutidas, bem como são apresentados a conclusão, as contribuições obtidas e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais para a compreensão desta dissertação. Nas seções e subseções seguintes são apresentadas definições relacionadas a sistemas criptográficos simétricos e assimétricos, cifras de fluxo e de bloco, modos de operação, reconhecimento de padrões em criptogramas, geradores de números aleatórios e pseudoaleatórios, estatística p-valor, bateria de testes estatísticos do NIST, criptografia pós-quântica e algoritmos de aprendizado de máquina.

2.1 Sistemas criptográficos

2.1.1 Sistemas simétricos

Na criptografia simétrica, também denominada criptografia de chave secreta, uma única chave é empregada para cifrar e decifrar os dados (13). A Figura 2 ilustra um esquema de criptografia simétrica, que possui quatro componentes essenciais: o texto claro, o algoritmo de criptografia (também conhecido como cifra), a chave secreta e o texto cifrado. O texto claro representa a mensagem original à qual se busca assegurar confidencialidade. Nesse contexto, a chave secreta é um componente empregado pelo algoritmo de criptografia para transformar o texto claro em texto cifrado.

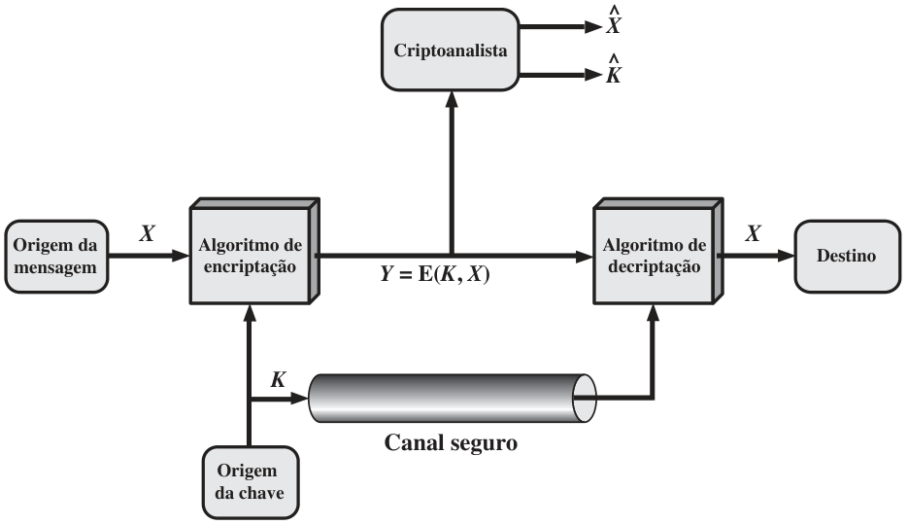


Figura 2 – Modelo de criptossistema simétrico. Adaptado de (1)

De acordo com (1), a segurança da criptografia simétrica depende da capacidade do algoritmo em evitar que um adversário, com acesso aos textos cifrados e ao algoritmo, consiga deduzir a chave utilizada ou decifrar os criptogramas.

A criptografia simétrica é inadequada em cenários que envolvem comunicação segura entre múltiplos usuários, uma vez que cada par de usuários necessita de uma chave secreta única, o que torna complexo o processo de distribuição e compartilhamento dessas chaves e limita significativamente sua utilidade em contextos mais abrangentes. No entanto, é possível superar essa dificuldade por meio de um mecanismo ou sistema de distribuição de chaves eficiente.

2.1.2 Sistemas assimétricos

Na criptografia assimétrica, ao contrário dos sistemas simétricos, cada usuário possui um par de chaves distinto: uma pública e outra privada. Essas chaves podem ser geradas pelo próprio usuário ou por uma terceira parte de confiança. A chave pública é divulgada a todos os usuários do sistema e serve para criptografar mensagens destinadas ao proprietário da chave. Por outro lado, a chave privada deve ser estritamente mantida em segredo, pois é usada para decifrar as mensagens criptografadas com a chave pública correspondente.

A combinação de sistemas assimétricos e simétricos é empregada para resolver o problema da distribuição segura de chaves de sessão, uma vez que os sistemas assimétricos são utilizados para distribuir chaves de forma segura e autenticar usuários, enquanto os sistemas simétricos são empregados para a comunicação propriamente dita, assegurando eficiência e segurança nos processos de troca de informações sensíveis. Essa abordagem híbrida, que origina os mecanismos de encapsulamento de chaves (KEM) (35), maximiza a segurança e a praticidade na corrente era digital.

Conforme Stallings (1), deve ser computacionalmente inviável determinar a chave privada a partir de textos cifrados ou da chave pública em criptografia. As Figuras 3 e 4 ilustram todos os componentes essenciais de um criptossistema assimétrico.

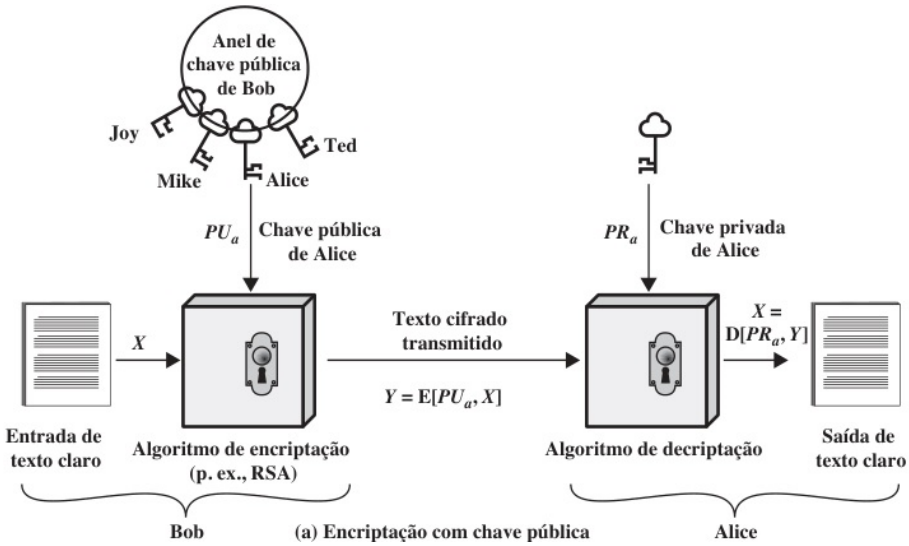


Figura 3 – Encriptação com chave pública em um criptossistema assimétrico(1)

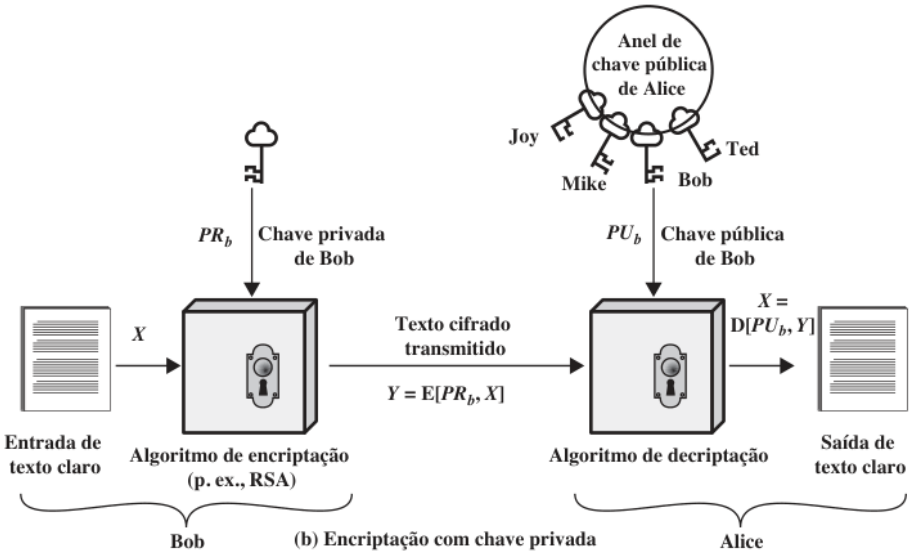


Figura 4 – Encriptação com chave privada em um criptossistema assimétrico(1)

2.1.3 Cifras de bloco e de fluxo

A cifra de bloco é um criptossistema que divide o texto claro em blocos de bits de tamanho predefinido, geralmente 64 ou 128 bits. Cada bloco é criptografado separadamente usando uma chave de cifragem exclusiva. A encriptação é realizada por meio de uma função de cifragem, que deve ser bijetora para possibilitar a decriptação, seu processo inverso (36). A escolha do tamanho dos blocos e das chaves de cifragem tem um impacto direto na resistência da cifra contra ataques criptoanalíticos. Um exemplo amplamente utilizado de cifra de bloco é o *Advanced Encryption Standard* (AES), que opera com blocos de 128 bits e chaves de 128, 192 e 256 bits.

A cifra de fluxo opera no nível de bit a bit, diferentemente da cifra de bloco, que atua em blocos de bits. Utiliza uma chave pseudoaleatória combinada com o texto claro por meio da operação "ou exclusivo"(XOR) para gerar o texto cifrado (1). A Figura 5 fornece uma ilustração comparativa entre a cifra de fluxo e a cifra de bloco, destacando suas diferenças essenciais.

2.2 Modo de operação

Para viablizar a utilização das cifras em diferentes aplicações, o NIST estabeleceu os modos de operação *Electronic Codebook* (ECB), *Cipher Block Chaining* (CBC), *Output Feedback* (OFB), *Counter* (CTR) e *Cipher Feedback* (CFB) na FIPS SP 800-38A. Cada modo de operação possui características específicas e é adequado para cenários específicos. A escolha do modo de operação apropriado depende dos requisitos de segurança, eficiência e praticidade da aplicação em questão. É importante ressaltar que a escolha incorreta do modo de operação pode comprometer a segurança do sistema de criptografia (37). A

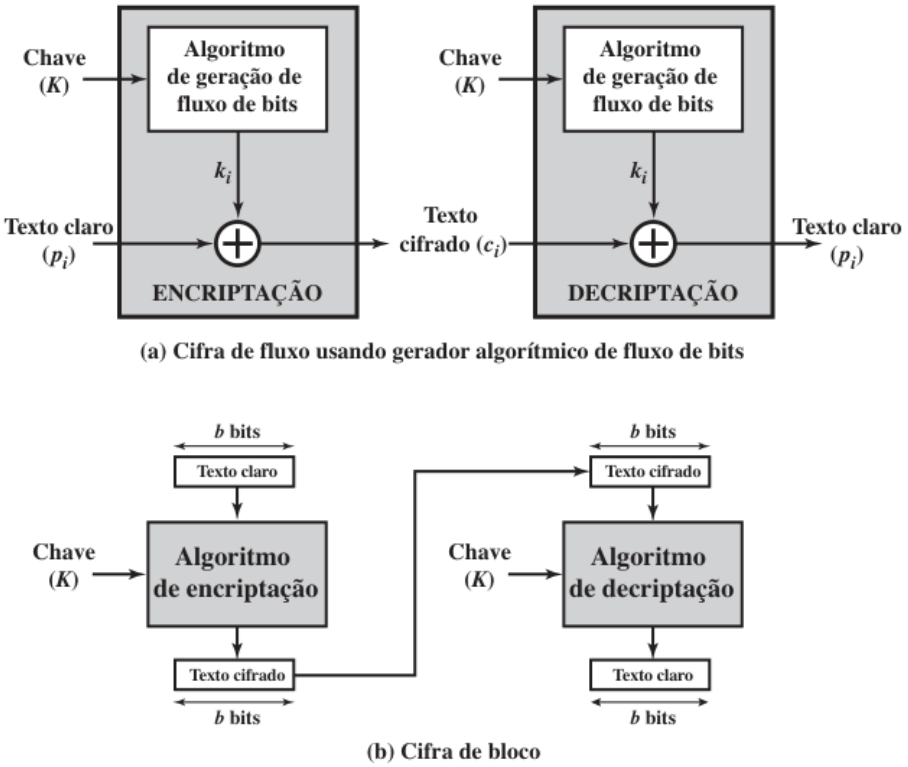


Figura 5 – Cifras de fluxo e de bloco. Adaptado de (1)

seguir, serão explicados os modos ECB e CBC.

2.2.1 Eletronic Codebook

O modo ECB, ilustrado na Figura 6, é o mais simples entre os modos de operação definidos pelo NIST (37). Nesse método, o texto claro é dividido em blocos de tamanho fixo, e cada bloco é criptografado de forma independente usando a mesma chave. Essa abordagem permite a paralelização, agilizando o processo de cifragem em comparação com outros modos de operação. No entanto, uma desvantagem significativa do modo ECB é que blocos idênticos de texto claro resultam em blocos de texto cifrado igualmente idênticos quando a mesma chave é usada. Isso pode facilitar ataques em certas circunstâncias (1).

Esses padrões repetitivos podem ser explorados por um adversário para identificar partes da mensagem cifrada e, potencialmente, decifrá-la. Portanto, o autor desaconselha o uso do modo ECB na maioria das aplicações de criptografia, exceto em casos específicos e limitados.

2.2.2 Cipher Block Chaining

O modo CBC, ilustrado na Figura 7, mitiga a vulnerabilidade do modo ECB ao garantir que blocos idênticos de texto claro resultem em blocos de texto cifrado diferentes, por meio do encadeamento dos blocos.

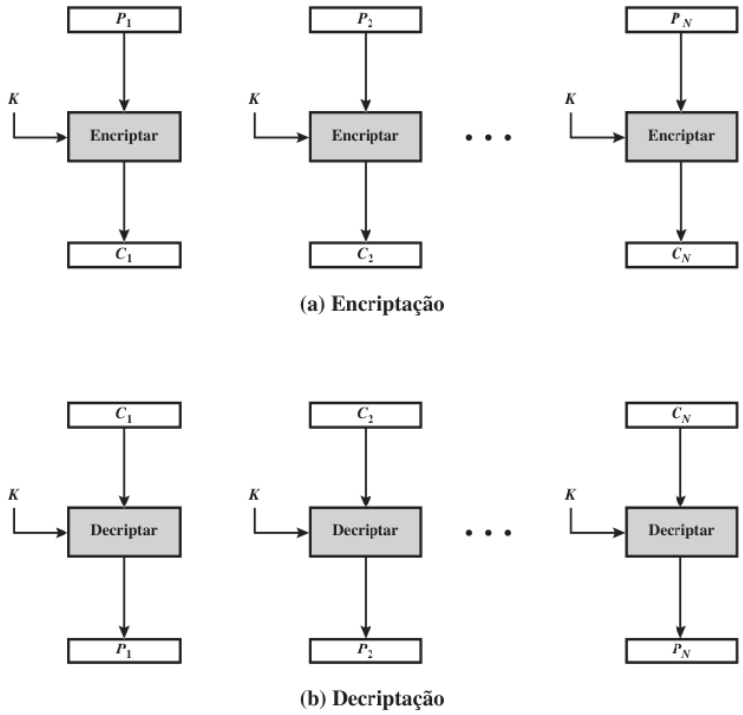


Figura 6 – Modo de operação ECB. Adaptado de (1)

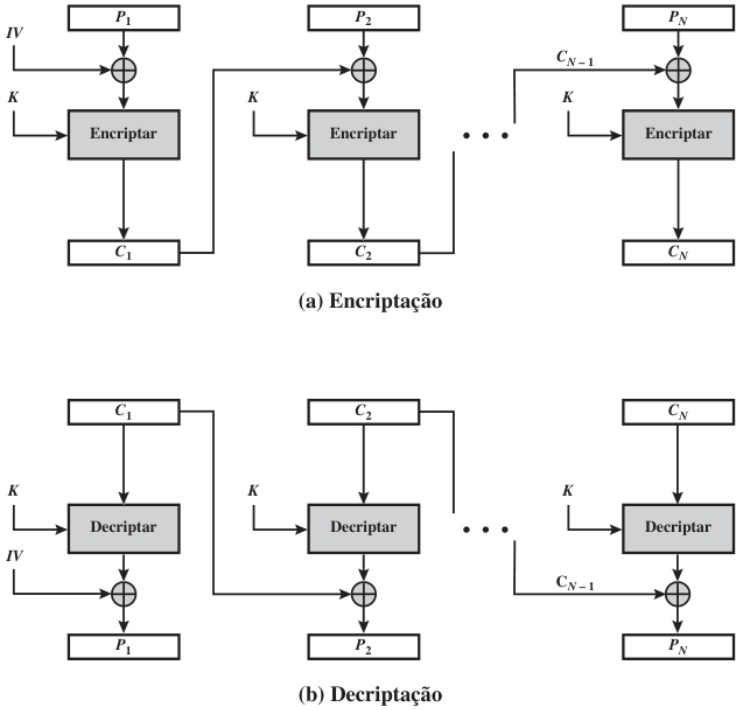


Figura 7 – Modo de operação CBC. Adaptado de (1)

Nesse modo, cada bloco de texto claro é combinado com o bloco de texto cifrado imediatamente anterior (ou com o vetor de inicialização (VI) no primeiro bloco), usando uma operação XOR. Embora não seja obrigatório, é comum utilizar um VI aleatório para cada mensagem, impedindo previsibilidade e garantindo a segurança da cifragem. Essa técnica cria dependência entre os blocos, reduzindo a presença de padrões nos blocos de

texto cifrado (37). No entanto, o modo CBC não permite paralelismo, o que resulta em uma operação mais lenta em comparação com o modo ECB.

2.3 Criptografia pós-quântica

Recentemente, houve um aumento significativo na quantidade de projetos dedicados à construção de computadores quânticos. Em 1994, Peter Shor (38) demonstrou que computadores quânticos são capazes de quebrar, em tempo polinomial, criptossistemas assimétricos baseados na dificuldade prática de fatorar o produto de grandes números primos e nos logaritmos discretos, como o RSA e o algoritmo Diffie-Hellman, tornando-os obsoletos.

Neste contexto, instituições públicas e privadas têm dedicado esforços significativos para desenvolver criptossistemas resistentes a ataques quânticos, originando a criptografia pós-quântica. Em 2016, o NIST (*National Institute of Standards and Technology*) iniciou um processo para selecionar mecanismos de encapsulamento de chaves resistentes a ataques perpetrados por computadores quânticos. Ao longo das 2ª e 3ª rodadas do concurso, quatro mecanismos baseados no problema matemático dos reticulados: Frodo (39), Crystals Kyber (2), NTRU (40) e Saber (41) foram selecionados pelo NIST na categoria *Public-key Encryption and Key-establishment Algorithms*, evidenciando que a criptografia baseada em reticulados é uma forte candidata para alcançar o objetivo de resistir a ataques quânticos, uma vez que nenhum algoritmo quântico conhecido resolve problemas de reticulados de forma eficiente (42) (43).

De acordo com a FIPS 203 (43), mecanismo de encapsulamento de chaves (Key-encapsulation mechanism - KEM) é um conjunto de algoritmos usado por duas partes para estabelecer uma chave secreta compartilhada em um canal público inseguro. Essa chave, seguramente estabelecida por um KEM, pode ser usada em algoritmos criptográficos de chave simétrica para criptografia e autenticação em comunicações seguras.

2.3.1 Criptografia baseada em reticulados

Os reticulados são conjuntos de pontos discretos no espaço Euclidiano $\mathbb{R}^n$ de dimensão n, definidos como as combinações lineares inteiras de vetores independentes (44). A Figura 8 ilustra um exemplo de reticulado formado pelas combinações lineares de $\mathbf{b}_1$ e $\mathbf{b}_2$. Por exemplo, o vetor $\mathbf{s}$ pode ser representado pela combinação $-2\mathbf{b}_1 + \mathbf{b}_2$.

Definição 1. Reticulado: Um reticulado Λ é um conjunto de combinações lineares inteiras dos vetores $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$, que são linearmente independentes em $\mathbb{R}^n$, e pertencem à base B, onde $m \leq n$. Cada vetor em Λ é formado pela soma dos produtos dos coeficientes inteiros u_i pelos vetores correspondentes $\mathbf{b}_i$ (45). Se $m = n$, chama-se este reticulado de reticulado de *rank* completo.

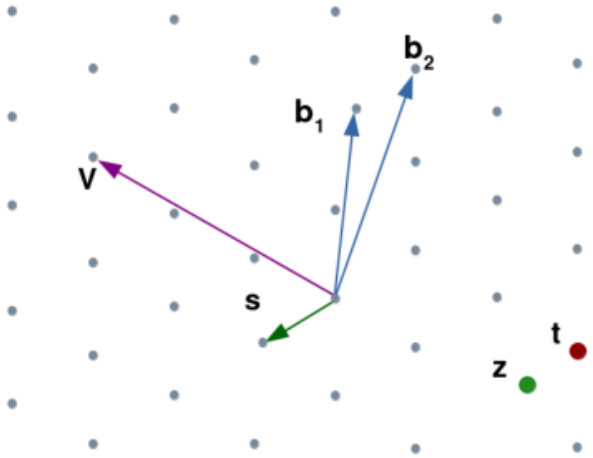


Figura 8 – Exemplo de reticulado formado pelos vetores $\mathbf{b}_1$ e $\mathbf{b}_2$

$$\Lambda = \left\{ \sum_{i=1}^{m} u_i \mathbf{b}_i \mid u_1, \ldots, u_m \in \mathbb{Z}, \mathbf{b}_i \in B \right\}$$

A razão de *Hadamard* $H(B)$ é uma medida associada à base B de um reticulado Λ. Para $B = \{\mathbf{u}_1\mathbf{b}_1 + \cdots + \mathbf{u}_m\mathbf{b}_m : \mathbf{u}_1, \ldots, \mathbf{u}_m \in \mathbb{Z}\}$, a razão de *Hadamard* é definida como:

$$H(B) = \left(\frac{1}{D(B)} \right)^{\frac{1}{n}}$$

onde $D(B)$ é o defeito ortogonal de B. O defeito ortogonal $D(B)$ é calculado como:

$$D(B) = \frac{\|\mathbf{b}_1\| \cdot \|\mathbf{b}_2\| \cdot \ldots \cdot \|\mathbf{b}_n\|}{|\det(B)|}$$

onde $\|\mathbf{b}_1\|, \|\mathbf{b}_2\|, \ldots, \|\mathbf{b}_n\|$ são as normas dos vetores $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$, e $\det(B)$ é o determinante da matriz formada pelos vetores da base B.

Quando $D(B) \geq 1$, tem-se que $0 < H(B) \leq 1$. Em uma base ortogonal, $D(B) = H(B) = 1$. À medida que a base se torna menos ortogonal, o defeito ortogonal aumenta, enquanto a razão de *Hadamard* se aproxima de 0.

A razão de *Hadamard* pode ser usada para classificar bases como boas ou ruins. Uma base boa para um reticulado é aquela em que $H(B)$ se aproxima de 1, indicando vetores curtos e quase ortogonais, ou seja, a base se aproxima da ortonormalidade. Por outro lado, uma base ruim possui valores próximos de 0 para $H(B)$, indicando vetores longos e não ortogonais. A Figura 9 ilustra exemplos de bases boas, representadas pela cor verde, e ruins, representadas pela cor vermelha.

Observa-se que bases boas têm utilidade na construção de sistemas criptográficos, onde chaves secretas são derivadas dessas bases sólidas e, a partir delas, geram-se bases ruins para serem usadas como chaves públicas.

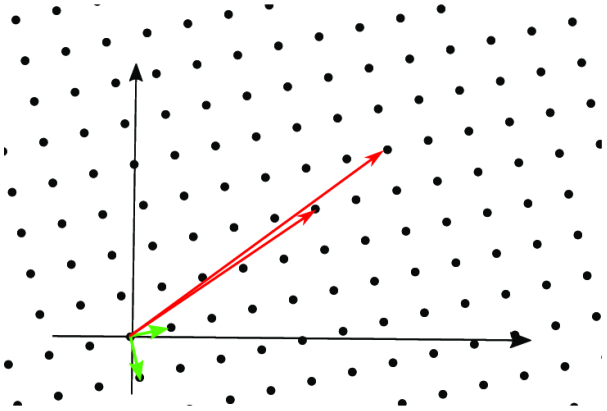


Figura 9 – Representação de bases em um reticulado Λ

A segurança de sistemas criptográficos baseados em reticulados em $\mathbb{R}^n$ fundamenta-se na complexidade de certos problemas matemáticos de difícil resolução, conhecidos como *hard lattice problems* (44). Um marco importante foi estabelecido por Ajtai (46), que introduziu a *Short Integer Solution* (SIS) como um método para encontrar o vetor mais curto em um reticulado, usando uma função unidirecional.

Regev(42) estabeleceu uma conexão entre sistemas criptográficos e reticulados por meio do método *Learning With Errors* (LWE), que se tornou a base dos criptossistemas baseados em reticulados, empregando os problemas NP-difíceis *Shortest Vector Problem* (SVP) e *Closest Vector Problem* (CVP).

Definição 2. Problema do Vetor Mais Curto (SVP): Dada uma base B para um reticulado Λ, encontrar o vetor não nulo mais curto em Λ (Figura 10).

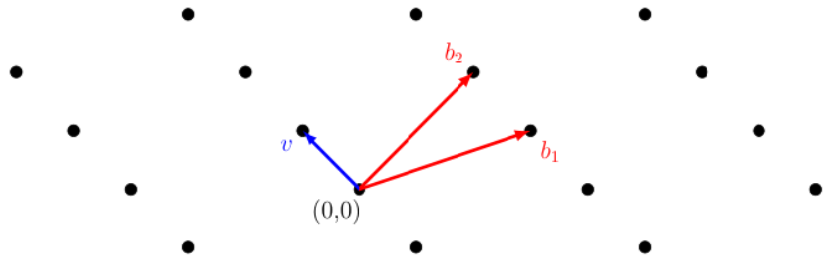


Figura 10 – Exemplo de SVP em um reticulado em $\mathbb{R}^2$ com 2 vetores de base b_1 e b_2 desenhados em vermelho. O vetor mais curto v está desenhado em azul.

Definição 3. Problema do Vetor Mais Próximo (CVP): Dada uma base B para um reticulado $\Lambda \subset \mathbb{R}^n$ e um ponto $p \in \mathbb{R}^n$, encontrar o vetor mais próximo em Λ a p (Figura 11).

No SVP, dado dois vetores de base $\mathbf{b}_1$ e $\mathbf{b}_2$ de um reticulado, o objetivo é encontrar o vetor mais curto pertencente ao reticulado formado por essas bases. Na Figura 8, esse vetor é representado por $\mathbf{s}$. No CVP, busca-se encontrar um vetor $\mathbf{z}$ que não pertence necessariamente ao reticulado, mas que seja o mais próximo possível de um vetor $\mathbf{t}$ que pertence ao reticulado.

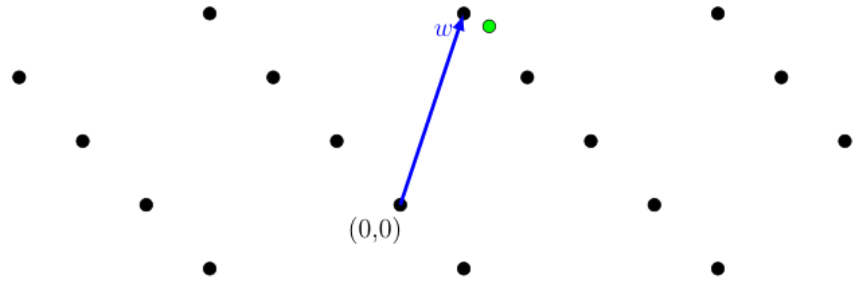


Figura 11 – Exemplo de CVP em um reticulado em $\mathbb{R}^2$. O vetor mais próximo ao ponto p, em verde, é o vetor w, desenhado em azul.

Ambos os problemas demonstraram ser resistentes não apenas a ataques clássicos, mas também teoricamente a ataques quânticos. A Figura 8 ilustra um reticulado bidimensional, embora, em aplicações criptográficas reais, os reticulados geralmente tenham dimensões muito maiores. Além disso, a escolha de bases pouco ortogonais entre si torna ainda mais desafiante a resolução desses problemas. Conforme observado por Harrigan (47), existem variantes desses problemas com implicações criptográficas significativas.

2.3.2 Problema *Learning With Errors*

A criptografia baseada em reticulados fundamenta-se no problema matemático chamado *Learning With Errors* (LWE). Nesta Subseção, apresentam-se os fundamentos desse problema.

Suponha um conjunto de equações lineares com um vetor secreto $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ de coeficientes inteiros, onde o objetivo seja determinar o vetor $\mathbf{s}$. Quando há um número suficiente de equações ($m \geq n$), a solução s para o problema pode ser encontrada em tempo polinomial.

$$
\begin{aligned}
a_{1,1}\mathbf{s}_1 + a_{1,2}\mathbf{s}_2 + \ldots + a_{1,n}\mathbf{s}_n &= a \\
a_{2,1}\mathbf{s}_1 + a_{2,2}\mathbf{s}_2 + \ldots + a_{2,n}\mathbf{s}_n &= b \\
&\vdots \\
a_{m,1}\mathbf{s}_1 + a_{m,2}\mathbf{s}_2 + \ldots + a_{m,n}\mathbf{s}_n &= m
\end{aligned}
$$

Entretanto, se as equações lineares estiverem apenas aproximadamente corretas, a solução torna-se não trivial, pois os erros em cada equação podem se acumular, levando a uma solução inconsistente ou significativamente distante do valor real.

A adição de um vetor de erro $\mathbf{e}$ ao produto $\mathbf{a} \cdot \mathbf{s}$ dificulta consideravelmente a determinação do vetor secreto $\mathbf{s}$. Sem esses erros, o sistema poderia ser resolvido usando eliminação de Gauss. No entanto, a adição contínua de erros resulta na perda de informações

úteis na aproximação obtida, dificultando assim a descoberta do vetor secreto $\mathbf{s}$, mesmo quando o erro $\mathbf{e}$ é pequeno (42).

$$
\begin{aligned}
a_{1,1}\mathbf{s}_1 + a_{1,2}\mathbf{s}_2 + \ldots + a_{1,n}\mathbf{s}_n + e_1 &\approx a \\
a_{2,1}\mathbf{s}_1 + a_{2,2}\mathbf{s}_2 + \ldots + a_{2,n}\mathbf{s}_n + e_2 &\approx b \\
&\vdots \\
a_{m,1}\mathbf{s}_1 + a_{m,2}\mathbf{s}_2 + \ldots + a_{m,n}\mathbf{s}_n + e_m &\approx m
\end{aligned}
$$

Resolver o sistema linear e, em seguida, arredondar para o inteiro mais próximo não garante uma solução precisa (47). Por exemplo, considerando $s = (3, 7)$ e $e_1 = e_2 = -1$, tem-se o seguinte sistema linear:

$$
\begin{aligned}
5s_1 + 3s_2 &\approx 35 \\
4s_1 + 2s_2 &\approx 27
\end{aligned}
$$

Após a representação matricial e a redução por linha, obtém-se a solução $s = \left(\frac{11}{2}, \frac{5}{2}\right)$, que pode ser arredondada para $s = (6, 3)$. Essa solução, distante do valor original, ilustra a complexidade da resolução do problema CVP.

De acordo com Harrigan (47), resolver o problema LWE em n-dimensões implica em uma solução igualmente eficaz para um problema de reticulado com dimensão $\sqrt{n}$. Essa relação, combinada com a falta, até agora, de algoritmos capazes de resolver problemas de reticulado, foi o impulso para o desenvolvimento de algoritmos de criptografia baseados em LWE (48).

Com o tempo, surgiram novas variantes do LWE, como o *Ring*-LWE proposto por (49). Nessa variante, as operações são realizadas em anéis polinomiais em vez de vetores n-dimensionais, resultando em chaves de tamanho menor.

Outra variante do LWE, chamada *Module*-LWE (MLWE), foi apresentada em (50) e (51). Nessa versão, os elementos do anel são substituídos por elementos modulares de um mesmo anel, utilizando-se um anel de potência de dois, como $\mathbb{Z}[X]/\langle X^n + 1\rangle$ com $n = 2k$ (49).

O MLWE tornou-se padrão em criptossistemas pós-quânticos, como o NIST, que se baseia no KEM Crystals Kyber. Isso se deve às vantagens proporcionadas por esse tipo de anel, que, em determinados cenários, resultam em um melhor desempenho nas operações (52). De acordo com a FIPS 203 (43), acredita-se, até o momento, que o MLWE seja seguro contra adversários com acesso a computadores quânticos.

2.3.3 Crystals Kyber

O *Cryptographic Suite for Algebraic Lattices Kyber*, conhecido como Crystals Kyber, é o mecanismo de encapsulamento de chaves pós-quântico adotado pelo governo norte-americano. As mensagens cifradas pelo seu esquema de encriptação de chave pública possuem 256 bits, permitindo sua integração com algoritmos de chave simétrica, como o AES. Isso possibilita o desenvolvimento de protocolos híbridos que combinam sistemas simétricos e assimétricos, podendo suportar criptografia pós-quântica e clássica. O processo de criptografia de chave pública do Crystals Kyber envolve três operações principais: geração de chaves, encriptação e decriptação (2).

O Crystals Kyber utiliza parâmetros específicos, indicados na Tabela 2, que consistem em inteiros positivos k, dt, du, dv, mantendo $n = 256$ constante. Esses parâmetros são ajustados para escalonar o nível de segurança do algoritmo, atendendo às diversas exigências de segurança do NIST. No espaço de mensagens $M = \{0,1\}^{256}$, cada mensagem $m \in M$ é um polinômio em R com coeficientes em $\{0,1\}$. O esquema de criptografia de chave pública Kyber.PKE = (KeyGen, Enc, Dec) é explicado nos algoritmos apresentados na Figura 12. Os textos cifrados são representados por $(u,v) \in \{0,1\}^{256\cdot k\cdot du} \times \{0,1\}^{256\cdot dv}$ (2).

No Kyber, todas as entradas e saídas das funções são tratadas como arrays de bytes. Para simplificar a notação, utiliza-se B para denotar o conjunto $\{0, \ldots, 255\}$, ou seja, o conjunto de inteiros sem sinal de 8 bits (bytes). Consequentemente, B_k representa o conjunto de arrays de bytes com comprimento k, e B^* representa o conjunto de arrays de bytes de comprimento arbitrário (ou fluxos de bytes) (2).

Tabela 2 – Parâmetros Crystals Kyber

Versão	**Nível de Segurança**	**Parâmetros**				
		n	**k**	**q**	(η_1, η_2)	**(du, dv)**
Kyber 512	1	256	2	3329	(3,2)	(10,4)
Kyber 768	3	256	3	3329	(2,2)	(10,4)
Kyber 1024	5	256	4	3329	(2,2)	(11,5)

No processo de geração do par de chaves pública e privada, inicialmente ocorre a geração da chave privada $\mathbf{s}$ e do vetor de erros $\mathbf{e}$. Em seguida, a chave pública, representada pelo vetor $\mathbf{t}$, é obtida multiplicando e somando matrizes, seguindo a forma $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.

$$\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \ldots & a_{kk} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_k \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{pmatrix}$$

No processo de cifragem de um bloco de texto claro m de 256 bits, a mensagem é inicialmente convertida para um polinômio, onde cada bit da mensagem corresponde

Algorithm 1 Kyber.CPA.KeyGen(): geração de chaves

Saída: Chave secreta $sk \in B^{12 \cdot k \cdot n/8}$
Saída: Chave pública $pk \in B^{12 \cdot k \cdot n/8+32}$

1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $A \leftarrow \mathcal{R}_q^{k \times k} := \text{Sam}(\rho)$ ▷ Gera matriz $\hat{A} \in \mathbb{R}_q^{k \times k}$ no domínio NTT
3: $(s, e) \leftarrow \beta_\eta^k \times \beta_\eta^k := \text{Sam}(\sigma)$ ▷ Amostra $s \in \mathbb{R}_q^k$ de B_{η_1}, $e \in \mathbb{R}_q^k$ de B_{η_1}
4: $t := \text{Compress}_q(As + e)$
5: **retorne** $(pk := (t, \rho), sk := s)$ ▷ $pk := \text{As} + e$, $sk := s$

Algorithm 2 Kyber.CPA.Enc(pk = (t, ρ), m $\in M$): encriptação

Entrada: Chave pública $pk \in B^{12 \cdot k \cdot n/8+32}$
Entrada: Mensagem $m \in B^{32}$
Entrada: *Seed* aleatória $r \in B^{32}$
Saída: Cifra $c \in B^{d_u \cdot k \cdot n/8 + dv \cdot n/8}$

1: $r \leftarrow \{0,1\}^{256}$
2: $t := \text{Decompress}_q(t)$
3: $A \leftarrow \mathcal{R}_q^{k \times k} := \text{Sam}(\rho)$ ▷ Gera matriz $\hat{A} \in \mathbb{R}_q^{k \times k}$ no domínio NTT
4: $(r, e_1, e_2) \leftarrow \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \text{Sam}(r)$ ▷ Amostra $r \in \mathbb{R}_q^k$ de B_{η_1}, $e_1 \in \mathbb{R}_q^k$ de B_{η_2}, $e_2 \in \mathbb{R}_q$ de B_{η_2}
5: $u := \text{Compress}_q(A^T r + e_1, d_u)$
6: $v := \text{Compress}_q(t^T r + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
7: **retorne** $c := (u, v)$

Algorithm 3 Kyber.CPA.Dec(sk = s, c = (u, v)): decifragem

Entrada: Chave secreta $sk \in B^{12 \cdot k \cdot n/8}$
Entrada: Cifra $c \in B^{d_u \cdot k \cdot n/8 + dv \cdot n/8}$
Saída: Mensagem $m \in B^{32}$

1: $u := \text{Decompress}_q(u, d_u)$
2: $v := \text{Decompress}_q(v, d_v)$
3: **retorne** $\text{Compress}_q(v - s^T u, 1)$ ▷ $m := \text{Compress}_q(v - s^T u, 1))$
4: **retorne** m

Figura 12 – Algoritmos Crystals Kyber.PKE. Adaptado de (2)

a um coeficiente do polinômio. Durante essa conversão, também são gerados o vetor r e os vetores de erro e_1 e e_2, onde cada r_i, $e_{1,i}$ e $e_{2,i}$ que pertencem respectivamente aos conjuntos de amostras $r_i \in \mathbb{R}_q^k$ de B_{η_1}, $e_1 \in \mathbb{R}_q^k$ de B_{η_2} e $e_2 \in \mathbb{R}_q$ de B_{η_2}.

Utilizando a chave pública, o texto cifrado é obtido após aplicação das funções Compress_q aos vetores u e v, onde $u := A^T r + e_1$ e $v := t^T r + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m$. Matricialmente representam-se os vetores u e v da seguinte forma:

$$u = A^T r + e_1 = \begin{pmatrix} a_{11}r_1 + a_{21}r_2 + \cdots + a_{k1}r_k \\ a_{12}r_1 + a_{22}r_2 + \cdots + a_{k2}r_k \\ \vdots \\ a_{1k}r_1 + a_{2k}r_2 + \cdots + a_{kk}r_k \end{pmatrix} + \begin{pmatrix} e_{1_1} \\ e_{1_2} \\ \vdots \\ e_{1_k} \end{pmatrix}$$

$$v = t^T r + e_2 = \left(t_1, t_2, \ldots, t_k\right) \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{pmatrix} + \begin{pmatrix} e_{2_1} \\ e_{2_2} \\ \vdots \\ e_{2_k} \end{pmatrix} + \left\lceil \frac{q}{2} \right\rfloor \cdot m$$

No processo de decifragem, o texto claro m é obtido pela operação $m = \text{Compress}_q(v - s^T u, 1)$ (2). A subtração $v - s^T u$ pode ser reescrita como $= w + \left\lceil \frac{q}{2} \right\rfloor \cdot m$, onde $\|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor$. (2) definem $m' = \text{Compress}_q(v - s^T u, 1)$, de modo que

$$\left\lceil \frac{q}{4} \right\rfloor \geq \|v - s^T u - \left\lceil \frac{q}{2} \right\rfloor \cdot m'\|_\infty = \|w + \left\lceil \frac{q}{2} \right\rfloor \cdot m - \left\lceil \frac{q}{2} \right\rfloor \cdot m'\|_\infty.$$

Pela desigualdade triangular e pelo fato de que $\|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor$, obtém-se

$$\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m - m')\|_\infty < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor,$$

o que (para todos os q ímpares) implica que $m = m'$, e comprova a corretude do Kyber.PKE (2).

Bos et al.(2) definem as funções Compress_q e Decompress_q como $\text{Compress}_q(x, d) = \left\lceil \left(\frac{2d}{q}\right) \cdot x \right\rfloor \bmod 2d$ e $\text{Decompress}_q(x, d) = \left\lceil \left(\frac{q}{2d}\right) \cdot x \right\rfloor$, são utilizadas com $x \in \mathbb{R}_q$ ou $x \in \mathbb{R}_q^k$. Essas funções descartam bits menos significativos e de baixa ordem no texto cifrado, reduzindo o tamanho do texto cifrado. A função Decompress_q também cria intervalos de tolerância a erros, onde o bit de mensagem 0 é mapeado para 0 e 1 é mapeado para $\left\lceil \frac{q}{2} \right\rfloor$. Para obter a mensagem decodificada correta, é necessário reverter a expansão causada pelos erros e_1 e e_2. Neste contexto, Compress_q é empregada na decifração do resultado da subtração $v - s^T u$, que é aproximadamente $\frac{q}{2} \cdot m$: se $v - s^T u$ está mais próximo de $\left\lceil \frac{q}{2} \right\rfloor$ do que de 0, a mensagem decodificada é 1; caso contrário, é 0. Os parâmetros do algoritmo Crystals Kyber, apresentados na Tabela 2, foram definidos para garantir uma taxa de falha de decriptografia δ inferior a 2^{-128}.

2.3.4 Saber

O Saber é um mecanismo de encapsulamento de chaves baseado no problema matemático *Learning with Rounding* (LWR) que participou do concurso de padronização

pós-quântica do NIST. Ele possui três variantes: LightSaber, Saber e FireSaber, correspondendo aos níveis de segurança 1, 3 e 5 estabelecidos pelo NIST, equivalentes à dificuldade de quebrar as versões 128, 192 e 256 do AES.

O problema *Decision*-LWE, inicialmente proposto por Regev (42), envolve a dificuldade de distinção de amostras totalmente aleatórias $(\mathbf{a}, u) \leftarrow U(\mathbb{Z}_q^{l\times 1} \times \mathbb{Z}_q)$ de amostras LWE da forma $((\mathbf{a}, b = \mathbf{a}^T\mathbf{s} + e) \in \mathbb{Z}_q^{l\times 1} \times \mathbb{Z}_q)$, onde $\mathbf{s} \leftarrow \beta\mu(\mathbb{Z}_q^{l\times 1})$ permanece constante para todas as matrizes $\mathbf{a} \leftarrow U(\mathbb{Z}_q^{l\times 1})$ e $e \leftarrow \beta\mu(\mathbb{Z}_q)$ é um pequeno vetor de erro.

O problema LWR, introduzido por Banerjee et al. (53), é uma versão determinística (não aleatória) do problema LWE. No LWR, os erros ou ruídos são gerados de forma determinística ao escalar e arredondar coeficientes de q para p $(p < q)$. Uma amostra LWR tem o formato $(\mathbf{a}, b = \left\lfloor \frac{p}{q}(\mathbf{a}^T\mathbf{s}) \right\rceil_p^m)$ para $\mathbf{s} \leftarrow \beta\mu(\mathbb{Z}_q^{l\times 1})$ fixo em uma distribuição uniforme $\mathbf{a} \leftarrow U(\mathbb{Z}_q^{l\times 1})$.

O termo escalar refere-se à multiplicação de um vetor ou matriz por um número, chamado de escalar. No contexto do LWR, isso significa multiplicar os coeficientes do vetor por p/q antes de realizar o arredondamento. Os coeficientes são escalados de acordo com p/q, onde p e q são números inteiros e $p < q$. O arredondamento é então aplicado aos coeficientes escalados para gerar o ruído ou erro na amostra LWR.

Semelhante ao *Decision*-LWE, o problema *Decision*-LWR consiste na dificuldade de distinguir amostras LWR de amostras totalmente aleatórias $(\mathbf{a}, u) \leftarrow U(\mathbb{Z}_q^{l\times 1} \times \mathbb{Z}_q)$, de modo que a segurança do protocolo reside desta dificuldade. Uma amostra Mod-LWR é dada por $(\mathbf{a}, b = \left\lfloor \frac{p}{q}(\mathbf{a^T}s) \right\rceil_p^m) \in R_q^{l\times 1} \times R_p$, onde $\mathbf{s} \leftarrow \beta\mu(R_q^{l\times 1})$ é fixo para todas as amostras e $\mathbf{a} \leftarrow U(R_q^{l\times 1})$.

O Saber emprega parâmetros inteiros positivos k, q, p, t, e μ com $n = 256$ para atender aos diversos requisitos de segurança estabelecidos pelo NIST, conforme detalhado na Tabela 3. O esquema de criptografia de chave pública Saber.PKE = (KeyGen, Enc, Dec) é delineado nos algoritmos da Figura 13 (41). No contexto deste esquema, o espaço de mensagens M é definido como $M = \{0,1\}^{256}$, onde cada mensagem $m \in M$ e as chaves de sessão encapsuladas possuem 256 bits, viabilizando a integração com algoritmos de chave simétrica (41).

Tabela 3 – Parâmetros Saber

Versão	**Nível de Segurança**	**Parâmetros**					
		n	**k**	**q**	**p**	**t**	μ
LightSaber	1	256	2	2^{13}	2^{10}	2^2	10
Saber	3	256	3	2^{13}	2^{10}	2^3	8
FireSaber	5	256	4	2^{13}	2^{10}	2^5	6

O algoritmo `Saber.PKE.KeyGen` é responsável tanto pela geração da chave pública pk quanto da chave secreta sk. Inicialmente, uma semente aleatória $seed_{\mathbf{A}}$ é criada a partir de uma distribuição aleatória de 256 bits ($\{0,1\}^{256}$). Utilizando essa semente, uma matriz

Algorithm 4 Saber.PKE.KeyGen()

Saída: Chave secreta sk
Saída: Chave pública pk

1: $seed_{\mathbf{A}} \leftarrow U(\{0,1\}^{256})$
2: $A \leftarrow \text{gen}(seed_{\mathbf{A}}) \in R_q^{l \times l}$ ▷ Gera matriz pseudoaleatória $A \in R_q^{l \times l}$ a partir de $seed_{\mathbf{A}}$
3: $r \leftarrow U(\{0,1\}^{256})$
4: $s \leftarrow \beta\mu(R_q^{l \times 1}; r)$
5: $\mathbf{b} = ((\mathbf{A}^T\mathbf{s} + \mathbf{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$
6: **retorne** $(pk := (seed_{\mathbf{A}}, \mathbf{b}), sk := \mathbf{s})$

Algorithm 5 Saber.PKE.Enc$((\text{seed}_{\mathbf{A}}, \mathbf{b}), m; r)$

Entrada: Chave pública $(\text{seedA}, b) \in B^{288}$
Entrada: Mensagem $m \in B^{32}$
Entrada opcional: Semente aleatória $r \in B^{256}$
Saída: Mensagem $c_m \in B^{32}$

1: $\mathbf{A} = \text{gen}(\text{seed}_{\mathbf{A}}) \in R_q^{l \times l}$
2: **if** r não foi especificado **then**
3: $\quad r \leftarrow U(\{0,1\}^{256})$
4: **end if**
5: $s' \leftarrow \beta\mu(R_q^{l \times 1}; r)$
6: $b' = ((\mathbf{A}\mathbf{s}' + \mathbf{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in R_p^{l \times 1}$
7: $v' = \mathbf{b}^T(\mathbf{s}' \mod p) \in R_p$
8: $\mathbf{c_m} = ((v' + h_1 - 2^{\epsilon_p - 1}m) \mod p) \gg (\epsilon_p - \epsilon_t) \in R_T$
9: **retorne** $c := (\mathbf{c_m}, \mathbf{b}')$

Algorithm 6 Saber.PKE.Dec$(s, (c_m, \mathbf{b}'))$

Entrada: Chave secreta s
Entrada: Cifra $(c_m, \mathbf{b}')$
Saída: Mensagem $m' \in B^{32}$

1: $v = \mathbf{b}'^T(\mathbf{s} \mod p) \in R_p$
2: $\mathbf{m}' = ((v + h_2 - 2^{\epsilon_p - \epsilon_t - 1}c_m) \mod p) \gg (\epsilon_p - 1) \in R_2$
3: **retorne** $\mathbf{m}'$

Figura 13 – Algoritmos Saber.PKE. Adaptado de (3)

pseudoaleatória A de dimensões $l \times l$ sobre um anel q é gerada. Uma segunda semente aleatória r de 256 bits é gerada utilizando o mesmo gerador pseudoaleatório empregado para $seed_{\mathbf{A}}$.

Utilizando a semente r, um vetor s é gerado através da aplicação das funções de arredondamento e modulação $\beta\mu(R_q^{l \times 1}; r)$. Este vetor s representa a chave secreta. Enquanto isso, a concatenação da $seed_{\mathbf{A}}$ com o vetor b $((b = A^T s + h) mod\ q)$ forma a chave pública.

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_l \end{pmatrix} \qquad \mathbf{b} = \mathbf{A}^T\mathbf{s}+\mathbf{h} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1l} \\ a_{21} & a_{22} & \dots & a_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{ll} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_l \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_l \end{pmatrix}$$

O algoritmo `Saber.PKE.Enc` cifra uma mensagem m usando a chave pública $(\text{seed}_{\mathbf{A}}, \mathbf{b})$. Inicialmente, uma matriz pseudoaleatória $\mathbf{A}$ de dimensões $l \times l$ é gerada a partir da semente $\text{seed}_{\mathbf{A}}$. Se a semente aleatória r de 256 bits não for especificada, ela é gerada aleatoriamente, e Em seguida, um vetor $\mathbf{s}'$ é criado aplicando uma distribuição binomial $\beta\mu(R_q^{l\times 1}; r)$. O vetor $\mathbf{s}'$ é usado para calcular $\mathbf{b}'$. O vetor v' é obtido calculando o produto escalar de $\mathbf{b}$ com $\mathbf{s}'$ módulo p. Finalmente, a cifra $\mathbf{c_m}$ é calculada aplicando a operação de módulo p à soma de v' com h_1 subtraído de $2^{\epsilon_p-1}m$. O texto cifrado c é obtido pela concatenação dos vetores $\mathbf{c_m}$ e $\mathbf{b}'$.

$$\mathbf{b'} = \mathbf{A} \cdot \mathbf{s}' + \mathbf{h} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1l} \\ a_{21} & a_{22} & \dots & a_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{ll} \end{pmatrix} \cdot \begin{pmatrix} s'_1 \\ s'_2 \\ \vdots \\ s'_l \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_l \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_l \end{pmatrix}$$

$$\mathbf{v'} = (\mathbf{b}^T \cdot \mathbf{s}' \text{mod p}) = \begin{pmatrix} b'_1 & b'_2 & \dots & b'_l \end{pmatrix} \cdot \begin{pmatrix} s'_1 \\ s'_2 \\ \vdots \\ s'_l \end{pmatrix} = \begin{pmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_p \end{pmatrix}$$

$$\mathbf{c}_m = \mathbf{v}' + h_1 - 2^{\epsilon_p-1} \cdot \mathbf{m} = \begin{pmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_p \end{pmatrix} + \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{1p} \end{pmatrix} - 2^{\epsilon_p-1} \cdot \mathbf{m} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{2t} \end{pmatrix}$$

$$\mathbf{c} = (\mathbf{c_m}, \mathbf{b}') = \left(\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{2t} \end{pmatrix}, \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_l \end{pmatrix} \right)$$

O algoritmo `Saber.PKE.Dec` recebe a chave secreta sk e a cifra $(c_m, \mathbf{b}')$ como entradas. Primeiro, calcula v realizando o produto escalar da transposta de $\mathbf{b}'$ com ($\mathbf{s}$ mod p). Em seguida, $\mathbf{m}'$ é computado subtraindo $2^{\epsilon_p-\epsilon_t-1}c_m$ de v, somando h_2 e aplicando a operação de módulo p. O resultado final é obtido deslocando $\epsilon_p - 1$ bits para a direita, gerando assim a mensagem decifrada $\mathbf{m}'$.

$$\mathbf{m'} = \mathbf{v} - 2^{\epsilon_p - \epsilon_t - 1} \cdot \mathbf{c_m} + \mathbf{h_2}$$

$$= \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{pmatrix} - \begin{pmatrix} 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_1} \\ 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_2} \\ \vdots \\ 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_{2t}} \end{pmatrix} + \begin{pmatrix} h_{2_1} \\ h_{2_2} \\ \vdots \\ h_{2_p} \end{pmatrix} = \begin{pmatrix} v_1 - 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_1} + h_{2_1} \\ v_2 - 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_2} + h_{2_2} \\ \vdots \\ v_p - 2^{\epsilon_p - \epsilon_t - 1} \cdot c_{m_p} + h_{2_p} \end{pmatrix}$$

Cada versão do algoritmo Saber tem sua própria probabilidade de falha/erro de decriptação: LightSaber (2^{-120}), Saber (2^{-136}) e FireSaber (2^{-165}). Além dos parâmetros listados na Tabela 3, são definidas três constantes: o vetor $h \in \mathbb{R}^{l \times 1}$ com coeficientes $2^{\epsilon_q - \epsilon_p - 1}$, o polinômio $h_1 \in \mathbb{R}^q$ com coeficientes $2^{\epsilon_q - \epsilon_p - 1}$ e o polinômio $h_2 \in \mathbb{R}^q$ com coeficientes $(2^{\epsilon_p - 2} - 2^{\epsilon_p - \epsilon_t - 2})$, onde $\epsilon_q = 13$, $\epsilon_p = 10$ e $\epsilon_t = 3$. Essas constantes são usadas para realizar operações de arredondamento, reduzindo a probabilidade de falhas (41).

2.3.5 Frodo

O Frodo é um mecanismo de encapsulamento de chaves pós-quântico baseado no problema LWE, que também participou do concurso seletivo do NIST. Utilizando os parâmetros listados na Tabela 4, as versões Frodo640, Frodo976 e Frodo1344 atendem respectivamente aos níveis de segurança 1, 3 e 5. Dependendo da versão selecionada, o Frodo gera textos cifrados ou chaves de sessão cifradas de 128, 192 e 256 bits (39).

De acordo com o Instituto *Bundesamt für Sicherheit in der Informationstechnik* (BSI) (54), o Frodo possui segurança contra ataques de computadores quânticos. Embora não seja um finalista devido à sua baixa eficiência computacional em comparação com outros candidatos, sua segurança não é questionada. Isso se deve ao fato de que o projeto do Frodo prioriza a segurança em detrimento da velocidade (55).

Conforme Jr et al.(4), apresenta-se uma descrição do FrodoPKE nos algoritmos 7, 8 e 9, omitindo detalhes sobre a geração (pseudo)aleatória usando primitivas simétricas e amostragem da distribuição de erro χ.

O algoritmo `FrodoPKE.Keygen()` inicia gerando aleatoriamente as matrizes A, S e E, e em seguida calcula B usando a fórmula $B = (A \cdot S + E) \mod q$. As chaves pública pk e secreta sk são geradas e retornadas como $\text{pk} = (\mathbf{A}, \mathbf{B})$ e $\text{sk} = \mathbf{S}$.

No algoritmo `FrodoPKE.Enc()`, a chave pública $\text{pk} = (A, B)$ e uma mensagem μ são inseridas como entrada. O algoritmo gera vetores aleatórios S' e E' utilizando a distribuição χ, calcula $B' = S'A + E'$ e V, e, em seguida, produz o texto cifrado $c = (C_1, C_2)$, onde $C_1 = B'$ e $C_2 = V + \text{Encode}(\mu)$.

Tabela 4 – Parâmetros Frodo

Versão	**Nível de Segurança**	**Parâmetros**				
		D	**q**	**n**	m	**B**
Frodo640	1	15	32768	640	8	2
Frodo976	3	15	65536	976	8	3
Frodo1344	5	16	65536	1344	8	4

Por fim, no `FrodoPKE.Dec()`, utilizando a chave secreta sk e a cifra $c = (C_1, C_2)$, o algoritmo subtrai $C_1 S$ de C_2 para obter M, o qual é então decodificado para recuperar o texto claro μ.

Algorithm 7 FrodoPKE.Keygen()

Saída: Chave pública $pk = (\mathbf{A}, \mathbf{B})$
Saída: Chave secreta $sk = \mathbf{S}$

1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}$, $\mathbf{S} \leftarrow \chi^{n \times n}$, $\mathbf{E} \leftarrow \chi^{n \times n}$
2: $\mathbf{B} \leftarrow (\mathbf{A} \cdot \mathbf{S} + \mathbf{E}) \mod q$
3: **retorne** $(pk = (\mathbf{A}, \mathbf{B}), sk = \mathbf{S})$

Algorithm 8 FrodoPKE.Enc(pk, μ)

Entrada: Chave pública $pk = (\mathbf{A}, \mathbf{B})$
Entrada: Mensagem μ
Saída: Cifra $c = (\mathbf{C}_1, \mathbf{C}_2)$

1: $(\mathbf{A}, \mathbf{B}) \leftarrow pk$
2: $\mathbf{S}', \mathbf{E}' \leftarrow \chi^{m \times n}$
3: $\mathbf{E}'' \leftarrow \chi^{m \times n'}$
4: $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$
5: $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
6: $(\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + \text{Encode}(\mu))$
7: **retorne** $c = (\mathbf{C}_1, \mathbf{C}_2)$

Algorithm 9 FrodoPKE.Dec(sk, c)

Entrada: Chave secreta sk
Entrada: Cifra $c = (\mathbf{C}_1, \mathbf{C}_2)$
Saída: Mensagem μ

1: $\mathbf{S} \leftarrow sk$
2: $(\mathbf{C}_1, \mathbf{C}_2) \leftarrow c$
3: $\mathbf{M} \leftarrow \mathbf{C}_2 - \mathbf{C}_1\mathbf{S}$
4: **retorne** $\mu = \text{Decode}(\mathbf{M})$

Figura 14 – Algoritmos Frodo.PKE. Adaptado de (4)

Para complementar a descrição do FrodoPKE, detalham-se as funções Encode e Decode. A função Encode é responsável por codificar um inteiro k no intervalo $0 \leq k < 2^B \leq q$ como um elemento em $\mathbb{Z}_q$ usando a seguinte fórmula:

$$\text{Encode}(k) = k \cdot q/2^B$$

A função Decode extrai um inteiro de B bits a partir de um elemento de $\mathbb{Z}_q$ usando a seguinte fórmula:

$$\text{Decode}(k) = \left\lfloor c \cdot 2^B/q \right\rceil \cdot \mod 2^B$$

As funções Encode e Decode possuem uma propriedade de correção de erro descrita em (39): seja $q = 2^D$ e $\bar{B} \leq D$. Então, para quaisquer k e e em $\mathbb{Z}$ tal que $0 \leq k < 2^B$ e $-q/2^{B+1} \leq e < q/2^{B+1}$, Decode(Encode($k$) + e) = k.

A correção no FrodoPKE ocorre porque a decriptação é obtida por $M = C_2 - C_1 S = \text{Encode}(\mu) + S'E - E'S + E''$. Se as entradas de E'' forem suficientemente pequenas, então Decode(M) = μ, entretanto, se a entrada no erro de decodificação E'' exceder o limiar de decodificação $[-q/2^{B+1}, q/2^{B+1})$, ocorrerá uma falha na decodificação na entrada correspondente de μ.

2.3.6 NTRU

O NTRU é um KEM que se baseia no problema *Ring Learning with Errors* (RLWE). Embora tenha participado no processo seletivo do NIST, já era conhecido antes mesmo do concurso de padronização em 2016, como documentado por Hoffstein et al. em 1996 (40).

A versão submetida ao NIST combina os algoritmos NTRUEncrypt e NTRU-HRSS-KEM (56). O NTRUEncrypt é utilizado para criptografia de chave pública, enquanto o NTRU-HRSS-KEM é responsável pela geração e encapsulamento de uma chave de sessão de 32 bytes. Ambos os algoritmos possuem um custo computacional mais elevado, especialmente durante a geração de chaves, se comparados aos finalistas Crystals Kyber e Saber. Ressalta-se que o NTRU é um algoritmo mais antigo entre os finalistas, o que sugere um alto nível de segurança (55).

O NTRUEncrypt (40) requer a escolha dos parâmetros N, p, e q. N é um número primo que determina o grau do polinômio truncado, q e p devem ser primos entre si, com q substancialmente maior do que p. De acordo com (5), os valores sugeridos para esses parâmetros encontram-se listados na Tabela 5.

Tabela 5 – Parâmetros NTRU

Nível de segurança	**Parâmetros**		
	N	**p**	**q**
Moderada	167	3	128
Padrão	251	3	128
Alta	347	3	128
Máxima	503	3	256

Algorithm 10 NTRUEncrypt.Keygen()

Entradas: Parâmetros q, p
Saídas: Chave pública $h(x)$, Chave privada $f(x)$ e $g(x)$

1: $f(x) \leftarrow$ Escolha aleatória de um polinômio que seja invertível em R_q e R_p
2: $g(x) \leftarrow$ Escolha aleatória de um 'pequeno' polinômio
3: $f_q(x) \leftarrow$ Inverso do polinômio $f(x)\ mod\ q$
4: $f_p(x) \leftarrow$ Inverso do polinômio $f(x)\ mod\ p$
5: $h(x) = p \cdot f_q(x) \cdot g(x)\ mod\ q$
6: **Retorne** $(pk = h(x), sk = (f(x), f_p(x)))$

Algorithm 11 NTRUEncrypt.Enc()

Entradas: Texto claro $m(x)$, Chave pública pk, Parâmetro q
Saída: Texto cifrado $e(x)$

1: $m(x) \leftarrow$ Codificação do texto claro em um polinômio ternário
2: $r(x) \leftarrow$ Escolha aleatória de um 'pequeno' polinômio
3: $e(x) = h(x) \times r(x) + m(x)\ mod\ q$
4: **Retorne** Texto cifrado $e(x)$

Algorithm 12 NTRUEncrypt.Dec()

Entradas: Texto cifrado $e(x)$, Chave secreta sk, Parâmetros q, p
Saídas: Texto claro $m(x)$

1: $a(x) = f(x) \times e(x)\ mod\ q$
2: $m(x) = f_p(x) \times a(x)\ mod\ p$
3: **Retorne** Texto claro $m(x)$

Figura 15 – Algoritmos NTRUEncrypt. Adaptado de (5)

O parâmetro q é tipicamente escolhido na forma de 2^n devido às suas vantagens computacionais. Como q e p devem ser primos entre si, 3 é geralmente o menor valor escolhido para p. O parâmetro N é responsável por determinar o nível de segurança do sistema. É importante destacar que um pequeno polinômio se refere a um polinômio ternário, que é consideravelmente menor em comparação aos polinômios em R_q (5).

O processo de geração de chaves no NTRUEncrypt envolve a escolha aleatória dos polinômios $f(x)$ e $g(x)$. $f(x)$ deve ser invertível tanto em R_q quanto em R_p. São calculados os inversos $f_q(x)$ e $f_p(x)$ de $f(x)$ módulo q e p respectivamente. A chave pública $h(x)$ é obtida através de $h(x) = p \cdot f_q(x) \cdot g(x) \mod q$. A chave privada consiste no par $(f(x), f_p(x))$.

Para a encriptação, o texto claro $m(x)$ é codificado como um polinômio ternário (-1,0,1) e um polinômio "pequeno"$r(x)$ é escolhido aleatoriamente. O texto cifrado $e(x)$ é calculado por $e(x) = h(x) \times r(x) + m(x) \mod q$.

No processo de decifração, $a(x)$ é calculado multiplicando $f(x)$ por $e(x)$ módulo q, e $m(x)$ é obtido multiplicando $f_p(x)$ por $a(x)$ módulo p. O texto claro $m(x)$ é finalmente retornado como saída. Importante notar que se $f(x)$ é expresso como $f(x) = 1 + pF(x)$ e $f_p(x) = 1 \mod p$, a segunda etapa da decifração pode ser simplificada para $m(x) = a(x) \mod p$.

2.4 Geradores de Números Aleatórios e Pseudoaleatórios

Em aplicações criptográficas, sequências binárias aleatórias ou pseudoaleatórias são empregadas para gerar sementes ou chaves de criptografia (57). Essas sequências podem ser produzidas tanto por Geradores de Números Aleatórios (*Random Number Generators*, RNG), quanto por Geradores de Números Pseudoaleatórios (*Pseudo-Random Number Generators*, PRNG).

2.4.1 Geradores de Números Aleatórios

Os Geradores de Números Aleatórios (RNG) utilizam fontes de determinismo difícil e funções de processamento para criar aleatoriedade (30, 57). As saídas do RNG podem ser usadas diretamente como números aleatórios ou como entrada para geradores de números pseudoaleatórios (16).

Fontes de determinismo difícil produzem números aleatórios de forma imprevisível e não repetitiva, e podem se basear em eventos físicos ou processos naturais intrinsecamente aleatórios, como ruído térmico, decaimento radioativo e amostras de som em ambientes ruidosos (58).

De acordo com Gutmann(59), a maioria dos usuários não tem acesso a hardware especializado para análise de fontes aleatórias, o que leva à necessidade de métodos alternativos para obter dados aleatórios. Alternativas incluem a medição do tempo de turbulência do ar nas cabeças de leitura de um disco rígido (60) e o tempo de pressionamento de teclas durante a inserção de senhas (61, 59).

2.4.2 Geradores de Números Pseudoaleatórios

Geradores de Números Pseudoaleatórios (PRNGs) usam uma ou mais sementes para produzir números pseudoaleatórios (57). A imprevisibilidade da semente é crucial em contextos que exigem aleatoriedade, muitas vezes obtida de um RNG (30). As saídas do PRNG são determinísticas, derivadas da semente, com toda a aleatoriedade restrita à semente. Sequências pseudorrandômicas, quando construídas com transformações que eliminam correlações estatísticas, podem parecer mais aleatórias do que aquelas de fontes físicas. Portanto, as saídas do PRNG, por serem reproduzíveis a partir da semente, podem

ter melhores propriedades estatísticas e ser geradas mais rapidamente do que as de um RNG (30).

Entre os PRNGs mais comuns, estão os Geradores Congruentes Lineares (62), Geradores de Atraso de Fibonacci, Geradores de Registradores de Deslocamento, Geradores Híbridos (63) e a biblioteca OpenSSL (64, 65).

2.4.3 Estatística *P-value*

Ao testar a aleatoriedade de uma sequência, são formuladas duas hipóteses: a nula (H0) conclui que a sequência é aleatoriedade, enquanto a alternativa (H1 ou Ha) considera que a sequência analisada é não aleatória (30).

No teste de hipótese, assume-se inicialmente que a hipótese nula (H0), que indica a aleatoriedade da sequência, é verdadeira, e, posteriormente, procuram-se evidências para rejeitá-la. O teste é realizado com base em uma amostra, sem a necessidade de analisar toda a população. Esse processo está sujeito a dois tipos de erros: tipo I (falso positivo) e tipo II (falso negativo) (30). A Tabela 6 resume os resultados do teste de hipótese.

Tabela 6 – Resultados do teste de hipótese

Situação	**Conclusão**	
	Aceita H0	Rejeita H0
H0 é verdadeiro	Sem erro	Erro tipo I
H1 é verdadeiro	Erro tipo II	Sem erro

O erro tipo I refere-se à rejeição errônea da hipótese nula (H0) e está relacionado ao *P-value*. Um *P-value* < 0.001 indica que, em média, 1 em 1000 H0 será rejeitada erroneamente, aceitando a hipótese alternativa com 99,99% de confiança. Se *P-value* < 0.01, 1 em 100 H0 será rejeitada, indicando não aleatoriedade da sequência com 99,9% de confiança; caso contrário, a sequência é considerada aleatória (30).

Um *P-value* de 1 indica aleatoriedade perfeita, enquanto 0 indica completa não aleatoriedade. O nível de significância α é escolhido para o teste. Se $P \geq \alpha$, aceita-se a hipótese nula, indicando aleatoriedade da sequência. Se $P < \alpha$, rejeita-se a hipótese nula, indicando não aleatoriedade. O parâmetro α representa a probabilidade do erro tipo I, geralmente escolhido no intervalo [0.001, 0.01] (30).

O segundo erro ocorre quando H0 é aceita, mas é falsa, representado pelo valor β. Diferentemente de α, β não pode ser precisamente determinado, pois, ao assumir que H0 é falsa, o parâmetro a ser testado pode assumir infinitos valores, resultando em diferentes valores para β (30).

2.4.4 Bateria de testes estatísticos do NIST

O NIST desenvolveu a *NIST Statistical Test Suite* (NIST STS), uma bateria composta por 15 testes estatísticos desenvolvidos para avaliar o nível de aleatoriedade de sequências binárias geradas por Geradores de Números Aleatórios (RNGs) e Geradores de Números Pseudoaleatórios (PRNGs) em contextos criptográficos. É importante ressaltar que a aprovação nos testes estatísticos não garante a eficácia do sistema criptográfico contra ataques criptoanalíticos, e não existe um teste ou conjunto de testes que possa confirmar absolutamente a aleatoriedade de uma sequência. Além disso, é esperado que haja algumas falhas em sequências pseudoaleatórias analisadas (30).

Na metodologia NIST STS, cada teste estatístico possui uma abordagem específica com seus próprios cálculos matemáticos e estatísticos para determinar o valor de referência conhecido como *P-value*. Em todos os 15 testes, uma regra de decisão é aplicada a um nível de significância de 1%. Se o *P-value* for ≥ 0.01, a sequência analisada é considerada aleatória. A Tabela 7 oferece uma síntese dos testes da NIST STS, conforme descrito em (30). A Tabela 8 apresenta o tamanho mínimo da entrada, a quantidade de *P-value(s)* e a regra de decisão correspondente a cada teste.

2.5 Aprendizado de máquina e reconhecimento de padrões em criptogramas

O Aprendizado de Máquina (AM) é uma área da Inteligência Artificial (IA) dedicada ao desenvolvimento de sistemas computacionais que adquirirem conhecimento automaticamente (66). Esses sistemas tomam decisões baseadas em experiências passadas, ou seja, em casos ou exemplos anteriores, melhorando seu desempenho gradativamente ao resolver problemas anteriores (67).

Mesmo com o AM sendo uma excelente ferramenta para aquisição automática de conhecimento, é importante ressaltar que não existe um único algoritmo que apresente o melhor desempenho para a resolução de todos os problemas de AM. É essencial compreender tanto o potencial quanto as limitações dos diversos algoritmos e estabelecer metodologias para avaliar sua eficácia em problemas específicos (6).

A indução é uma forma de inferência lógica que generaliza conceitos de exemplos específicos, indo da parte para o todo. Embora as hipóteses geradas pela indução nem sempre reflitam a verdade absoluta, esse método é fundamental para adquirir conhecimento e prever eventos futuros. Sua eficácia depende da escolha cuidadosa dos exemplos, de modo que uma pequena quantidade de exemplos ou exemplos mal escolhidos podem levar a conclusões pouco confiáveis.

No aprendizado supervisionado, o algoritmo utiliza um conjunto de exemplos de

Tabela 7 – Descrição dos Testes da NIST STS

Teste	**Objetivo**
Teste de frequência (Monobit)	Determina se o número de uns e zeros em uma sequência é aproximadamente o esperado para uma sequência verdadeiramente aleatória.
Teste de frequência dentro de blocos	Verifica se a frequência de m-bit blocos em uma sequência é como seria esperado para uma sequência verdadeiramente aleatória.
Teste de corrida	Determina se o número de corridas de uns e zeros de vários comprimentos é o esperado para uma sequência aleatória, indicando oscilação entre substrings.
Teste para a mais longa corrida de 1's em um bloco	Analisa se a distribuição de longas corridas de 1's coincide com as probabilidades teóricas.
Teste do posto para matrizes binárias	Verifica se a distribuição da classificação de matrizes de 32x32 bits coincide com as probabilidades teóricas.
Teste da transformação discreta de Fourier (Espectral)	Analisa se a frequência espectral da sequência binária é como seria esperado para uma sequência verdadeiramente aleatória.
Teste de não sobreposição de padrão	Verifica se o número de ocorrências de um modelo não periódico específico coincide com o esperado para uma sequência verdadeiramente aleatória.
Teste de sobreposição de padrão	Analisa se o número de ocorrências para todo modelo de 1's coincide com o esperado para uma sequência verdadeiramente aleatória.
Teste estatístico Universal de Maurer	Determina se uma sequência binária comprime além do esperado para uma sequência verdadeiramente aleatória.
Teste de complexidade linear	Verifica se a sequência é complexa o suficiente para ser considerada verdadeiramente aleatória.
Teste serial	Analisa se o número de ocorrências de padrões sobrepostos de tamanho 2^m m-bits é aproximadamente o esperado para uma sequência verdadeiramente aleatória.
Teste de entropia aproximada	Compara a frequência de sobreposição de blocos de comprimento 2 consecutivos/adjacentes (m e $m+1$) com o esperado para uma sequência normalmente distribuída, determinando se a sequência parece mais regular do que o esperado de uma sequência verdadeiramente aleatória.
Teste das somas cumulativas (Cusums)	Determina se o máximo de somas acumuladas em uma sequência é muito grande ou muito pequeno, indicando muitos 1's ou 0's no início das primeiras (últimas) etapas.
Teste de excursões aleatórias	Verifica o número de ciclos dentro de uma sequência e determina se o número de visitas a um determinado estado, $[-4,-1]$ e $[1,4]$, excede o esperado para uma sequência verdadeiramente aleatória.
Teste variante de excursões aleatórias	Determina se o número total de visitas aos estados, entre $[-9,-1]$ e $[1,9]$, excede o esperado para uma sequência verdadeiramente aleatória.

Tabela 8 – Síntese NIST STS

Teste	**Tamanho mínimo da entrada**	**Qtd** $P - value(s)$	**Regra de decisão**
Frequency (Monobit)	10^2 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Frequency within a Block	10^2 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Runs	10^2 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Longest Run of Ones in a Block	8, 128 e 10^4 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Binary Matrix Rank	38.912 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Discrete Fourier Transform (Spectral)	10^3 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Non-overlapping Template Matching	10^6 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Overlapping Template Matching	10^6 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Maurer's "Universal Statistical"	387.840 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Linear Complexity	10^6 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Serial	10^6 bits	2	Se P-$value \geq 0.01$, sequência considerada aleatória
Approximate Entropy	10^2 bits	1	Se P-$value \geq 0.01$, sequência considerada aleatória
Cumulative Sums (Cusum)	10^2 bits	2	Se P-$value \geq 0.01$, sequência considerada aleatória
Random Excursions	10^6 bits	8	Se P-$value \geq 0.01$, sequência considerada aleatória
Random Excursions Variant	10^6 bits	18	Se P-$value \geq 0.01$, sequência considerada aleatória

treinamento com rótulos de classe conhecidos para criar uma hipótese que pode prever os rótulos de novos exemplos. O conjunto de treinamento $E = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ é empregado, em que cada exemplo é caracterizado por um vetor de atributos x e um rótulo de classe y. Para rótulos discretos, é chamado de classificação; para valores contínuos, é chamado de regressão (68). Por outro lado, no aprendizado não supervisionado, os algoritmos analisam exemplos e procuram agrupá-los em *clusters* (69). Após a formação dos *clusters*, uma análise adicional é necessária para compreender o significado de cada agrupamento no contexto do problema em questão (6).

Na Figura 16, ilustra-se a hierarquia de aprendizado, destacando o aprendizado supervisionado utilizando classificação, foco desta dissertação.

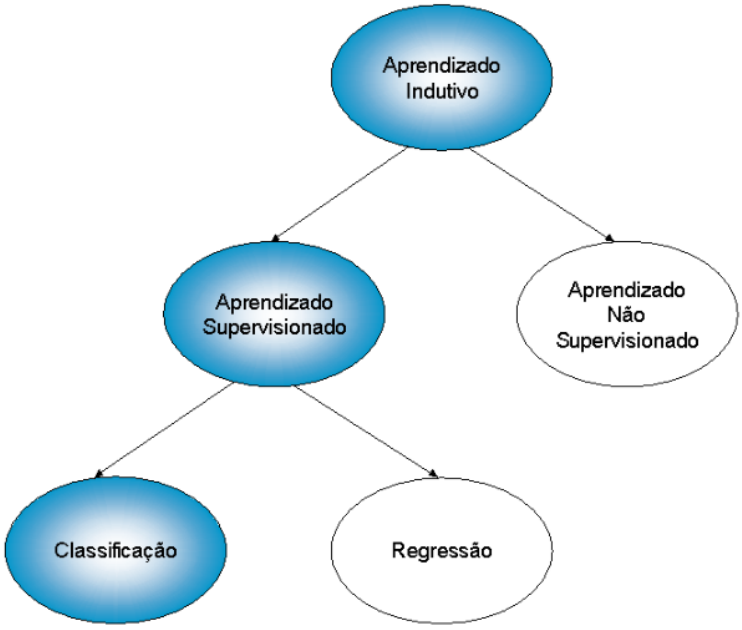


Figura 16 – Hierarquia do aprendizado. Fonte: (6)

No contexto do reconhecimento de padrões em criptogramas, algoritmos de aprendizado supervisionado podem ser empregados para classificar criptogramas com base em características extraídas, permitindo a identificação do algoritmo criptográfico usado (70).

Com base nos conteúdos apresentados no Capítulo 3, realizou-se uma avaliação de algoritmos de AM para identificar o mais eficaz em ataques de distinção em criptogramas gerados por esquemas de encriptação de chave pública pós-quânticos e nas chaves de sessão encapsuladas pelos KEM mencionados. Foram selecionados os algoritmos *K-Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), *Random Forest* (RF) e *Naive Bayes* (NB), amplamente utilizados nos trabalhos relacionados, conforme indicado na Tabela **??**, devido ao fato de que estes algoritmos possuem abordagens e características distintas, permitindo uma análise abrangente dos vetores representativos provenientes dos criptogramas e chaves de sessão gerados pelos sistemas criptográficos mencionados.

2.5.1 K-Nearest Neighbor

O KNN, apresentado na pesquisa de Aha, Kibler e Albert (71), é um algoritmo de aprendizado supervisionado do tipo *lazy* amplamente utilizado em problemas de classificação. Sua filosofia baseia-se em classificar um exemplo utilizando exemplos similares cujas classes são conhecidas. Este tipo de sistema, conhecido como *lazy*, requer que os exemplos de treinamento sejam mantidos em memória para classificar novos casos, diferentemente dos sistemas *eager*, que descartam os exemplos após induzir o modelo.

A ideia geral do KNN consiste em encontrar os k exemplos rotulados mais próximos do exemplo não classificado. Com base nos rótulos desses exemplos mais próximos, é tomada a decisão sobre a classe do exemplo não rotulado. O processo classificatório do KNN requer pouco esforço, embora o custo computacional para rotular um novo exemplo possa ser relativamente alto, especialmente se for necessário compará-lo com todos os exemplos de treinamento.

A Figura 17 ilustra essa ideia usando um conjunto de exemplos de treinamento descritos por duas classes, com exemplos positivos (+) e negativos (-). Se o KNN for utilizado com $k = 1$, o novo exemplo E_i será classificado de acordo com o único vizinho mais próximo, que é da classe positiva (+).

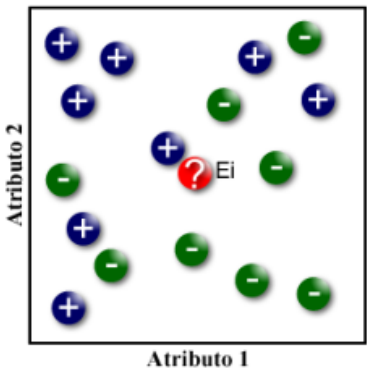


Figura 17 – Exemplo de classificação KNN

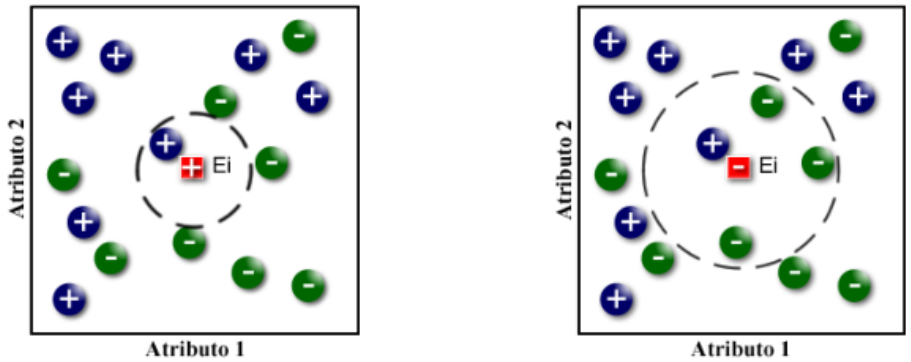
Para uma execução eficaz do KNN, é importante escolher exemplos de treinamento representativos, definir uma métrica de similaridade apropriada e determinar o número de vizinhos mais próximos. Estas escolhas impactam diretamente na precisão das previsões do algoritmo.

A eficiência do KNN está diretamente ligada à quantidade de exemplos de treinamento. Comparar um exemplo com todos os exemplos armazenados pode ser lento, especialmente em grandes conjuntos de dados. Portanto, armazenar apenas os exemplos mais representativos de cada classe otimiza o processo, reduzindo o esforço computacional da classificação e economizando espaço de memória (71).

O KNN suporta diversas medidas de distância, como a Euclidiana, a Manhattan, a

Minkowski e a Hamming, cruciais para determinar os vizinhos mais próximos e realizar classificação ou predição. O algoritmo classifica exemplos baseando-se nas classes dos k vizinhos mais próximos. Se $k = 1$, o exemplo é classificado com a mesma classe do vizinho mais próximo. Se $k > 1$, a classe majoritária dos k vizinhos mais próximos é atribuída ao exemplo para a classificação.

Na Figura 18, são ilustrados ambos os casos na classificação do exemplo E_i, usando um conjunto de exemplos positivos (+) e negativos (-) descritos por dois atributos.



(a) Exemplo de $1NN$ (b) Exemplo de $4NN$

Figura 18 – Exemplos de aplicação do algoritmo kNN, com $k = 1$ e $k = 4$

É importante observar que não existe um valor único de k que seja apropriado para a solução de todos os problemas, de modo que o valor k deve ser avaliado para cada problema em particular. Utilizar valores ímpares de k evita empates na votação da classe majoritária dos k vizinhos mais próximos. Além disso, é possível atribuir pesos aos vizinhos mais próximos com base na medida de similaridade. Se pesos são atribuídos, os k vizinhos mais próximos são ordenados por similaridade crescente com o exemplo a ser classificado. As classes dos exemplos mais similares têm maior peso na decisão da classificação em relação às classes dos exemplos menos similares.

Nos experimentos realizados nesta dissertação foram testadas diversas configurações de hiperparâmetros do classificador `KNeighborsClassifier`. Entre eles, os hiperparâmetros a seguir obtiveram melhor desempenho conjunto e encontram-se listados a seguir:

- **`n_neighbors`**: Quantidade de vizinhos a serem considerados (selecionado $k = 5$);
- **`weights`**: Peso dos vizinhos. Pode ser 'uniform' (todos os vizinhos têm o mesmo peso) ou 'distance' (os vizinhos mais próximos têm mais peso do que os mais distantes) (selecionada função 'uniform');
- **`algorithm`**: Algoritmo utilizado para computar os vizinhos mais próximos. Pode ser 'auto', 'ball_tree', 'kd_tree' ou 'brute' (selecionado 'auto');

- `p`: Parâmetro de potência para a métrica de Minkowski. Quando `p=1`, usa-se a distância de Manhattan (L1), enquanto `p=2` usa a distância euclidiana (L2). Pode ser qualquer número real positivo (selecionado valor `p=2`);
- **`metric`**: Métrica de distância usada para calcular a distância entre pontos (selecionada métrica de distância 'euclidiana');
- **`metric_params`**: Parâmetros adicionais para a métrica de distância (selecionado valor padrão=*None*).

2.5.2 Support Vector Machines

O SVM, ou Máquina de Vetores de Suporte, é uma técnica amplamente empregada em aprendizado supervisionado, útil tanto para problemas de classificação quanto de regressão. Sua principal ideia é encontrar um hiperplano ideal que separe as diferentes classes do problema de maneira ótima. Esse hiperplano pode lidar com variáveis lineares e não lineares, transformando os dados de treinamento em um espaço multidimensional, como ilustrado na Figura 19 (72).

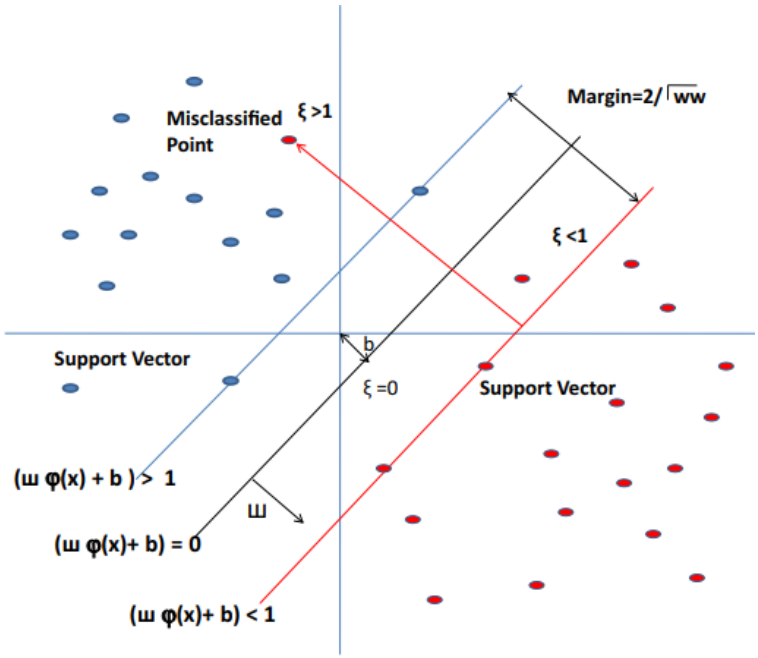


Figura 19 – Problema separável por SVM em espaço bidimensional. Fonte: (7)

O hiperplano é uma fronteira otimizada que minimiza os erros de classificação, sendo construído como um plano de decisão que distingue as diferentes classes, como mostrado na Figura 20. O SVM identifica pontos cruciais chamados vetores de suporte, que definem a fronteira de decisão e proporcionam a maior margem de separação entre as classes (73). O algoritmo separa as classes com a maior distância marginal no plano de decisão (73, 74, 75).

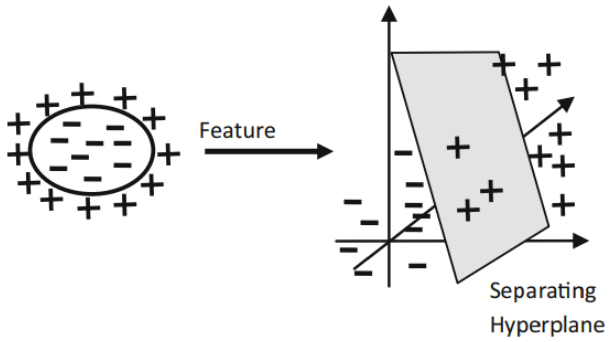


Figura 20 – Espaço de Decisão. Fonte: (7)

Conforme ilustrado na Figura 21, a separação ótima entre as classes no SVM é alcançada por meio de um hiperplano L. Este hiperplano é posicionado de forma a maximizar a margem, que representa a distância entre as bordas L_1 e L_2, permanecendo o mais próximo possível dos pontos mais próximos de cada classe. A margem pode ser definida como a distância entre o hiperplano de separação (limite de decisão) e as amostras de treinamento mais próximas a este hiperplano, conhecidas como vetores de suporte.

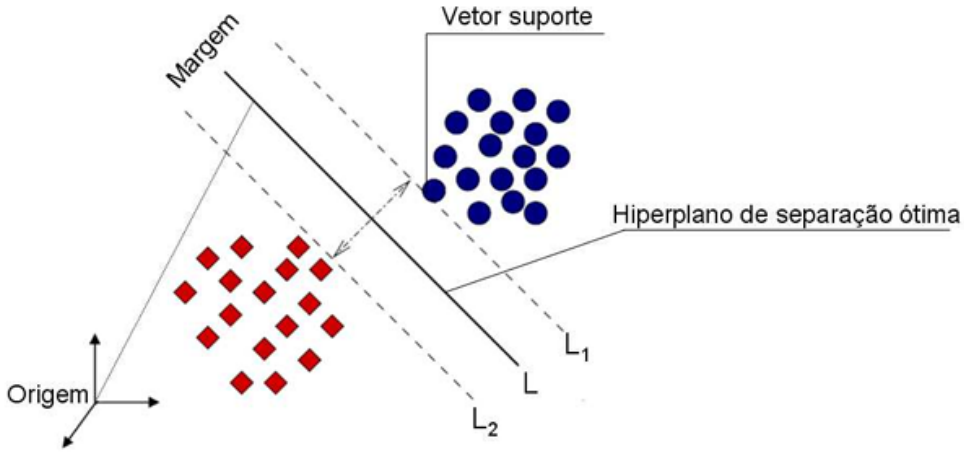


Figura 21 – Esquema de classificação SVM. Fonte: (8, 9)

A utilização de *kernels* amplia a aplicabilidade do SVM para uma variedade de problemas complexos e não-lineares. *Kernels* são funções matemáticas que transformam os dados de entrada em um espaço multidimensional, onde padrões não-lineares podem ser separados de forma mais eficaz (75). Como observado por Chang e Lin (76), *kernels* possibilitam que o SVM realize transformações não-lineares implicitamente, eliminando a necessidade de um mapeamento explícito dos dados, o que seria computacionalmente caro. Schölkopf et al. (77) destacaram que *kernels* são essenciais quando os dados não podem ser separados linearmente no espaço original, permitindo assim encontrar um hiperplano de separação eficaz em um espaço transformado. A vantagem da função de kernel é sua capacidade de simplificar o problema, dependendo apenas da dimensionalidade do espaço de entrada, não do espaço de características.

Nos experimentos realizados nesta pesquisa, várias combinações desses hiperparâmetros foram testadas. Entre elas, as configurações que se destacaram globalmente foram as seguintes:

- **C** (*Regularization parameter*): Controla a suavidade do limite de decisão. Um valor menor torna o SVM mais tolerante a erros, enquanto um valor maior busca minimizar ao máximo os erros (selecionado valor padrão= 2);
- **kernel** (*Kernel function*): Especifica o tipo de função de kernel a ser utilizada (selecionado Kernel (*Radial basis function kernel*)

  Neste kernel, os dados de entrada são mapeados para um espaço de alta dimensionalidade usando uma função de base radial, também conhecida como função gaussiana. A fórmula da função de base radial é dada por:

  $$K(x, y) = \exp\left(-\gamma \cdot \|x - y\|^2\right)$$

  onde x e y representam dois pontos no espaço de entrada ou dois conjuntos de características de diferentes exemplos, $\|x - y\|$ é a distância euclidiana entre esses pontos e γ (um parâmetro positivo) controla a largura da função de base radial. Quanto menor o valor de γ, mais ampla é a função e vice-versa.
- **gamma** γ (*Kernel coefficient*): Influencia a suavidade da fronteira de decisão. Um valor maior produz uma fronteira de decisão mais complexa, enquanto um valor menor gera uma fronteira mais suave (selecionado o valor padrão 'scale', que é calculado como $1/(\text{n\_features} \times X.\text{var}())$);
- **probability** (*Probability estimates*): Indica se o modelo deve calcular probabilidades para as classes (selecionado valor `False`);
- **tol** (*Tolerance for stopping criterion*): Tolerância para parar o processo de ajuste (selecionado valor 1×10^{-3});
- **class_weight** (*Weights associated with classes*): Define os pesos das classes (selecionado valor `None`).

2.5.3 Random Forest

O RF é um algoritmo de aprendizado supervisionado amplamente utilizado em reconhecimento de padrões e problemas de classificação com alta dimensionalidade e viés (78, 79). Sua abordagem se baseia em *ensemble learning*, combinando várias árvores de decisão, cada uma construída a partir de subconjuntos aleatórios de atributos (*features*) de exemplos rotulados. Isso cria um modelo robusto e estável, reduzindo a variabilidade

inerente aos classificadores baseados em árvores. Esta característica o torna eficaz em grandes conjuntos de dados (80).

Cada árvore individual em uma *Random Forest* contribui para a classificação de novos exemplos. Durante sua criação, o conjunto de exemplos é dividido em subconjuntos e a árvore é construída com atributos escolhidos aleatoriamente. Ao classificar uma nova amostra, as folhas de cada árvore são consideradas, e a decisão final é baseada na maioria das respostas das árvores, evitando decisões equivocadas (78).

Após a criação dos conjuntos de árvores a partir dos exemplos rotulados, o modelo classifica novas amostras. Cada subconjunto emite um voto ponderado sobre a classe do atributo-chave, e a decisão final é determinada pela classificação com o maior número de votos entre todas as árvores, considerando a similaridade e precisão de cada uma (80).

A utilização do RF traz vantagens significativas. É mais eficaz que uma única árvore de decisão, mantendo altas taxas de acerto em diferentes conjuntos de dados e evitando o sobreajuste (*overfitting*). Além disso, é menos sensível a ruídos e erros nos dados, oferecendo uma classificação estável mesmo diante de imperfeições nos dados. Sua capacidade de classificação automatizada dispensa ajustes manuais (81).

A Figura 22 ilustra o método de classificação RF, onde diversas árvores aleatórias são geradas a partir de um conjunto de exemplos rotulados X. Cada árvore tem sua própria regra de decisão, resultando em várias regras relevantes para a tomada de decisão. O conjunto de regras mais preciso classifica novas amostras, gerando uma classificação representada por Y (10).

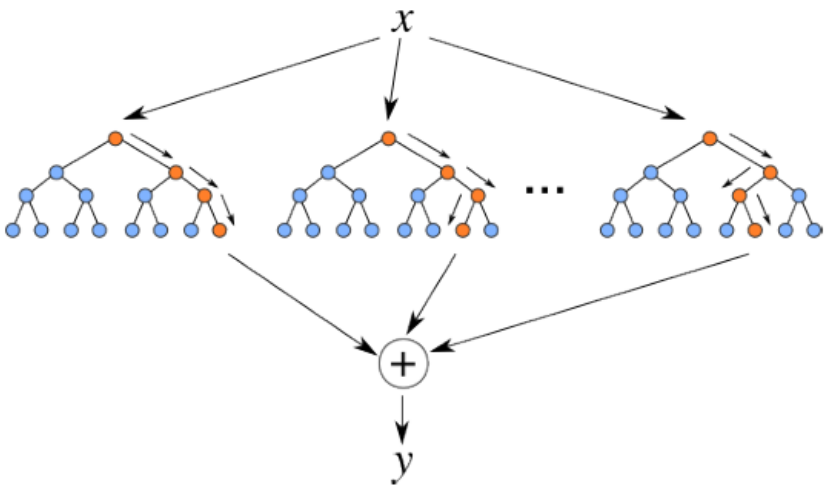


Figura 22 – Método de classificação RF (10).

O RF utiliza o coeficiente de Gini para avaliar a heterogeneidade do conjunto de exemplos e determinar a ramificação mais provável (82). O índice de Gini em um nó é calculado por:

$$Gini(S) = 1 - \sum_{i=1}^{c}(p_i)^2$$

onde p_i representa a frequência relativa da classe i no conjunto de dados e c é o número de classes.

Semelhante aos demais classificadores, várias configurações de hiperparâmetros foram testadas a fim de otimizar o desempenho do algoritmo Random Forest. Dentre as configurações avaliadas, os seguintes hiperparâmetros, a seguir apresentados, foram identificados como os mais eficazes:

- `max_depth`: Hiperparâmetro que controla a profundidade máxima de cada árvore na floresta (selecionado `max_depth=5`), indicando que cada árvore na floresta foi limitada a uma profundidade máxima de 5 nós;
- `random_state`: Este hiperparâmetro controla a aleatoriedade do modelo por meio do valor de semente usado pelo gerador de números aleatórios para reprodução dos resultados (adotado o valor fixo `random_state=21`), o que significa que os resultados do modelo serão os mesmos sempre que o código for executado com o mesmo conjunto de dados;
- **criterion**: Função responsável pela medição da qualidade da divisão em cada nó da árvore (selecionado índice *gini*);
- **min-samples-split**: Número mínimo de exemplos para dividir um nó (adotado valor 10);
- **min-samples-leaf**: Número mínimo de exemplos em uma folha (adotado valor 1).

2.5.4 Naive Bayes

NB é um algoritmo amplamente utilizado na classificação de objetos de diferentes categorias devido à sua simplicidade e eficácia, especialmente em grandes conjuntos de dados (66). No contexto de variáveis aleatórias $X_1, X_2, ..., X_n$ representando características do domínio D_X e uma variável aleatória não observada Y do domínio $D_Y = \{0, 1\}$, o NB assume independência condicional entre os atributos, de modo que cada atributo contribui de forma independente para a probabilidade de pertencer a uma classe específica, desconsiderando as dependências entre as variáveis do sistema (67).

NB opera sob a premissa da independência condicional entre suas variáveis, expressa como $P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n|Y = y) = \prod_{i=1}^{n} P(X_i = x_i|Y = y)$. O algoritmo, considerando C como o número de classes de Y, calcula a probabilidade posterior $P(Y = y|X = x)$ usando o teorema de Bayes (66):

$$P(Y = y|X = x) = \frac{P(Y = y) \times \prod_{i=1}^{n} P(X_i = x_i|Y = y)}{\sum_{y=1}^{C} P(Y = y) \times \prod_{i=1}^{n} P(X_i = x_i|Y = y)}$$

Nesse contexto, $P(Y = y)$ e $P(X_i = x_i|Y = y)$ representam as probabilidades a priori e condicionais de classe, respectivamente.

$$P(Y = y|X = x) = \frac{P(Y = y, X = x)}{P(X = x)} =$$

$$\frac{P(Y = y)P(X = x|Y = y)}{P(X = x)} =$$

$$\frac{P(Y = y)P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n|Y = y)}{\sum_{y=1}^{C} P(Y = y) \sum_{i=1}^{n} P(X_i = x_i|Y = y)}$$

Na prática, não se está interessado em $P(X = x)$. Em vez disso, o foco é normalizar para $P(Y = y|X = x) = 1$. Além disso, para estimar as probabilidades a priori $P(Y = y)$, utiliza-se a frequência de ocorrência de cada classe nos dados de treinamento, desde que haja um grande conjunto de dados de treinamento. Isso se deve ao fato de que o número de termos é determinado pelo número de possíveis instâncias multiplicado pelo número de valores-alvo possíveis, exigindo múltiplas ocorrências de cada instância no conjunto de treinamento para estimativas confiáveis (66).

Uma peculiaridade interessante do método de aprendizado NB é a ausência de uma busca explícita pelo espaço de hipóteses possíveis. Em vez disso, a hipótese é formada sem uma busca ativa, simplesmente contando a frequência das várias combinações de dados nos exemplos de treinamento.

De acordo com Faceli et al.(67), o NB se destaca por seu treinamento rápido e eficiente, realizando apenas uma varredura pelos dados de treinamento. Após o treinamento, a classificação de novas instâncias é direta, tornando-o ideal para cenários em tempo real. Além disso, ele demonstra insensibilidade a características irrelevantes, podendo lidar com esses atributos sem afetar sua precisão. Sua versatilidade também é evidente na capacidade de manipular tanto dados discretos quanto contínuos, tornando-o aplicável a uma variedade de problemas.

Entretanto, o NB apresenta algumas limitações. Sua suposição de independência condicional entre os atributos pode ser excessivamente simplista em certos casos, impactando a precisão do classificador. Além disso, em tarefas que envolvem pequenos conjuntos de dados, o NB pode apresentar resultados inferiores, uma vez que necessita de um número adequado de ocorrências de cada instância para fazer estimativas confiáveis (67). Esses aspectos positivos e limitações devem ser considerados ao utilizar o NB em diferentes contextos e problemas de classificação.

Considerando os diversos contextos, no ambiente do *Scikit-Learn*, é possível utilizar diferentes variações do NB, como o Gaussiano, Multinomial e o Bernoulli. Cada variação possui suas próprias aplicações específicas, dependendo das características dos dados e do problema em questão. Nos experimentos realizados nesta dissertação, essas variações foram testadas, e o modelo (`GaussianNB()`) do *Scikit-Learn* demonstrou um desempenho global superior.

2.5.5 Métricas de avaliação dos resultados

Os resultados dos classificadores podem ser avaliados utilizando as métricas acurácia, precisão, *recall* e pontuação F1 (*F1-Score*). TP (*True Positive*) representa o número de exemplos classificados corretamente como positivos, enquanto TN (*True Negative*) representa o número de exemplos classificados corretamente como negativos. FP (*False Positive*) representa o número de exemplos classificados erroneamente como positivos e FN (*False Negative*) representa o número de exemplos classificados erroneamente como negativos.

A acurácia mede a proporção de exemplos classificados corretamente em relação ao total do conjunto de dados. A precisão mede a proporção de verdadeiros positivos entre os exemplos classificados como positivos, enquanto o *recall* se refere à proporção de verdadeiros positivos em relação a todos os exemplos positivos. O *F1-Score* é uma métrica que combina a precisão e o *recall* em um único valor, fornecendo uma medida geral do desempenho do classificador. Quanto mais próximo o valor do *F1-Score* estiver de 1 (ou 100%), melhor será o desempenho do classificador em termos de precisão e *recall*, indicando um bom equilíbrio entre essas duas métricas. A seguir, apresentam-se as fórmulas para essas métricas:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precisão} = \frac{TP}{TP + FP}$$

$$\textit{Recall} = \frac{TP}{TP + FN}$$

$$\textit{F1-Score} = 2 \times \frac{\text{Precisão} \times \textit{Recall}}{\text{Precisão} + \textit{Recall}}$$

As matrizes de confusão são cruciais na avaliação de algoritmos de AM, especialmente em tarefas de classificação. Elas comparam as classificações reais com as previsões do algoritmo, proporcionando uma análise detalhada do desempenho do modelo. Essas matrizes representam instâncias reais em linhas e previsões em colunas.

Na matriz de confusão exemplificada na Figura 23, para a classe A, houve 3 previsões corretas (*True Positives*) e 5 *False Negatives* (amostras da classe A erroneamente classificadas como outras classes). A seguir, apresenta-se o cálculo da acurácia obtida pelo classificador, bem como a precisão, o *Recall* e o *F1-Score* da classe A.

1. **Acurácia**:
$$\frac{TP + TN}{TP + TN + FP + FN} = \frac{126}{180} = 0.7$$

2. **Precisão (Classe A)**:
$$\frac{TP}{TP + FP} = \frac{3}{3 + 5} = \frac{3}{8} = 0.375$$

3. ***Recall* (Classe A)**:
$$\frac{TP}{TP + FN} = \frac{3}{3 + 27} = \frac{3}{30} = 0.10$$

4. ***F1-Score* (Classe A)**:
$$2 \times \frac{\text{Precisão} \times \textit{Recall}}{\text{Precisão} + \textit{Recall}} = 2 \times \frac{0.375 \times 0.10}{0.375 + 0.10} \approx 0.16$$

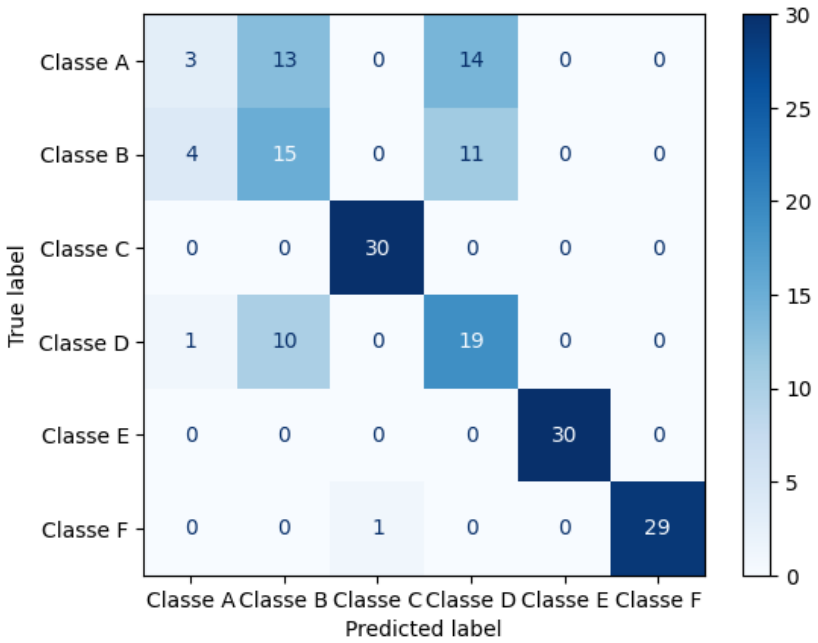


Figura 23 – Exemplo Matriz de Confusão

3 REVISÃO DA LITERATURA

Neste Capítulo, apresenta-se uma revisão bibliográfica sobre a identificação de padrões em cifras simétricas e assimétricas, com especial atenção para a aplicação de modelos de aprendizado de máquina na predição de resultados. Observou-se a ausência de pesquisas específicas sobre algoritmos de criptografia pós-quântica, e constatou-se que apenas 19% das pesquisas identificadas incluíram informações sobre os hiperparâmetros dos classificadores de aprendizado de máquina utilizados. Essas lacunas na literatura fundamentam a abordagem inovadora proposta nesta pesquisa.

3.1 Metodologia da pesquisa e bases de artigos científicos

A revisão bibliográfica foi conduzida através da análise de diversas fontes acadêmicas, incluindo veículos renomados como *Association for Computing Machinery* (ACM), *Institute of Electrical and Electronic Engineers* (IEEE), *Google Scholar*, *ScienceDirect/Elsevier*, *Springer*, o Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), *International Association for Cryptologic Research* (IACR), *Journal of Information Security and Cryptography* - ENIGMA e *Cryptologia Journal*. Essas fontes foram selecionadas por conterem pesquisas relevantes ao tema central desta pesquisa.

Foram realizadas buscas por trabalhos publicados desde 2001, em inglês e português, utilizando palavras-chave como *Identificação de Cifras*, *Ataque de Distinguir*, *Reconhecimento de Padrões*, *Classificação de Cifras*, *Aprendizado de Máquina* e *Identificação de Criptografia Pós-Quântica*. As referências bibliográficas dos trabalhos encontrados também foram examinadas para complementar o mapeamento dos estudos sobre o tema.

Os trabalhos identificados foram submetidos a um filtro baseado em três critérios: presença das palavras-chave no título ou resumo, avaliação preliminar de relevância pelo resumo e número de citações em outros trabalhos. Com base na aplicação desses critérios, os vinte e seis artigos apresentados a seguir foram considerados relevantes. Esses artigos constituíram a base para a análise e discussão dos resultados, conforme detalhado na Tabela 9.

De acordo com os princípios de design de cifras, elas não devem vazar ou propagar qualquer informação no texto cifrado que possa revelar detalhes do texto claro, da chave criptográfica usada ou do criptossistema.

De acordo com os princípios teóricos e de design das cifras, espera-se que uma cifra não revele informações sobre o texto claro, a chave criptográfica ou a própria cifra que foi utilizada para criptografar o texto claro (83, 84, 85, 86, 87). Identificar uma cifra apenas

com base nos textos cifrados é um desafio significativo. O texto cifrado, por sua natureza, não possui estrutura e, portanto, é intrinsecamente difícil inferir informações relacionadas à sua sintaxe, semântica ou léxico (88).

Em síntese, a metodologia ilustrada na Figura 24 é comumente utilizada pelos pesquisadores para identificar algoritmos criptográficos. Nessa abordagem, uma base de dados é criptografada por vários criptossistemas, gerando criptogramas, que, através de diferentes métodos, são representados como vetores. Posteriormente, estes vetores representativos são analisados por algoritmos classificadores de aprendizado de máquina, levando à identificação do criptossistema.

Tabela 9 – Critérios para análise dos trabalhos relacionados

Item	**Critério**
1	Abordagem de aprendizado de máquina
2	Algoritmos de aprendizado de máquina
3	Hiperparâmetros dos algoritmos de aprendizado de máquina
4	Medidas de desempenho
5	Algoritmos de criptografia
6	Modo de operação
7	Tipos de chave e vetor de inicialização
8	Tamanho e quantidade de amostras analisadas
9	Tipo de criptografia

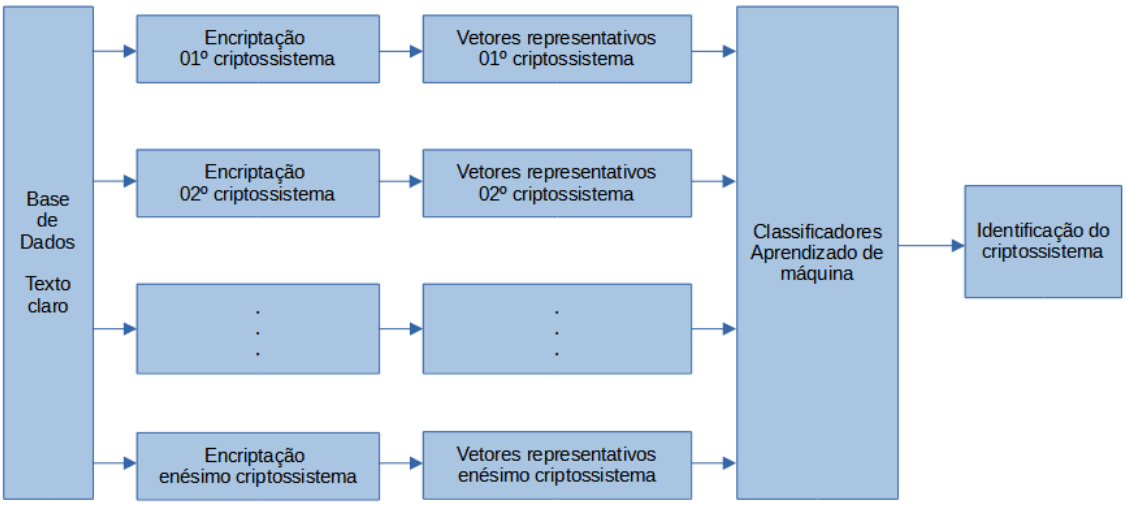


Figura 24 – Modelo genérico de identificação de criptossistemas

3.2 Trabalhos relacionados

1. *Identification of block ciphers using support vector machines* (2006)

Dileep e Chandra Sekhar (18) analisaram a frequência de elementos do alfabeto e seus n-gramas, enquanto os algoritmos de aprendizado de máquina trataram a identificação

do algoritmo de criptografia como uma tarefa de classificação de padrões. Essa método supervisionado revelou assinaturas distintas dos criptossistemas ao utilizar uma quantidade específica de textos cifrados.

Os autores adotaram um modelo *bag-of-words* para representar sequências de bits com comprimentos fixos e variáveis. Para palavras de comprimento fixo, eles selecionaram sequências de 4 bits nos textos cifrados, resultando em um alfabeto de 16 palavras. Para palavras de comprimento variável, os autores identificaram as três palavras mais frequentes em um *corpus* de 50 textos cifrados, cada um com 4000 bits.

Na etapa de experimentação da pesquisa, os textos cifrados foram gerados a partir de um texto claro de 4000 bits, que continha 500 caracteres ASCII. Foram utilizados os seguintes algoritmos de criptografia e modos de operação: DES no modo ECB, DES no modo CBC, 3DES, Blowfish, AES e RC5 no modo ECB, e foram usadas chaves aleatórias para cada algoritmo.

Os autores realizaram experimentos utilizando os classificadores KNN com valores de k igual a 5, 15 e 25, além do SVM com os *kernels* linear, polinomial, sigmoide e gaussiano. Os resultados mostraram que o classificador SVM obteve as maiores taxas de acurácia, variando de 27,25% a 95%. Em particular, o kernel gaussiano apresentou os melhores resultados.

2. *Application of data mining in cryptanalysis* (2009)

Khadivi e Homayouni (89) aplicaram técnicas de mineração de dados para analisar 59 imagens bitmap e 59 arquivos de texto criptografados pelo algoritmo AES, usando um modelo de criptografia por substituição clássico. Apesar de não fornecerem detalhes específicos sobre o AES, o modelo de substituição, o modo de operação e os tipos de chaves utilizados, os autores concluíram que a distribuição de frequência de símbolos em imagens é mais uniforme do que em textos. Baseando-se nessa análise, estabeleceram uma regra para distinguir dados criptografados pelo AES: se a variância dos dados criptografados for menor que 1.448×10^{-5}, então a base de dados é um texto; caso contrário, é uma imagem.

A variância nos dados criptografados representa a dispersão dos símbolos ou padrões na sequência binária criptografada. Os textos criptografados apresentam uma variância menor, indicando uma distribuição mais concentrada de símbolos, enquanto as imagens criptografadas possuem uma distribuição mais dispersa. A análise da variância possibilitou identificar padrões distintivos, permitindo a distinção entre textos e imagens criptografados pelo AES.

Os autores também consideraram a presença de sequências de pixels com a mesma cor e bits iguais em imagens, uma característica menos comum em textos, para distinguir os dados criptografados. Definiram um fator de similaridade para medir a probabili-

dade de encontrar blocos iguais na saída criptografada e estabeleceram outra regra de classificação: se o fator de similaridade for maior que zero, os dados são considerados imagens; caso contrário, são classificados como texto. Nos experimentos realizados, todos os textos criptografados foram corretamente classificados, assim como 55,9% das imagens criptografadas.

Apesar de se concentrar no AES e em um modelo de substituição específico, este estudo é relevante, pois além de distinguir o algoritmo de criptografia, essa abordagem também permitiu identificar o tipo de base de dados criptografada, seja texto ou imagem.

3. *Cipher-text classification with data mining* (2010)

Khadivi e Momtazpour(90) aplicaram técnicas de mineração de dados para distinguir diferentes bases de dados criptografadas pelo AES, utilizando os classificadores LDA, KNN e SVM.

Embora os autores não tenham informado detalhes específicos do AES, do tamanho das amostras, dos tipos de chave (fixa ou variável) e dos modos de operação utilizados, na etapa de experimentação da pesquisa, os autores criptografaram 1600 amostras de livros infantis, 1600 amostras de textos relacionados a finanças e telecomunicações e 1600 amostras de imagens com o algoritmo AES.

Posteriormente, foi criado um alfabeto composto por palavras de 16 bits. O classificador SVM usou a função de kernel RBF, escolhida após experimentos preliminares. Para o KNN, o valor de K foi 20, também selecionado experimentalmente. Nos classificadores LDA e KNN, a distância euclidiana foi a métrica de distância. A validação cruzada com 10 *folds* foi utilizada para treinamento e teste dos classificadores.

Apesar da ausência de regras de decisão específicas, similar ao estudo anterior, a contribuição deste trabalho foi a expansão do escopo de bases de dados criptografadas pelo AES. Os resultados revelaram uma acurácia aproximada de 60% para o LDA, 62% para o KNN e 65% para o SVM na distinção das bases de dados criptografadas.

4. *Classifying encryption algorithms using pattern recognition techniques* (2010)

Suhaila O. Sharif, L.I. Kuncheva e S.P. Mansoor (91) conduziram um ataque de distinção nas cifras de bloco DES, IDEA, AES e RC2, usando os algoritmos de aprendizado de máquina Naive Bayes (NB), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Instance based learning (IBL), Bagging (Ba), AdaBoostM1, Rotation Forest (RoFo) e Decision Tree (C4.5).

No método supervisionado utilizado, os algoritmos DES (56 bits), IDEA (64 bits), AES (128, 192, 256 bits) e RC2 (42, 84, 128 bits) cifraram 30 arquivos de 512 KB no modo ECB, com chaves distintas. A natureza dos textos claros não foi especificada pelos autores.

Os arquivos criptografados foram divididos em blocos de 8 bits e histogramas de 8 bits foram gerados usando o software Matlab®, resultando em 256 *features* (2^8). Esses histogramas foram classificados usando a plataforma WEKA com validação cruzada de 10 *folds*.

Dois conjuntos de dados foram criados na pesquisa, A e B. O conjunto A tinha 8 classes, uma para cada algoritmo de criptografia, com 240 amostras (30 para cada algoritmo). O conjunto B tinha 4 classes, com 240 arquivos de entrada (60 para DES, IDEA, AES e RC2).

Os resultados foram apresentados apenas em termos da métrica acurácia. No conjunto A, o classificador RoFo teve a maior acurácia (30,83%), acertando 74 amostras de 240. Em contraste, o classificador IBL teve a menor acurácia (12,5%), classificando corretamente 30 amostras. No conjunto B, novamente o RoFo teve a maior acurácia (53,33%), classificando corretamente 128 amostras das 240. O IBL obteve a menor acurácia (30,42%), acertando 73 amostras.

5. *Identification of encryption algorithm using decision tree* (2011)

O estudo de Manjula et al. (92) buscou distinguir os algoritmos de criptografia DES, 3DES, AES, Blowfish, RC2, RC4, IDEA, RSA e ECC, usando o algoritmo de aprendizado de máquina C4.5 a partir dos textos cifrados correspondentes. Os autores não mencionaram o modo de operação utilizado nos experimentos, nem os tipos e tamanhos das chaves.

Durante a fase de experimentação da pesquisa, os pesquisadores criptografaram o mesmo arquivo de texto claro usando os criptossistemas mencionados com as ferramentas *Crypttool* e *SmartSec ECC encryptor*. Em seguida, geraram 2600 amostras rotuladas, construindo vetores representativos dos criptogramas com características relacionadas à entropia dos caracteres, das letras maiúsculas, das letras minúsculas, dos símbolos, dos números, dos trigramas e dos tetragramas presentes nos textos cifrados.

O conjunto de treinamento foi composto por aproximadamente 2000 arquivos cifrados, variando de 1 KB a 3 MB. Para a fase de testes, selecionaram-se cerca de 600 arquivos criptografados, também com diferentes tamanhos.

No primeiro experimento, foram avaliados modelos de classificação com base na quantidade de arquivos nos conjuntos de treinamento e testes. O conjunto de treinamento variou de 500 a 2000 arquivos, enquanto o conjunto de testes variou de 65 a 205 arquivos. Os resultados alcançaram taxas de identificação corretas entre 70.2% e 75.4%.

No segundo experimento, os modelos foram baseados no tamanho dos arquivos nos conjuntos de treinamento e testes. O conjunto de treinamento variou entre 150 e 1000 arquivos, enquanto o conjunto de testes variou entre 100 e 600 arquivos, com amostras de tamanhos entre 2000 e 100000 bytes. Os resultados obtidos revelaram taxas de identificação

corretas entre 72.2% e 78.4%.

Na comparação entre os dois experimentos, observou-se que o método de identificação por tamanho mostrou-se mais eficaz do que o método baseado na quantidade de arquivos. Este método necessitou de um número menor de amostras de treinamento para atingir uma taxa de identificação de 78,4%. Embora os autores tenham utilizado técnicas inovadoras, como a análise dos parâmetros de entropia e experimentos diferenciados por tamanho e quantidade de arquivos, essas abordagens, apesar de criativas, não são reproduzíveis devido à falta de detalhes. Portanto, sua eficácia não pode ser confirmada.

6. *Identification of N block ciphers* (2011)

William Souza et al. (93) desenvolveram um método de agrupamento utilizando a técnica hierárquica aglomerativa de ligação simples para identificar n-cifras de bloco com base em padrões presentes nos criptogramas. Esse método não supervisionado foi aplicado em um caso específico com $n = 5$, no qual os autores analisaram criptogramas gerados pelos algoritmos criptográficos MARS, RC6, Rijndael, Serpent e Twofish operando nos modos ECB e CBC.

Os criptogramas $c_i = (c_{1,i}, c_{2,i}, \ldots, c_{n,i})$ e $c_j = (c_{1,j}, c_{2,j}, \ldots, c_{n,j})$ tiveram suas similaridades comparadas usando o cosseno do ângulo entre eles, resultando numa matriz de similaridade. O estudo cifrou 30 textos claros em inglês em diferentes tamanhos (1024 a 10240 bytes) em blocos de 128 bits, nos modos ECB e CBC, totalizando 1350 criptogramas com a ferramenta *WARS Text*.

No primeiro experimento, os criptogramas foram separados em cinco grupos, cada um contendo criptogramas de uma única cifra de bloco, alcançando 100% de precisão. No entanto, a revocação variou de 3% a 97% devido ao alto nível de aleatoriedade no modo CBC. O o procedimento utilizado não misturou no mesmo grupo criptogramas cifrados por algoritmos diferentes.

Dada a evidência de que as propriedades intrínsecas de cada cifra geram assinaturas nos criptogramas, o segundo experimento teve por objetivo verificar se a combinação de diferentes cifras gera assinaturas nos criptogramas, de modo que a última cifra utilizada seja corretamente identificada. Cada coleção de criptogramas foi composta por 150 amostras, cifradas com uma única chave aleatória de 128 bits.

Os resultados demonstraram que os criptogramas foram separados com precisão máxima com base na última cifra utilizada na combinação. Isso indica que as assinaturas geradas pelas cifras anteriores foram eliminadas, e apenas a assinatura da última cifra permaneceu.

No terceiro experimento, foram realizadas duas fases para avaliar a segurança da combinação de cinco cifras de blocos. Na primeira fase, as quatro primeiras cifras

utilizaram o modo CBC e a última cifra usou o modo ECB. Na segunda fase, as quatro primeiras cifras usaram o modo ECB e a última cifra usou o modo CBC. Cada fase envolveu nove coleções, cada uma contendo 150 criptogramas gerados por composições de cinco cifras de blocos com uma chave aleatória de 128 bits, seguindo o mesmo procedimento do experimento anterior. O objetivo foi investigar se a segurança da combinação das cifras seria comprometida nessas configurações.

Os resultados do terceiro experimento mostraram que durante as quatro primeiras cifrações no modo CBC não houve repetição de padrões. Além disso, a quinta cifração no modo ECB não comprometeu o efeito das cifrações anteriores. No entanto, quando as quatro primeiras cifrações foram realizadas no modo ECB, seguidas por uma última cifração no modo CBC, todos os padrões repetidos propagados ao longo das quatro cifrações anteriores no modo ECB foram destruídos.

7. Método não supervisionado de Reconhecimento de padrões criptográficos (2012)

O estudo conduzido por Narazain (94) teve como objetivo diferenciar as cifras DES, BF, Camellia, AES, SEED, RC4, CAST, RC2 e 3DES por meio de seus correspondentes criptogramas gerados exclusivamente no modo CBC. Foi utilizado um método não supervisionado que envolveu a distância de Hamming, teoria dos grafos (coloração em grafo) e estatística Qui-quadrado (χ^2), com a mesma chave sendo empregada em todos os arquivos cifrados de cada algoritmo.

Foram conduzidos cinco experimentos para distinguir diferentes tipos de criptogramas. O primeiro experimento tinha como objetivo distinguir criptogramas gerados por Blowfish, DES e RC2. O segundo experimento visava distinguir criptogramas gerados por DES ou RC2. O terceiro experimento tinha como objetivo distinguir criptogramas gerados por DES e não-DES. O quarto experimento tinha como objetivo distinguir criptogramas gerados por RC6 ou não-RC6 utilizando a estatística Qui-quadrado (χ^2). Por fim, o quinto experimento tinha como objetivo distinguir criptogramas gerados por DES ou não-DES também utilizando a estatística Qui-quadrado (χ^2).

No primeiro experimento, 300 criptogramas de 8KB cifrados em modo CBC foram analisados, com uma precisão de 43,65% e uma revocação de 40,67%. Foram classificados corretamente 122 criptogramas.

No segundo experimento, 200 criptogramas de 8KB cifrados em modo CBC foram analisados, com uma precisão de 55,94% e uma revocação de 56%. Foram classificados corretamente 112 criptogramas.

No terceiro experimento, 500 criptogramas de 32KB cifrados em modo CBC por cinco algoritmos foram analisados, dos quais 100 foram gerados pelo DES e 400 por

outros algoritmos. A precisão foi de 54,80% e a revocação de 57,50%. Foram classificados corretamente 294 criptogramas.

No quarto experimento, 500 criptogramas de 32KB cifrados em modo CBC por cinco algoritmos foram analisados, dos quais 100 foram gerados pelo RC6 e 400 por outros algoritmos. Utilizando o ataque de distinção qui-quadrado sobre criptogramas cifrados pelo RC6, a precisão foi de 52,13% e a revocação de 51,88%. Foram classificados corretamente 157 criptogramas.

No quinto experimento, 500 criptogramas de 32KB cifrados em modo CBC por cinco algoritmos foram analisados, dos quais 100 foram gerados pelo RC6 e 400 por outros algoritmos. Utilizando o método proposto e o ataque de distinção qui-quadrado, a precisão foi de 45,17% e a revocação de 47%. Foram classificados corretamente 124 criptogramas. Utilizando apenas o método proposto, a precisão foi de 54,60% e a revocação de 57,25%. Foram classificados corretamente 266 criptogramas.

Com base nos resultados, verificou-se que o método proposto se mostrou mais eficiente na identificação de cifras operando no modo CBC em comparação ao teste do Qui-quadrado. O método proposto obteve uma taxa de acerto superior a 70% na classificação correta de criptogramas de 32KB gerados pelos algoritmos Camellia e SEED, identificando-os como não provenientes do DES. Por outro lado, o teste do Qui-quadrado apresentou uma taxa de sucesso média de apenas 49%. Isso demonstra a superioridade do método proposto, nos testes binários, frente à estatística Qui-quadrado (χ^2) na tarefa de identificar corretamente as cifras no modo CBC.

8. *On the effectiveness of using state-of-the-art machine learning techniques to launch cryptographic distinguishing attacks* (2012)

Chou et al. (95) analisaram criptogramas AES 128, DES e RC4 nos modos ECB e CBC provenientes de três diferentes bases de dados: texto (notícias da Reuters em 1987, com documentos $<$ 128 bytes), imagens (Caltech 101, 19.043 imagens JPEG), e áudio (MajorMinor, 2.174 arquivos WAVE).

Na fase de experimentação, foram selecionadas 1000 amostras de cada categoria. No conjunto de dados de texto, foram escolhidos os 1000 documentos de maior tamanho. No conjunto de dados de imagens, foram selecionadas as 1000 maiores imagens das categorias motocicleta e avião. Essa abordagem resultou em um conjunto menor, mas representativo, de amostras.

Todas as amostras foram criptografadas usando uma chave aleatória fixa para cada cifra e IVs aleatórios. Para cada amostra criptografada, um vetor representativo com doze *features* foi gerado. As duas primeiras características dizem respeito à entropia, calculada em diferentes palavras de 16 e 12 bits, respectivamente. As características três e

quatro representam a quantidade de símbolos diferentes presentes nos criptogramas. As características cinco a oito são histogramas de 16, 64 e 128 bits respectivamente. A nona característica é a contagem das palavras de 4 bits mais comuns. A décima característica é o tamanho dos intervalos entre palavras do formato 0x00. A décima primeira característica representa a quantidade de bits zero entre o primeiro e o 128º byte das amostras e a décima segunda característica é a entropia entre o primeiro e o 128º byte das amostras.

Em seguida, os vetores representativos foram divididos em cinco partes com a mesma quantidade de vetores representativos, sendo quatro partes usadas para treinar os algoritmos de aprendizado de máquina e uma parte usada para testar o desempenho dos algoritmos. Foi adotada a estratégia de validação cruzada, e os resultados finais foram calculados como a média dos resultados obtidos nos conjuntos de teste.

Após a classificação binária, os resultados mostraram que a menor acurácia do SVM variou entre 48.10% (após na análise de canal Reuters cifrado pelas cifras AES e DES utilizando apenas as duas últimas *features* no modo CBC) e 52.55% (canal Reuters cifrado com AES e DES analisando apenas quinta e décima *features* no modo CBC).

No experimento do modo ECB, com um índice de acerto aleatório de 50%, o SVM apresentou uma acurácia mínima de 50% ao analisar o canal de áudio MajorMinor cifrado por AES e DES, usando as quatro primeiras *features*, enquanto atingiu uma acurácia máxima de 100% ao cifrar o canal de texto Reuters também com AES e DES, considerando a quinta, sexta, sétima e oitava *features*. No modo CBC, também com índice aleatório de 50%, a menor acurácia do SVM foi 48.10% ao analisar o canal de texto da Reuters cifrado por AES e DES, utilizando apenas as duas últimas características, enquanto a acurácia máxima de 52.55% foi alcançada após cifrar o canal Reuters com AES e DES, considerando apenas a quinta e décima características.

Com base nesses resultados, os autores concluíram que, ao contrário das demais pesquisas, os algoritmos de aprendizado de máquina não conseguem extrair informações úteis de criptogramas produzidos por cifras modernas no modo CBC ao considerar as *features* utilizadas. Além disso, não conseguiram distinguir esses criptogramas com uma probabilidade maior que o aleatório.

9. *Pattern analysis of cipher text: A combined approach* (2013)

Mishra et al. (96) propuseram uma nova abordagem para identificar criptogramas gerados pelos algoritmos AES, DES e Blowfish no modo de operação ECB, empregando chaves aleatórias.

A abordagem proposta divide o processo de identificação em três etapas sequenciais. No primeiro estágio, avaliou-se o tamanho dos blocos dos criptogramas. No segundo estágio, foram analisadas a entropia, definida por Robert Gray (97) como a medida de incerteza

na variável aleatória e como medida de incerteza ou aleatoriedade de variáveis aleatórias individuais ou combinadas, juntamente com a recorrência das amostras, caracterizada por um padrão em que algum evento ocorre com intervalo de tempo constante. No último estágio, utilizou-se o algoritmo C4.5 para classificar os criptogramas.

Para avaliar o desempenho do sistema proposto, os autores utilizaram conjuntos de amostras de 128, 256, 512 e 1024 bits provenientes de arquivos de texto no formato UTF8. Foram selecionadas 10, 200, 700 e 2000 amostras de cada tamanho, respectivamente.

Os resultados mostram que utilizando 10 amostras de criptogramas de 128 bits no modo ECB, foi alcançada uma acurácia de 83% na classificação. À medida que o quantidade de amostras aumentou para 200, 700 e 2000, houve alteração na acurácia do modelo, com valores de 64%, 87.3% e 89.1%, respectivamente. Criptogramas com 512 bits apresentaram acurácia de 87.3%, enquanto os criptogramas de 1024 bits alcançaram uma acurácia de 89.1%.

Embora os autores não tenham fornecido as acurácias dos algoritmos de criptografia nem especificado qual teve o melhor desempenho, os resultados experimentais indicam que a abordagem combinada proposta é eficaz na identificação de criptogramas gerados pelos algoritmos AES, DES e Blowfish no modo ECB, usando chaves aleatórias. A eficácia alcançada superou significativamente o índice aleatório de 33,34%.

10. *A distinguishing attack with a neural network* (2013)

Souza e Tomlinson(98) propõem o uso de redes neurais artificiais para identificar cifras de bloco. O objetivo é agrupar criptogramas gerados por diferentes cifras, como MARS, RC6, Rijndael, Serpent e Twofish, utilizando uma chave de 128 bits.

Os experimentos foram realizados utilizando 30 textos claros em dois tamanhos: 6144 e 8192 bytes. Esses textos foram criptografados pelos cinco algoritmos finalistas do concurso AES, com uma chave única de 128 bits e no modo ECB. O objetivo era agrupar os criptogramas em cinco clusters correspondentes às diferentes cifras de bloco.

No primeiro experimento, foram utilizados 150 criptogramas de 8192 bytes gerados pelas cifras de bloco. Uma rede neural com 400 neurônios foi configurada para o treinamento, usando a medida do ângulo cosseno, 10 épocas, taxa de aprendizado de 0,9 e vizinhança de 400 neurônios. A fase de convergência foi realizada em 10 épocas, com taxa de aprendizado de 0,01 e vizinhança de 1.

Os resultados mostraram clusters bem definidos, onde cada cifra de bloco e os textos aprendidos pelos neurônios foram representados no mapa. Essa distinção foi possível devido às características intrínsecas das cifras, que geram assinaturas nos criptogramas, permitindo sua identificação. É importante destacar que a fase de convergência não teve um impacto significativo na melhoria do agrupamento ou na posição dos criptogramas no

mapa.

No segundo experimento, uma coleção de criptogramas de 6144 bytes foi empregada com a mesma configuração de redes neurais. No entanto, de acordo com os autores, os resultados não foram satisfatórios, e os experimentos com a coleção de 6144 bytes não foram bem-sucedidos.

Corroborando os resultados dos trabalhos relacionados, os autores concluíram que uma coleção de criptogramas gerados no modo ECB pode ser identificada e posteriormente atacada por uma rede neural, contanto que os criptogramas tenham pelo menos 8192 bytes e a rede neural esteja adequadamente configurada, especialmente no número de neurônios.

11. *Review of a new distinguishing attack using block cipher with a neural network* (2014)

Assim como na pesquisa de William Souza et al. (93), Lomte e Shinde (99) também empregaram um método não supervisionado baseado em redes neurais para agrupar n cifras de bloco, sendo aplicável a qualquer valor de n. O estudo avaliou cinco cifras de bloco: MARS, RC6, Rijndael, Serpent e Twofish, cada uma com uma chave única de 128 bits.

Para representar os textos cifrados, foram adotados vetores de n dimensões, onde n representa o número de blocos distintos na coleção de criptogramas. Cada bloco representa um eixo no espaço vetorial, permitindo que um texto cifrado seja mapeado como um ponto nesse espaço. Na etapa subsequente da metodologia proposta, foram empregadas redes neurais compostas por um conjunto de neurônios, sendo cada um deles associado a um peso sináptico de n-dimensões. Neste estudo, o cálculo dos pesos sinápticos dos neurônios foi realizado utilizando o ângulo de cosseno.

Durante a fase experimental da pesquisa, foi conduzido um teste inicial para avaliar a capacidade das redes neurais em identificar padrões e formar clusters em textos claros de 6144 e 8192 bytes em oito idiomas diferentes, incluindo português, espanhol, francês, alemão, dinamarquês, holandês, grego e hebraico. Semelhante à pesquisa (98), uma rede neural foi configurada com 400 neurônios, 10 épocas, taxa de aprendizado de 0,9 e vizinhança de 400 neurônios. Os resultados obtidos foram bem-sucedidos, com a formação de clusters bem definidos que confirmaram a hipótese proposta.

Na fase seguinte, cada uma das cinco algoritmos de cifra de bloco, MARS, RC6, Rijndael, Serpent e Twofish, gerou 5 criptogramas a partir de textos claros, totalizando 45 criptogramas. Os criptogramas foram analisados em blocos de 16 bytes (128 bits) por uma rede neural com 400 neurônios. A medida de ângulo de cosseno foi utilizada, juntamente com um tamanho de vizinhança de 400. As autoras não forneceram informações sobre o modo de operação utilizado, a natureza dos textos claros utilizados na pesquisa nem o

tamanho dos criptogramas gerados.

Na conclusão da pesquisa, ao analisar o mapa bidimensional resultante do uso da rede neural, observou-se que o processo de agrupamento de cifras foi bem-sucedido, no qual rodos os criptogramas submetidos à rede neural foram agrupados corretamente nos clusters formados, confirmando a eficácia do ataque proposto para classificar e distinguir as cifras de bloco analisadas.

12. *Cryptographic algorithm identification using machine learning and massive processing* (2016)

No estudo realizado por Flávio Mello e Xexéo (100), o objetivo era identificar algoritmos criptográficos a partir de criptogramas gerados pelos algoritmos RC4, Blowfish, DES, Rijdael, RSA, Serpent e Twofish, no modo ECB, utilizando chaves aleatórias. Para isso, eles encriptaram uma base de dados composta por textos.

Os textos utilizados neste experimento são de sete idiomas distintos: Português, Espanhol, Inglês, Alemão, Hebreu, Cirílico e Mandarim, no formato Unicode. Foram selecionadas 600 amostras de textos distintos para cada idioma, extraídos de jornais e revistas, sem repetição de sentenças, com um mínimo de 140.000 caracteres em cada corpus.

Os criptogramas foram representados por histogramas que indicam a quantidade de ocorrências de blocos de bits em cada criptograma, juntamente com a marcação do algoritmo criptográfico utilizado. Para identificar os algoritmos criptográficos, foram empregados os classificadores C4.5, Naive Bayes, PART, Multilayer Perceptron, FT e WiSARD, analisando blocos de 2 bits até 34 bits.

Do corpus encriptado, 66% foi usado para treinar o modelo de classificação e 34% foi destinado aos testes. Foi garantido que cada algoritmo criptográfico tivesse a mesma quantidade de exemplares de amostras, totalizando 2.772 instâncias de treino para cada algoritmo criptográfico e 1.428 instâncias para o conjunto de teste.

No classificador C4.5, as taxas de acerto na identificação dos algoritmos convergiram para quase 100% a partir de um tamanho de blocos de 28 bits, exceto na identificação do ARC4. Os classificadores PART e FT não apresentaram tal convergência, mas ainda assim mostraram uma capacidade crescente de identificação. Além disso, em todos esses três classificadores, houve um incremento significativo na taxa de acerto na identificação do DES e do Blowfish quando a quantidade de bits utilizada era maior que 16 bits.

O classificador Complement Naive Bayes apresentou os melhores resultados, sendo capaz de identificar todos com 100% de precisão (exceto o RSA) a partir de tamanho de blocos de 21 bits. Neste classificador, também houve um salto de qualidade na identificação

dos algoritmos DES, Blowfish e Rijndael quando o tamanho de bloco ultrapassou 16 bits. Os classificadores Multilayer Perceptron e WiSARD também mostraram uma capacidade crescente de identificação, sendo o segundo melhor que o primeiro. Por fim, o classificador Complement Naive Bayes apresentou os melhores resultados de forma geral, uma vez que convergiu mais rapidamente que o segundo melhor classificador, o C4.5.

13. *Machine learning for cryptographic algorithm identification* (2016)

Em continuidade à pesquisa (100), Flávio Mello et al. (101) abordam a análise de textos criptografados com o objetivo de identificar os algoritmos criptográficos utilizados. Embora o estudo tenha avaliado quatro algoritmos criptográficos, a metodologia é genérica e pode ser aplicada a uma ampla gama de algoritmos.

Na fase experimental da pesquisa, foram selecionados arquivos de texto em português, espanhol e inglês. Essa seleção permitiu a comparação dos resultados e a avaliação da sensibilidade dos algoritmos criptográficos em diferentes circunstâncias, já que os dois primeiros idiomas são linguisticamente mais complexos e o último tem uma estrutura mais simples.

Cada corpus utilizado na pesquisa foi composto por 200 amostras distintas de textos claros coletados de jornais e revistas. Cada amostra tinha no mínimo 6.000 caracteres e não havia repetição de fragmentos de texto. Os algoritmos DES, Blowfish, RC4 e RSA foram aplicados para criptografar cada arquivo de texto. No entanto, os autores não mencionaram o modo de operação utilizado nem se as chaves foram fixas ou aleatórias. A escolha desses algoritmos permitiu uma análise abrangente de cifras de bloco, cifras de fluxo, algoritmos de chave simétrica e algoritmos de chave pública.

Para representar os criptogramas analisados, diferentes histogramas foram construídos utilizando blocos de tamanho que variavam de 4 a 16 bits. Posteriormente, os algoritmos J48, FT, PART, Complement Naive Bayes e Multilayer Perceptron foram utilizados para proceder a classificação.

O classificador J48 mostrou uma taxa de identificação mais alta para DES, Blowfish e RSA com blocos de 16 bits, destacando-se o RSA com uma precisão de 87,77%. No caso do FT, a codificação RC4 foi bem identificada, especialmente com blocos de 8 bits, alcançando uma precisão de 66,01%. O classificador PART apresentou resultados semelhantes ao J48 e FT, com destaque para o RC4, que foi mais facilmente identificado com blocos de 4 bits, atingindo uma precisão de 70%.

No Complement Naive Bayes, houve um aumento significativo na precisão para todos os algoritmos, destacando-se o Blowfish, quase totalmente reconhecido com 99,54% de acurácia, e o RSA com uma acurácia de 89,36%. Tanto DES quanto RC4 foram completamente identificados (100%), sendo que o RC4 foi plenamente identificado em

blocos de 8 a 16 bits, ao contrário dos outros classificadores. O classificador Multilayer Perceptron não pôde ser calculado para blocos com 12 bits ou mais devido a limitações de memória no computador usado. Entretanto, obteve 100% de precisão na identificação do RSA a partir de blocos de 8 bits.

A metodologia empregada na pesquisa permitiu a identificação dos algoritmos com base em blocos de tamanhos específicos. Segundo os autores, blocos de 4, 8 e 16 bits são adequados para essa classificação, embora blocos maiores, como 24 e 32 bits, também possam ser relevantes. No caso específico do algoritmo RC4, os resultados demonstram que ele apresentou bom desempenho com blocos de 4 bits.

14. *An approach to identifying cryptographic algorithm from ciphertext* (2016)

Cheng Tan (17) empregou SVM na identificação dos algoritmos AES, Blowfish, 3DES, RC5 e DES no modo ECB. Os tamanhos dos textos claros analisados foram 0.8KB, 4KB, 20KB, 100KB e 500KB.

Inicialmente, foram criados conjuntos de textos cifrados para diferentes tamanhos, totalizando 1100 textos cifrados em cada conjunto, com 220 textos para cada algoritmo criptográfico estudado. Desses, 40 textos de cada algoritmo foram selecionados como conjunto de treinamento, enquanto os 180 textos restantes foram divididos em 9 grupos de 20 para formar o conjunto de testes. Os autores utilizaram o modelo *bag-of-words* com palavras de 4 bits para compor os vetores representativos dos criptogramas analisados.

Foram conduzidos três experimentos distintos. No primeiro experimento, utilizou-se a mesma chave para os conjuntos de treinamento e testes. No segundo experimento, empregaram-se chaves diferentes para os conjuntos de treinamento e testes. Por fim, no terceiro experimento, foi realizada a identificação um contra todos (*one against all*).

No primeiro experimento, ao analisar criptogramas de 4KB, 20KB, 100KB e 500KB dos algoritmos AES, Blowfish, 3DES, RC5 e DES, as acurácias foram 29.67%, 85.11%, 96.56% e 98%, respectivamente. Para criptogramas de 0.8KB, 4KB, 20KB, 100KB e 500KB dos algoritmos Blowfish, 3DES, RC5 e DES, as acurácias foram 28.33%, 98.33%, 99.17%, 100% e 100%, respectivamente.

No segundo experimento, ao analisar criptogramas de 4KB, 20KB, 100KB e 500KB dos algoritmos AES, Blowfish, 3DES, RC5 e DES, as acurácias foram 20%, 35.89%, 39.11% e 37.67%, respectivamente. Para criptogramas de 4KB, 20KB, 100KB e 500KB dos algoritmos Blowfish, 3DES, RC5 e DES, as acurácias foram 24.86%, 21.94%, 21.94% e 22.92%, respectivamente.

No terceiro experimento, os pesquisadores avaliaram a taxa de identificação entre diferentes algoritmos criptográficos em amostras de tamanhos variados. Para amostras de 500KB, a taxa de identificação entre AES e Blowfish foi de 94,17%, enquanto a taxa

entre AES e 3DES foi de 93,89%. Para amostras de 100KB, as taxas de identificação para AES-Blowfish e AES-3DES foram de 94,17%. Em amostras de 20KB, a taxa de identificação variou, sendo 88,09% para AES-RC5 e 89,44% para AES-DES. Em amostras de 4KB, as taxas de identificação diminuíram para todos os algoritmos, ficando em 50% para AES-Blowfish, AES-3DES, AES-RC5 e AES-DES. Também foram feitas outras comparações entre os algoritmos, como Blowfish-3DES, que obteve uma taxa de identificação de 43,89% para amostras de 4KB e 50,83% para amostras de 20KB.

Os resultados dos três experimentos confirmam conclusões de trabalhos relacionados. A identificação eficaz do algoritmo criptográfico é facilitada quando as mesmas chaves são utilizadas nos conjuntos treinamento e teste dos textos cifrados. No entanto, ao utilizar chaves diferentes, a identificação dos algoritmos se torna notoriamente mais difícil. Além disso, os pesquisadores observaram a capacidade de superar o índice aleatório na identificação um a um com o algoritmo AES. Embora não explicitamente mencionado no texto da pesquisa, há uma tendência de aumento na acurácia com amostras maiores.

15. *Machine learning applied to the recognition of cryptographic algorithms used for multimedia encryption* (2017)

Barbosa e Mello (102) empregaram mineração de dados para identificar cifras proveniente de arquivos multimídia criptografados pelos algoritmos DES, Blowfish, RSA e RC4 no modo ECB.

Os experimentos foram conduzidos em dois corpus distintos: um composto por arquivos de áudio no formato mp3 e outro por arquivos de vídeo nos formatos mp4 e avi. Cada corpus continha 250 amostras de arquivos diferentes. Os arquivos de áudio tinham, no mínimo, 3 MB de tamanho, enquanto os arquivos de vídeo possuíam, no mínimo, 1 GB.

Os autores não especificaram se as chaves eram fixas ou aleatórias. Após a encriptação dos arquivos, foram gerados histogramas referentes a quantidade de blocos de bits com tamanhos variando de 4 bits a 16 bits. Em seguida, os corpus foram divididos em duas partes: 66% para a montagem do modelo de classificação e 34% para os testes, mantendo a mesma quantidade de amostras de cada algoritmo em cada parte do corpus, evitando qualquer viés na identificação dos algoritmos.

Os algoritmos J48, FT, PART, Complement Naive Bayes e Multilayer Perceptron foram empregados para classificar os criptogramas. Os resultados indicaram que a identificação das cifras superou o acaso, especialmente o Complement Naive Bayes, que obteve uma taxa média de acurácia superior a 96% na classificação de criptogramas gerados pelo algoritmo RC4. Notou-se que o aumento da quantidade de informação contribuiu para o aumento da acurácia, sendo que blocos de 12, 14 e 16 bits foram eficazes na identificação das cifras DES, Blowfish e RSA, com uma taxa média de acurácia de 97% para o bloco de

16 bits.

Os resultados dos experimentos indicaram que o tamanho do bloco influencia a taxa de precisão dos algoritmos criptográficos. O ARC4 apresentou 70% de acurácia em blocos de 4 bits, enquanto o RSA foi identificado com acurácia de 60% em blocos de 8 bits. No classificador FT, o ARC4 foi reconhecido com uma taxa média de identificação de aproximadamente 70%, e os algoritmos DES, Blowfish e RSA tiveram uma taxa de acerto de 55% em blocos de 16 bits, comparado ao índice aleatório de 25%.

No classificador PART, o ARC4 foi identificado em 60% dos casos em blocos de 4 bits, e o RSA obteve uma taxa média superior a 84% em blocos de 16 bits. O classificador Complement Naive Bayes alcançou uma acurácia de aproximadamente 100% para todos os algoritmos em blocos de 16 bits. Por fim, no Multilayer Perceptron, o ARC4 alcançou 100% de acurácia em blocos de 11 bits, destacando-se também os blocos de 8 bits para a identificação do DES, Blowfish e RSA, com acurácias superiores a 50%, sendo o índice aleatório de 25%.

16. *Identifying Encryption Algorithms in ECB and CBC Modes Using Computational Intelligence* (2018)

Mello e Xexéo (103) empregaram os algoritmos C4.5, PART, FT, Complement Naive Bayes, Multilayer Perceptron e WiSARD para identificar os algoritmos DES, Blowfish, RSA, ARC4, Rijndael, Serpent e Twofish nos modos de operação ECB e CBC, utilizando corpora de textos claros nos idiomas português, espanhol, inglês, alemão, hebraico, cirílico e mandarim.

Os corpora de textos claros utilizados consistiram em 4.200 amostras divididas em sete corpora distintos, representando cada sistema de idioma. Cada corpus era composto por 600 amostras de textos diferentes coletados de jornais e revistas, sem repetição de frases. Cada amostra continha pelo menos 140.000 caracteres, correspondendo a aproximadamente 35 páginas de texto, um tamanho incomum para artigos de jornais e revistas. Para evitar padrões de chave no processo de mineração de dados, cada arquivo de texto em claro foi associado a uma chave única gerada por um gerador de bits pseudoaleatório (PRNG).

Cada arquivo criptografado foi representado por um histograma que representou a quantidade de blocos de bits de 2 a 34 bits. O limite superior de 34 bits foi determinado pela capacidade de memória e implementações dos módulos de processamento de dados e técnicas de aprendizado de máquina.

Os resultados revelaram uma identificação bem-sucedida dos algoritmos de criptografia no modo ECB, atingindo uma taxa de identificação de 100%. No entanto, o resultado mais notável foi a identificação dos algoritmos no modo CBC. Embora as taxas de identificação no modo CBC tenham sido mais baixas em comparação ao modo ECB,

elas alcançaram 50% e superaram a probabilidade aleatória (14,28%). Além disso, essas taxas aumentaram consistentemente e podem ser aprimoradas com computação intensiva. O classificador mais eficaz foi o Complement Naive Bayes, tanto em termos de identificação bem-sucedida quanto em relação ao tempo de execução.

17. *Identification of block ciphers under CBC mode* (2018)

Cheng Tan et al. (104) desenvolveram um método supervisionado para identificar as cifras AES, DES, 3DES, RC5 e Blowfish no modo CBC. Este modo de criptografia é considerado mais seguro do que o modo ECB devido aos altos níveis de aleatoriedade presentes nos textos cifrados gerados.

Na abordagem proposta para o ataque *Ciphertext Only*, os autores concatenaram os primeiros blocos de cada arquivo de texto cifrado, formando um arquivo composto por múltiplos primeiros blocos concatenados. Em seguida, identificaram e representaram as características desse arquivo como vetores. Esses vetores representativos foram usados para criar um banco de dados de características do texto cifrado, denominado *Feature Database* (FDB), sendo que cada cifra de bloco possui seu próprio FDB.

A metodologia proposta pelos autores se destaca por sua inovação ao abordar a identificação de cifras de bloco no modo CBC, conhecido pela alta não-linearidade dos textos cifrados, o que dificulta a identificação de padrões discerníveis. No entanto, a falta de informações detalhadas no artigo sobre os critérios utilizados para a identificação e extração das características dos criptogramas representa uma limitação significativa. Essa lacuna compromete a compreensão integral da metodologia e impede a reprodução dos experimentos.

Os autores prepararam 200 arquivos de texto cifrado para cada cifra, sendo que os primeiros 20 foram destinados ao treinamento do modelo de identificação e os 180 restantes foram utilizados no conjunto de testes. Quatro situações distintas foram analisadas com base nas chaves e IVs dos arquivos de texto cifrado: mesma chave e mesmo IV, mesma chave e IVs diferentes, chaves diferentes e mesmo IV, e chaves diferentes e IVs diferentes.

Na metodologia apresentada, o algoritmo SVM foi utilizado para classificar as cifras analisadas. Na identificação multiclasse, a taxa de identificação das cinco cifras de bloco analisadas foi de 98.44% para arquivos de texto cifrado de 500 KB, e 93% quando chaves e vetores de inicialização (IV) eram iguais e o tamanho do arquivo era de 100 KB. Na identificação um a um, o criptossistema AES foi diferenciado das outras quatro cifras de bloco quando as chaves ou IVs eram diferentes, com uma acurácia superior a 97% para arquivos de texto cifrado com tamanho igual a 100 KB. Mesmo para arquivos de texto cifrado com tamanho de apenas 4 KB, a taxa média de identificação foi superior a 80%.

18. *Machine learning approach for analysing encrypted data* (2018)

Pradeepthi e Saxena (105) propuseram uma abordagem inovadora baseada em clusterização para identificar algoritmos de criptografia em um cenário *Only ciphertext*. Eles introduziram uma representação alternativa dos criptogramas usando arquivos de diferença.

Para criar os arquivos de diferença, os autores calcularam a diferença entre cada par consecutivo de linhas por meio de uma operação XOR nos arquivos criptografados, que possuem n linhas, armazenando o resultado em um novo arquivo chamado arquivos de diferença. A partir do texto cifrado original, foi gerado o arquivo f_1, a partir do qual foi gerado o arquivo f_2, e assim por diante até chegar ao arquivo f_5.

No experimento, cinco textos distintos em inglês foram criptografados pelos algoritmos DES e L-Block, utilizando chaves de 64/56 bits e 80 bits, respectivamente (106). A partir de cada arquivo cifrado, foram gerados cinco arquivos de diferença para cada cifra. No entanto, informações sobre o modo de operação, a natureza das chaves (fixas ou aleatórias) e o tamanho dos textos claros não foram fornecidas. No total, foram produzidos 50 arquivos de diferença, os quais foram usados como entrada para a próxima etapa do experimento. Nessa fase, os dados foram analisados por algoritmos de clusterização disponíveis no software Weka, e o número de clusters em cada arquivo de diferença foi calculado.

Os autores exploraram diversos algoritmos de aprendizado de máquina, incluindo Canopy, Coweb, Farthest First, Filtered Cluster, Hierarchical Clustering, Density Based, Simple K-Means e Expectation Maximization (107), para a tarefa de clusterização. No entanto, devido à necessidade de conhecer previamente o número de clusters em muitos desses algoritmos, optaram por utilizar exclusivamente o algoritmo Expectation-Maximization. A escolha foi motivada pelo fato de que o Expectation-Maximization é capaz de agrupar os pontos de dados em clusters sem a exigência de um número predefinido de clusters, tornando-o uma opção adequada para o presente estudo.

Após a conclusão dos experimentos, a metodologia proposta demonstrou resultados eficazes. Os autores conseguiram distinguir os algoritmos de criptografia em diferentes configurações de clusters, variando de 0 a 60 clusters, com base no número de clusters obtidos para cada arquivo de diferença.

19. *Analysis of cryptosystem recognition scheme based on Euclidean distance feature extraction in three machine learning classifiers* (2019)

Fan e YaQun (108) utilizaram a distância euclidiana de textos cifrados gerados pelos algoritmos DES, 3DES, AES-128, AES-256, IDEA, SMS, Blowfish e Camellia-128 nos modos ECB e CBC como *features* nos classificadores Random Forest, Regressão Logística

e SVM.

No total, 1.001 amostras de 512 KB foram criptografadas para cada algoritmo e modo de operação, resultando em 16.016 arquivos de texto cifrado, todos usando chaves fixas. Os autores não especificaram se os vetores de inicialização foram fixos ou aleatórios.

Os resultados dos experimentos não foram categorizados por classe de algoritmo. O Random Forest obteve uma acurácia de 76,08%, enquanto o SVM e a Regressão Logística atingiram 38,81% e 64,84% de precisão, respectivamente. Contudo, na conclusão do estudo, os autores informaram que a acurácia geral dos classificadores no modo ECB foi de 75% e 13,5% no modo CBC, ambas superiores ao índice aleatório (12,5%).

Os resultados indicaram que o desempenho dos classificadores é fortemente influenciado pelo modo de operação utilizado na cifragem dos dados. O modo ECB apresentou uma acurácia significativamente superior a do modo CBC, um resultado esperado devido às diferentes características desses modos de operação.

20. *Block ciphers classification based on random forest* (2019)

No método supervisionado apresentado em (109), Hu e Yaqun utilizaram o Random Forest para classificar as cifras de bloco AES-128, AES-256, Blowfish-64, Camelia-128, DES-56, 3DES-56, IDEA-64 e SMS4-128 nos modos de operação ECB e CBC.

No método utilizado pelos autores, os textos cifrados foram divididos em blocos de 8 bits. A frequência de ocorrência dos bits foi observada de modo que as *features* f_1, f_2, ..., f_d foram obtidas, onde d representa a dimensão da característica.

Inicialmente, o primeiro bit de cada byte foi extraído para formar o vetor $c^1 = \{c_1^1, c_2^1, \ldots, c_{n/64}^1\}$, cujo comprimento é $n/64$ bytes. Em seguida, as frequências dos $n/64$ bytes $c_1^1, c_2^1, \ldots, c_{n/64}^1$ em c^1 foram registradas como a primeira característica extraída. Em seguida, o 2º bit, o 3º bit e assim por diante, até o 8º bit dos textos cifrados foram analisados e os textos cifrados correspondentes $c^2 = \{ c_1^2, c_2^2, ..., c_{n/64}^2\}$, $c^3 = \{ c_1^3, c_2^3, ..., c_{n/64}^3\}$, ..., $c^8 = \{ c_1^8, c_2^8, ..., c_{n/64}^8\}$ foram obtidos.

Durante a fase de coleta de dados, 1001 textos claros de 512KB do *dataset* Caltech-256 foram criptografados usando oito cifras mencionadas anteriormente nos modos ECB e CBC, resultando em um total de 16016 arquivos criptografados. Desses, 8008 foram cifrados no modo ECB e 8008 no modo CBC. Para cada cifra, os textos claros foram criptografados com a mesma chave, tanto na fase de treinamento quanto na fase de teste.

No experimento com o Random Forest, 7200 arquivos foram selecionados aleatoriamente dos 8008 arquivos criptografados para formar o conjunto de treinamento, enquanto os 808 arquivos restantes foram usados como conjunto de teste, em cada modo de operação. Para garantir a confiabilidade dos resultados, cada classificação foi repetida 10 vezes, e a

acurácia geral foi calculada pela média aritmética das 10 classificações.

O modelo proposto alcançou uma acurácia média de 87,9% no modo ECB, enquanto a classificação aleatória seria de 12,5%. No modo CBC, a acurácia média do método foi ligeiramente superior ao índice aleatório, atingindo 12,64%. Além da falta de eficácia do método proposto no modo CBC, o resultado ressalta a aleatoriedade desse modo e a consequente dificuldade de classificação. É importante notar que há poucos estudos que utilizam o modo CBC como objeto de classificação, o que dificulta a comparação com pesquisas existentes.

21. *Cryptosystem identification scheme based on ASCII code statistics* (2020)

Zhang et al. (2020) propuseram um método supervisionado para identificar os criptossistemas AES-128, AES-256, Blowfish-64, Camellia-128, DES, 3DES, IDEA-64 e SMS4-128 utilizando características estatísticas do código ASCII e empregando os classificadores Random Forest e MLP (110).

O estudo utilizou o *dataset* Caltech-256, composto por 30.607 imagens (111). 1000 arquivos de 512 KB foram criptografados nos modos ECB e CBC, utilizando a mesma chave aleatória. O mesmo IV também foi gerado aleatoriamente para a análise do modo CBC.

O código ASCII é um conjunto de códigos numéricos utilizado para representar caracteres em dispositivos eletrônicos. Cada caractere ASCII é representado por um número de 7 bits, variando de 0 a 127, e pode ser convertido em sua representação em caracteres correspondentes. Existem também versões estendidas do ASCII que incluem até 256 caracteres (112).

As imagens foram inicialmente convertidas em matrizes codificadas em base64 antes de serem armazenadas no banco de dados. Devido às variações na distribuição dos pixels entre diferentes imagens, houve diferenças na distribuição do código ASCII nos dados criptografados. Os autores analisaram a frequência dos códigos ASCII nos arquivos criptografados e usaram essa informação para criar um alfabeto, considerando os 256 códigos ASCII. Como resultado, os vetores representativos de cada arquivo criptografado tinham 256 dimensões.

O banco de dados foi dividido em duas partes: 70% foi designado como conjunto de treinamento e o restante utilizado como conjunto de teste. A precisão do método foi avaliada usando as métricas acurácia, precisão, *recall* e *F1-score*. No modo ECB, o método proposto alcançou uma acurácia de 84,5% ao empregar o classificador Random Forest. Entretanto, no modo CBC, devido às características estatísticas de criptografia serem mais complexas, a acurácia do esquema foi consideravelmente menor, atingindo 14%, ainda que superior ao índice aleatório de 12,5%.

22. *Classification of encryption algorithms based on ciphertext using pattern recognition techniques* (2020)

Em um cenário *Ciphertext Only*, Kavitha et al (113) propuseram um método para identificar as cifras IDEA, DES, AES e RC2 no modo ECB. Eles utilizaram os classificadores SVM, Bagging, AdaBoost, Neural Networks, NB, IBL, RoFo e DT.

Os autores cifraram 200 textos claros de 512 KB usando os algoritmos DES (56 bits), IDEA (64 bits), AES (128, 192 e 256 bits) e RC2 (42, 84 e 128 bits) no modo de operação ECB. Para cada algoritmo, 40 textos cifrados foram escolhidos para treinamento, enquanto os 160 restantes foram divididos em oito conjuntos de 20 textos cifrados para os testes.

Os autores empregaram o método *bag-of-words* para identificar os algoritmos criptográficos. No entanto, não foram apresentados detalhes sobre o tamanho das palavras utilizadas. Posteriormente, o conjunto de *features* foi dividido em dois conjuntos: o conjunto P, com oito classes e 220 arquivos de entrada, e o conjunto Q, com quatro classes e também 220 arquivos de entrada.

No conjunto de dados P, verificou-se que o classificador RoFo obteve uma acurácia de 31,90%, superando o índice aleatório de 25%. Em contrapartida, o classificador IBL apresentou a menor acurácia, com apenas 11,50% dos dados de entrada classificados corretamente, o que representa apenas 28 dos 220 arquivos. Os resultados no conjunto de dados Q mostraram que o RoFo novamente obteve a melhor performance, alcançando uma acurácia de 54,32%, classificando corretamente 130 dos 220 dados. Por outro lado, o classificador IBL apresentou a menor acurácia, com apenas 30,20% dos dados corretamente classificados, representando somente 73 dos arquivos de entrada.

23. *Cryptographic algorithms identification based on deep learning* (2020)

Ruiqi Xia et al. (28) propuseram uma metodologia baseada na aplicação de testes estatísticos da NIST Statistical Test Suite (NIST STS) (30), uma bateria de testes usada para avaliar a qualidade da aleatoriedade de sequências de bits, incluindo textos cifrados gerados por algoritmos criptográficos. Essa metodologia foi capaz de identificar padrões em textos cifrados produzidos pelos algoritmos AES, Kasumi, 3DES, Present, RSA e ElGamal operando no modo CBC.

Para conduzir os experimentos, os autores coletaram aproximadamente 1,2 GB de textos claros do Open American National Corpus (OANC). Em seguida, esse conjunto de 1,2 GB foi dividido em 1000 partes, cada uma com aproximadamente 1,1 MB. Durante a encriptação, os autores geraram os criptogramas usando chaves aleatórias. No entanto, não especificaram se os vetores de inicialização (VI) foram iguais ou aleatórios.

Os autores utilizaram os valores p retornados pelos testes estatísticos *Frequency within blocks*, *Runs* e *Serial*, dentre os quinze testes disponíveis na NIST STS, para compor os vetores representativos dos criptogramas.

Posteriormente, para a classificação dos vetores representativos, os autores utilizaram redes neurais artificiais com a função de ativação $ReLU$ e a camada básica de convolução *Conv1D*. O treinamento foi realizado em 200 épocas, com uma taxa de aprendizado de 0,01 e um tamanho de lote de 500. Do total das amostras, 60% foi usado para o conjunto de treinamento e os 40% restantes para o conjunto de teste.

Os resultados da classificação foram expressos em termos de precisão, revocação e acurácia. O AES-128 alcançou uma precisão de 89.23%, revocação de 72.1% e acurácia de 90.05%. O KASUMI apresentou uma precisão de 92.36%, revocação de 66.38% e acurácia de 91.46%. O 3DES obteve uma precisão de 90.6%, revocação de 69.05% e acurácia de 93.65%. O PRESENT demonstrou uma precisão de 93.12%, revocação de 65.28% e acurácia de 95.72%. O RSA alcançou uma precisão de 88.45%, revocação de 74.51% e acurácia de 88.79%. Por fim, o ElGamal apresentou uma precisão de 89.77%, revocação de 74.21% e acurácia de 90.93%.

Os resultados demonstraram que essa abordagem possui uma acurácia superior a 95% na correta identificação das cifras, em determinados casos. É importante destacar que esses índices são consideravelmente superiores aos obtidos por Mello e Xexéo (103), que apresentaram a melhor taxa de identificação entre os trabalhos relacionados analisados nesta dissertação.

Os autores não justificaram a escolha dos testes selecionados. Além disso, a análise da pesquisa não permite determinar se os altos índices obtidos são atribuíveis à representação dos vetores pelos valores p, ao uso da rede neural ou ao grande tamanho das amostras analisadas.

24. *Block Cipher Algorithm Identification Scheme Based on Hybrid Random Forest and Logistic Regression Model* (2022)

Complementando a metodologia proposta por Xia, Li e Chen(28), Yuan et al.(11) e Yuan et al.(12) utilizaram valores p de 10 testes da NIST STS (30) para compor os vetores representativos de 500 criptogramas de 1, 8, 64, 256 e 512 KB gerados pelos algoritmos AES, DES, Blowfish, Camellia, RC2 e RC6 no modo ECB. É importante notar que os autores empregaram dados aleatórios gerados pelo módulo criptográfico Fortuna do Python para o texto claro, utilizando o método do Acumulador Fortuna. Em ambas as pesquisas, foram usadas chaves iguais na encriptação.

No estudo (11), os pesquisadores utilizaram os classificadores RF, SVM e KNN, alcançando acurácias em torno de 70%. Por outro lado, em (12), os autores adotaram a

técnica de *ensemble learning*, combinando Random Forest e SVM. Com essa abordagem, conseguiram uma taxa de acerto superior a 72,5%. O *ensemble learning* envolve a combinação de múltiplos modelos de aprendizado de máquina para melhorar o desempenho e a precisão das previsões, o que resultou em uma melhoria na identificação dos algoritmos criptográficos.

25. *A block cipher algorithm identification scheme based on hybrid k-nearest neighbor and random forest algorithm* (2022)

Na continuação da pesquisa (11), Ke Yuan et al. (12) utilizaram valores p provenientes de 10 testes da NIST STS (30) para compor os vetores representativos de 500 criptogramas de 1, 8, 64, 256 e 512 KB gerados pelas cifras AES, 3DES, Blowfish, Cast e RC2 no modo ECB, empregando chaves idênticas na encriptação. Assim como na pesquisa anterior, os autores também utilizaram dados aleatórios gerados pelo módulo criptográfico Fortuna do Python para o texto claro, utilizando o método do Acumulador Fortuna.

A classificação do *dataset* e, consequentemente, a identificação dos criptossistemas, foram realizadas pelos classificadores SVM, KNN, RF e por um modelo híbrido baseado em *ensemble learning* denominado *Hybrid K-Nearest Neighbor and Random Forest* (HKNNRF). Esse modelo também alcançou taxas de identificação próximas a 70%, resultados semelhantes aos das pesquisas (11) e (12).

26. *Block ciphers identification scheme based on randomness test* (2022)

Caracterizando o atual estado da arte e consolidando a emprego de testes de aleatoriedade e aprendizado de máquina para realizar ataques de distinção, Xiuli Yu e Kai Shi Yu e Shi(114) reduzem a dimensionalidade dos vetores representativos dos criptogramas analisados ao empregar cinco dos quinze testes componentes da NIST STS (30). Essa quantidade de testes é maior do que a adotada em Xia, Li e Chen(28), que utilizou três testes de aleatoriedade, e é menor do que as adotadas em Yuan et al.(11) e Yuan et al.(12), que utilizaram 10 testes de aleatoriedade.

Os pesquisadores criptografaram 4000 arquivos de 256 KB do conjunto de dados de imagens Caltech-256 usando os algoritmos DES, AES, 3DES e Blowfish. A mesma chave aleatória foi aplicada nos modos de operação ECB e CBC. No conjunto de dados formado para o estudo, 75% dos dados foram dedicados ao treinamento do classificador MLP, enquanto os 25% restantes foram reservados para o conjunto de teste.

A classificação foi conduzida utilizando o algoritmo MLP. No modo ECB, o modelo alcançou uma acurácia de 42%, superando a precisão da classificação aleatória de 25%. O AES mostrou a maior precisão com 47,2%, enquanto o 3DES teve o pior desempenho, com cerca de 37%. Em termos de *recall*, o 3DES obteve o melhor resultado, aproximadamente

43,5%, enquanto o DES apresentou o pior desempenho, com 37,6%. A pontuação F1 para o AES foi de 43,9%, representando o melhor desempenho na classificação, enquanto os outros três algoritmos criptográficos tiveram resultados ligeiramente acima de 37%.

No modo CBC, a acurácia dos quatro algoritmos criptográficos foi de 29,8%, sensivelmente inferior à acurácia obtida no modo ECB, mas ligeiramente maior do que a probabilidade de classificação aleatória. Os resultados dos outros três índices variaram de 25% a 32%, próximos à taxa de acerto de 25% da classificação aleatória.

Os resultados destacam a dificuldade em distinguir algoritmos criptográficos no modo CBC em comparação com o modo ECB, e indicam que construir vetores representativos apenas com os valores p da NIST STS, sem o auxílio de outras metodologias, não é suficiente para superar a aleatoriedade proveniente do modo CBC.

Embora não tenha sido explicitado na pesquisa, a disparidade significativa nos valores p retornados pelos testes de aleatoriedade parece estar relacionada ao tamanho do bloco do AES em comparação aos outros três tipos de algoritmos criptográficos, resultando em uma acurácia maior para a classe AES em comparação com os outros três algoritmos criptográficos.

3.3 Análise e Considerações sobre Trabalhos Relacionados

Conforme os critérios estabelecidos para orientar a análise dos trabalhos relacionados (tabela 9), foram criadas as tabelas 10, 11, 12, 13 e 14 apresentadas no final deste Capítulo. Essas tabelas sintetizam as informações contidas nos trabalhos relacionados que foram identificados pelo método de pesquisa utilizado nesta dissertação.

As duas primeiras tabelas abordam o uso de algoritmos de aprendizado de máquina em ataques de distinção. A primeira tabela fornece detalhes sobre as abordagens e algoritmos de aprendizado de máquina empregados, juntamente com os hiperparâmetros e medidas de avaliação de desempenho adotadas. Por sua vez, a segunda tabela agrupa os estudos relacionados com base nos algoritmos de aprendizado de máquina utilizados e nas abordagens aplicadas, permitindo identificar quais classificadores e abordagens foram mais comumente adotados.

A terceira tabela é semelhante à primeira, pois detalha os algoritmos de criptografia analisados, juntamente com os modos de operação, condições das chaves e vetores de inicialização. Também são mencionadas a quantidade de amostras e seus tamanhos, além da acurácia obtida nas pesquisas e os tipos de criptografia investigados. Complementando essas informações, a quarta tabela apresenta as metodologias utilizadas nas pesquisas identificadas, assim como a qualidade das cifras analisadas, especificando se são cifras de bloco, de fluxo, simétricas ou assimétricas. Por fim, a quinta tabela permite identificar

quais criptogramas foram mais amplamente analisados na literatura. Apesar de não ser empregado aprendizado de máquina na análise de criptogramas, optou-se por incluir a pesquisa de (94) em virtude da sua relevância na análise do modo de operação CBC.

Diferentemente dos esquemas de identificação baseados em histogramas ou no modelo *bag-of-words*, um esquema de identificação fundamentado em testes de aleatoriedade cria os vetores representativos $Fea = \{fea_1, fea_2, \ldots, fea_M\}$ dos criptogramas $C_1, C_2, \ldots, C_N$ por meio dos valores p obtidos após a submissão dos criptogramas à NIST STS. Nesse contexto, M representa a dimensão do vetor representativo, e cada *feature* corresponde a um valor p. O nome do algoritmo criptográfico responsável pela geração do texto cifrado é utilizado como rótulo (*Label*). Os conjuntos de amostras rotuladas (Fea, Lab) são, então, introduzidos nos modelos de classificação de aprendizado de máquina, que são treinados e ajustados.

Durante a fase de teste, após o treinamento dos algoritmos de aprendizado de máquina utilizando os conjuntos (Fea, Lab), o rótulo *Label* é removido, e apenas as características (Fea) constituem o conjunto de teste. Este conjunto é então inserido no modelo de classificação já treinado para obter a identificação do algoritmo criptográfico.

As últimas quatro pesquisas analisadas representam o estado da arte na literatura sobre ataques de distinção de cifras, destacando o uso de testes de aleatoriedade e modelos de aprendizado de máquina para essa tarefa. É crucial observar que a escolha dos testes estatísticos e das características extraídas dos criptogramas pode ter um impacto significativo na acurácia do classificador. Além disso, o tamanho dos arquivos analisados e o modo de operação da cifra também desempenham importante papel na capacidade do algoritmo de aprendizado de máquina de realizar a identificação.

Após a consolidação e análise dos trabalhos relacionados nesta dissertação, cujos dados foram compilados nas Tabelas 10, 11, 12, 13 e 14, presentes no Apêndice 6, levando em consideração os critérios de análise apresentados na Tabela 9, podem-se fazer as seguintes considerações:

1. Para identificar padrões e prever o algoritmo criptográfico gerador de um determinado criptograma, alguns autores usaram algoritmos de aprendizado de máquina não supervisionados para agrupar os criptogramas, enquanto outros usaram algoritmos de aprendizado de máquina supervisionados para classificá-los através do treinamento de modelos. De acordo com os dados resumidos nas Tabelas 10 e 11, a abordagem não supervisionada foi adotada em 19,23% das pesquisas, enquanto a abordagem supervisionada foi utilizada em 80,77% delas, com a maioria apresentando resultados superiores ao acaso. As métricas de avaliação de desempenho mais comuns foram acurácia, precisão, revocação (*recall*) e pontuação F1 (*F1-score*). Com base nos resultados, a abordagem supervisionada obteve melhores resultados, especialmente

em *recall*. Portanto, ela será adotada nesta pesquisa para permitir a comparação do método proposto com outros trabalhos relacionados presentes na literatura;

2. Ao analisar a Tabela 11, observa-se que entre os algoritmos de aprendizado de máquina mais utilizados estão K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest (RF) e Naive Bayes (NB). Considerando este fato, esses algoritmos foram selecionados para serem empregados nos experimentos desta dissertação. Essa escolha foi baseada não apenas na ampla utilização desses algoritmos em trabalhos relacionados, mas também nas abordagens e características específicas que cada um oferece. Essa diversidade permite uma análise abrangente na identificação de padrões nos criptogramas gerados pelos criptossistemas pós-quânticos mencionados;

3. Predominantemente, os textos foram amplamente utilizados nos trabalhos relacionados, o que também é adotado nesta pesquisa para permitir comparações. As bases de dados variaram em tamanho nos estudos analisados. Por exemplo, Dileep e Sekhar(18) utilizaram 100 textos de 4000 bits, Manjula e Anitha(92) empregaram 2000 textos de tamanhos diversos, Barbosa, Vidal e Mello(101) analisaram 200 textos com pelo menos 6000 caracteres cada, e Mello e Xexeo(103) utilizaram 600 amostras com pelo menos 140.000 caracteres. Também foram encontrados estudos como Tan, Deng e Zhang(104), que utilizaram 200 amostras de tamanhos variados (4 KB, 20 KB, 100 KB e 500 KB), Fan e Zhao(108), que utilizaram 16.016 amostras de 512 KB, Kavitha et al.(113), que utilizaram 200 arquivos de texto com tamanho fixo de 512 KB, e Yu e Shi(114), que geraram 4000 arquivos de 256 KB a partir do conjunto de dados de imagens Caltech-256. Além disso, alguns estudos, como Souza e Tomlinson(98) e Lomte e Shinde(99), utilizaram corpora em vários idiomas, enquanto Mello e Xexeo(103) analisou textos em diferentes idiomas. Em contraste, Yuan et al.(12) utilizou números aleatórios gerados pelo método do Acumulador Fortuna do módulo de criptografia Crypto em Python como textos claros. Para identificar padrões nos criptogramas, foi utilizada uma extensa base de textos com tamanhos variados, seguindo a abordagem de estudos anteriores Yuan et al.(11) e Yuan et al.(12). Mais informações sobre a base de dados estão detalhadas no Capítulo 4;

4. Dentre os trabalhos relacionados, cinco pesquisas adotaram abordagem não supervisionada. Dessas, três utilizaram um pré-processamento baseado no cálculo da similaridade entre os criptogramas, utilizando a medida de cosseno. Em relação às pesquisas que adotaram abordagem supervisionada, foram empregados diferentes métodos: 50% utilizaram o método *bag-of-words*, 15% utilizaram histogramas, 11% utilizaram índices estatísticos e 15% utilizaram testes estatísticos da bateria de testes do NIST (30). O estado da arte atual, representado pelas pesquisas de Tan, Deng e Zhang(104), Yu e Shi(114), Yuan et al.(11) e Yuan et al.(12), utiliza a NIST STS (30) para gerar vetores representativos dos criptogramas analisados. Neste estudo,

na esteira do atual estado da arte e visando a possibilidade de comparações, também adotou-se a abordagem dos autores mencionados, utilizando os *P-values* da NIST STS para compor os vetores representativos dos criptogramas;

5. Apesar de ser uma metodologia simples e efetiva, nenhum dos autores que utilizaram os testes estatísticos do NIST alcançou uma identificação total ou uma acurácia de 100% em seus experimentos. Xia, Li e Chen(28) aplicou os testes *Frequency within a Block*, *Runs* e *Serial*, enquanto Yuan et al.(12) e Yuan et al.(11) utilizaram um conjunto mais abrangente de testes, incluindo *Approximate Entropy*, *Cumulative Sums (Cusums)*, *Discrete Fourier Transform (Spectral)*, *Frequency within a Block*, *Random Excursions*, *Random Excursions Variant*, *Serial*, *Runs*, *Maurer's "Universal Statistical"* e *Binary Matrix Rank*. Por sua vez, Yu e Shi(29) empregou *Frequency (Monobit)*, *Runs*, *Approximate Entropy*, *Non-overlapping Template Matching* e *Frequency within a Block*. A fim de verificar se esses resultados foram devidos a limitações anteriores e a uma escolha inadequada do espaço de testes nos experimentos, esta pesquisa adotará todos os testes da NIST STS para maximizar as taxas corretas de identificação de algoritmos criptográficos;

6. Uma parte dos trabalhos relacionados utiliza a combinação de Precisão e Recall como medidas de desempenho, derivando também o F1-Score, enquanto outros trabalhos utilizam a Acurácia. A Acurácia é uma medida geral de desempenho, mas não fornece informações específicas sobre a taxa de acerto do classificador em cada classe. Essas informações mais detalhadas são obtidas através das medidas de Precisão, Recall e F1-Score. Portanto, as medidas de desempenho mencionadas serão utilizadas para descrever os resultados alcançados pelos algoritmos de aprendizado nos experimentos desta pesquisa; e

7. Após analisar os trabalhos relacionados, foi observado que não há pesquisas que abordem especificamente criptossistemas pós-quânticos. Foi identificado que uma parte significativa desses trabalhos se concentra apenas no modo de operação ECB, utilizando chaves fixas e aleatórias. Com base nessa constatação, o objetivo desta pesquisa é preencher essa lacuna, explorando a análise de algoritmos pós-quânticos nos modos de operação ECB e CBC, utilizando chaves e vetores de inicialização aleatórios.

4 EXPERIMENTOS

O propósito deste estudo foi identificar padrões nos criptogramas gerados pelos esquemas de criptografia de chave pública pós-quânticos e nas chaves de sessão compartilhadas pelos KEM Frodo, Crystals Kyber, NTRU e Saber. Essa análise foi conduzida com base nas hipóteses apresentadas na Introdução. Diante do atual estado da arte, para atingir esse objetivo, foi adotada uma abordagem que integrou testes estatísticos da NIST STS (NIST SP 800-22) e técnicas de aprendizado de máquina, metodologia similar àquelas utilizadas por Xia, Li e Chen(28), Yuan et al.(11), Yuan et al.(12), Yu e Shi(114) e Yuan et al.(115).

Tendo em vista que não foram encontradas pesquisas relacionadas à identificação de padrões em criptogramas gerados por algoritmos pós-quânticos, além dos algoritmos pós-quânticos, os experimentos descritos nas próximas subseções incorporaram as cifras simétricas AES e Blowfish. Essa inclusão teve como objetivo possibilitar comparações com trabalhos relacionados.

Todos os experimentos foram conduzidos em um computador equipado com um processador Intel® Core™ i5-2450M CPU @ 2.50GHz × 4 e 6.0 GiB de RAM, rodando o sistema operacional Ubuntu 22.04 LTS de 64 bits. Para a análise dos criptogramas e das chaves de sessão compartilhadas, foi utilizada a ferramenta de código aberto NIST STS (*sp800_22_tests-master*), previamente empregada por Yuan et al.(11) e Yuan et al.(12), e a análise dos vetores representativos foi realizada pelos classificadores KNN, NB, SVM e RF disponíveis na plataforma *Scikit-Learn*.

A base de dados usada consistiu na concatenação de 67,5 MB (70.846.486 bytes) de textos em língua portuguesa, abrangendo diversos gêneros literários (116, 117, 118, 119, 120, 121). No modo ECB, diferentes chaves aleatórias e únicas foram empregadas, enquanto no modo CBC, IVs e chaves igualmente únicas e aleatórias foram utilizados, sem repetição.

A escolha da língua portuguesa como idioma base foi fundamentada em estudos anteriores, como o de Mello e Xexeo(103), que demonstraram que diferentes idiomas e gêneros literários não influenciam o processo de classificação. Optou-se por utilizar chaves e IVs aleatórios e não repetidos com o objetivo de prevenir possíveis vieses na análise dos dados (100).

Os resultados de cada classificador foram analisados usando as matrizes de confusão provenientes dos algoritmos selecionados, permitindo o cálculo das métricas de avaliação acurácia, precisão, *recall* e *F1-Score*.

Com o propósito de fornecer uma visualização mais acessível e simplificada dos resultados dos experimentos deste Capítulo, foram apresentadas as matrizes de confusão dos conjuntos de testes. Todos os dados consolidados foram organizadas nas tabelas 15, 16, 17, 18, 19 e 20, disponíveis no Apêndice 6, com o objetivo de facilitar a análise.

4.1 Experimento - Modo ECB

4.1.1 Organização do experimento

As Figuras 25 e 26 representam a metodologia empregada para identificar padrões no modo ECB. O experimento envolveu análises em três cenários diferentes, considerando arquivos com tamanhos de 20KB, 60KB e 100KB, com o objetivo de avaliar a eficácia do método proposto em diferentes volumes de dados. Para cada tamanho de arquivo mencionado, foram selecionados aleatoriamente 100 textos distintos e não repetidos.

Nos três cenários, foi empregado o mesmo número de amostras para avaliar todos os algoritmos criptográficos, garantindo uma comparação imparcial e eliminando possíveis viéses. Após a criptografia, os textos cifrados gerados foram submetidos ao NIST STS. Utilizando todos os resultados disponíveis, os resultados dos testes estatísticos forneceram 41 valores p. Esses valores, juntamente com a contagem de bits 0s e 1s no criptograma analisado, foram usados para formar os vetores representativos dos criptogramas. Portanto, cada criptograma foi representado por um vetor contendo 43 campos.

Para cada um dos três *datasets* do experimento, contendo 600 vetores representativos (100 para cada criptossistema), duas estratégias de divisão de dados foram empregadas. Inicialmente, adotou-se a estratégia *train-test split*, dividindo os conjuntos em dois grupos: um de treinamento, representando 70% das amostras totais (420 amostras), e outro de teste, com os 30% restantes (180 amostras). Essa divisão foi feita considerando as conclusões de Stapor et al.(122), que apontam um equilíbrio entre viés e variância, indicando proporções próximas a 66% e 33% entre os conjuntos de treinamento e teste. Em uma segunda abordagem, para cada um dos três *datasets*, adotou-se a estratégia *cross-validation* com o valor de K igual a 10.

De acordo com Borra e Ciaccio(123), o viés do método K-*fold* diminui à medida que o valor de K aumenta. Contudo, um valor de K muito elevado aumenta o custo computacional da estratégia e implica em uma amostra de teste pequena, elevando a variância. Na literatura, discute-se o valor ideal de K, sendo comuns as escolhas de $K = 2, 5$ ou 10. Segundo Kohavi(124), $K = 10$ apresenta melhor desempenho. Similarmente, Borra e Ciaccio(123) e Kim(125) também apontaram $K = 10$ em suas pesquisas. Considerando que $K = 10$ foi frequentemente utilizado em pesquisas anteriores, optou-se por esse valor. A estratégia de *cross-validation* com K igual a 10 consiste na divisão do conjunto de

dados em 10 partes aproximadamente iguais, chamadas de *folds*. O classificador é treinado em nove desses *folds* e testado no restante, repetindo-se esse processo 10 vezes com *folds* diferentes como conjuntos de testes. Ao final, são obtidas 10 métricas de desempenho, cuja média proporciona uma avaliação robusta do modelo, reduzindo o impacto de variações nos dados de treinamento e teste.

Conforme apresentado no Capítulo 2, os algoritmos pós-quânticos utilizam diferentes construções matemáticas baseadas em problemas relacionados a reticulados: Crystals-Kyber (*Module Learning With Errors* - MLWE), Saber (*Learning With Round* - MLWR) e Frodo (*Learning With Errors* - LWE). Devido a essas construções distintas e às características únicas dos dados de entrada, existe a probabilidade de que os criptogramas provenientes desses algoritmos apresentem diferenças estatísticas, de modo a tornar suas amostras identificáveis por meio dos padrões existentes nos criptogramas.

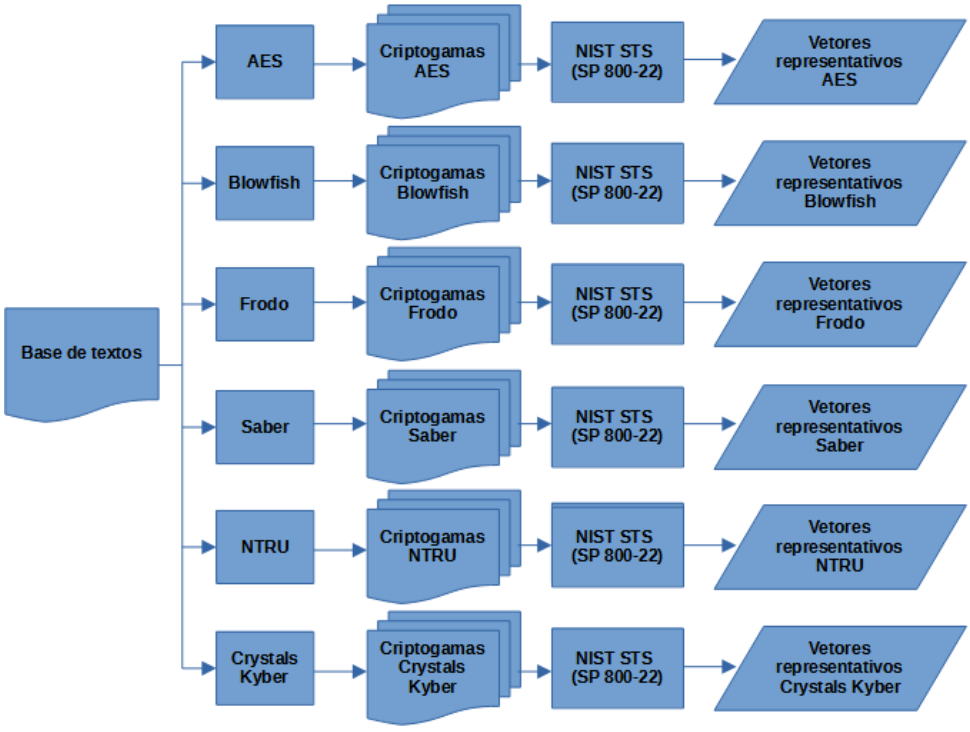


Figura 25 – Construção dos vetores representativos - modo ECB

4.1.2 Resultado Experimento ECB *Dataset* 20KB Estratégia *train-test split*

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 27, 28, 29 e 30, verifica-se que:

- **Para a classe AES:**
  - O KNN classificou corretamente todas as 30 amostras, alcançando uma precisão, *recall* e *F1-score* de 100%.

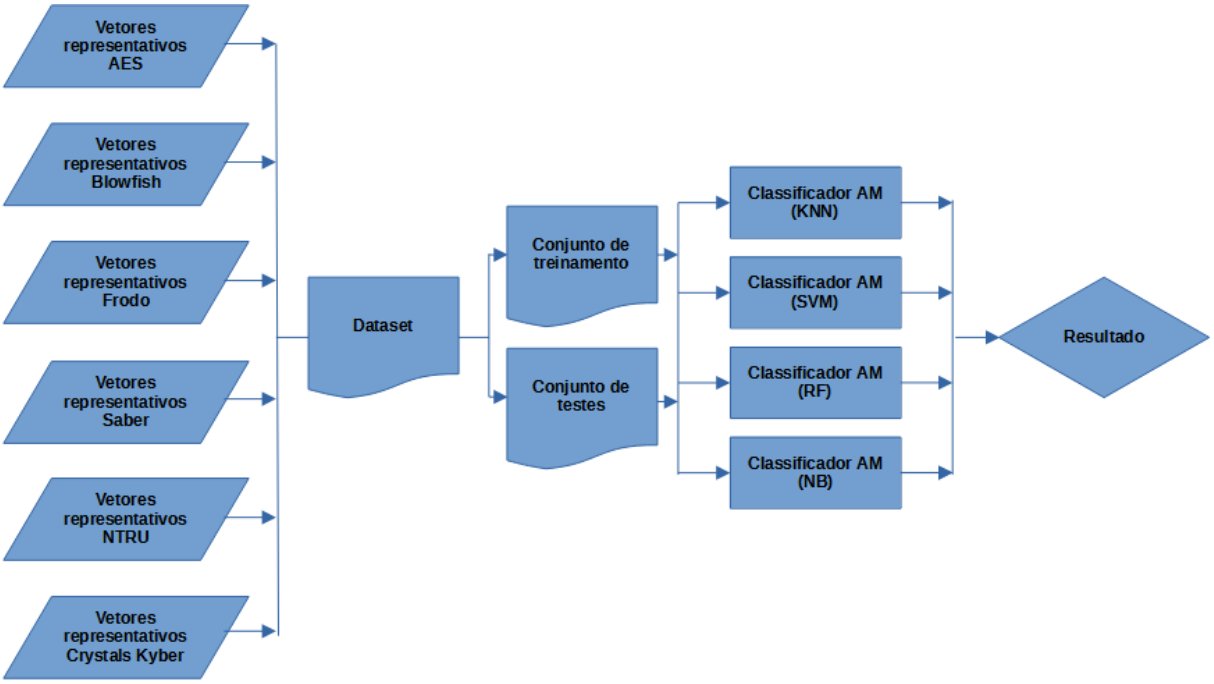


Figura 26 – Esquema de identificação - modo ECB

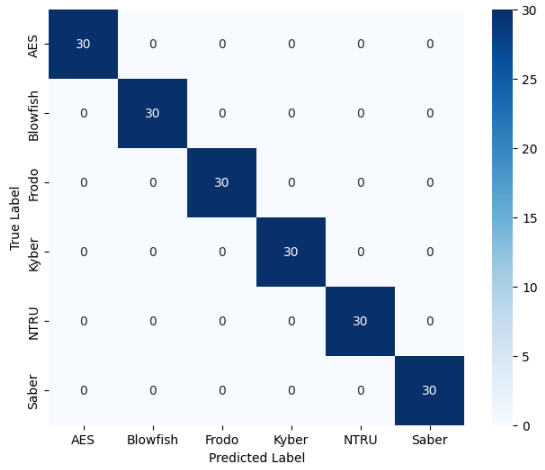


Figura 27 – Matriz de confusão - *train-test split* - KNN - 20KB

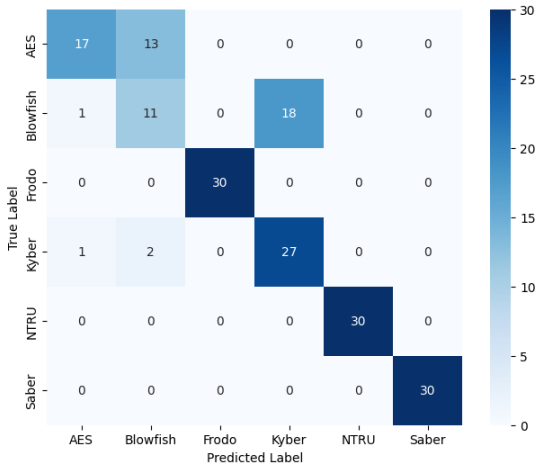


Figura 28 – Matriz de confusão - *train-test split* - NB - 20KB

– O NB classificou corretamente 17 amostras, com uma precisão de 89%, *recall* de 57% e *F1-score* de 69%.

– O SVM identificou corretamente 25 amostras, resultando numa precisão de 71%, *recall* de 83% e *F1-score* de 77%.

– O RF acertou 20 amostras, com uma precisão de 69%, *recall* de 67% e *F1-score* de 68%.

- **Para a classe Blowfish:**

  – O KNN acertou todas as 30 amostras, exibindo uma precisão, *recall* e *F1-score* de 100%.

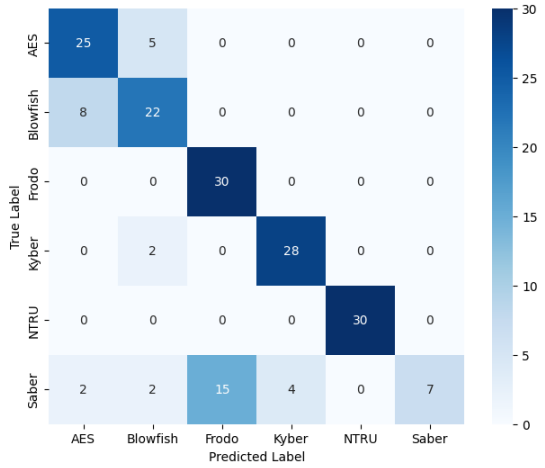


Figura 29 – Matriz de confusão - *train-test split* - SVM - 20KB

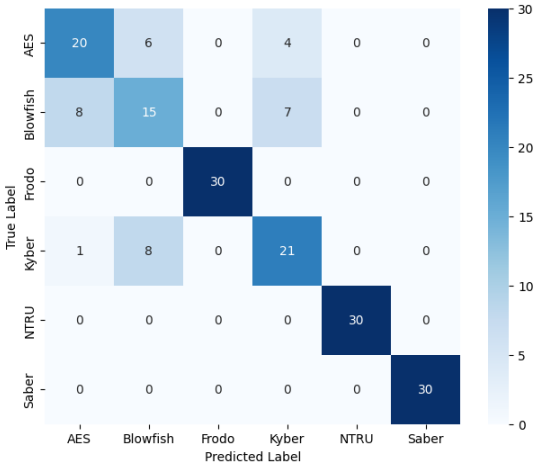


Figura 30 – Matriz de confusão - *train-test split* - RF - 20KB

  - O NB classificou corretamente 11 amostras, com uma precisão de 42%, *recall* de 37% e *F1-score* de 39%.
  - O SVM também classificou corretamente 22 amostras, obtendo uma precisão de 71%, *recall* de 73% e *F1-score* de 72%.
  - O RF acertou 15 amostras, resultando numa precisão de 52%, *recall* de 50% e *F1-score* de 51%.

- **Para as classes Frodo e NTRU:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram as 30 amostras, com precisão, *recall* e *F1-score* de 100%.

- **Para a classe Kyber:**
  - O KNN acertou 30 amostras, alcançando precisão, *recall* e *F1-score* de 100%.
  - O NB acertou 27 amostras, com uma precisão de 100%, *recall* de 90% e *F1-score* de 95%.
  - O SVM identificou corretamente 28 amostras, obtendo uma precisão de 88%, *recall* de 93% e *F1-score* de 90%.
  - O RF acertou 21 amostras, resultando numa precisão de 66%, *recall* de 70% e *F1-score* de 68%.

- **Para a classe Saber:**
  - KNN, NB e RF classificaram corretamente as 30 amostras, com precisão, *recall* e *F1-score* de 100%.
  - O SVM acertou 7 amostras, alcançando um *recall* de 23% e *F1-score* de 38%.

4.1.3 Resultado Experimento ECB *Dataset* 20KB Estratégia *cross-validation*

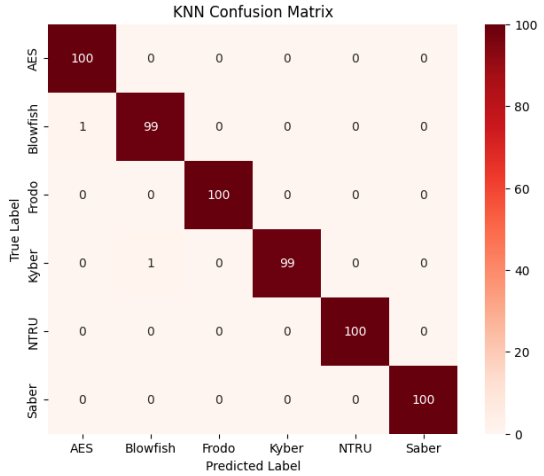


Figura 31 – Matriz de confusão - *cross-validation* - KNN - 20KB

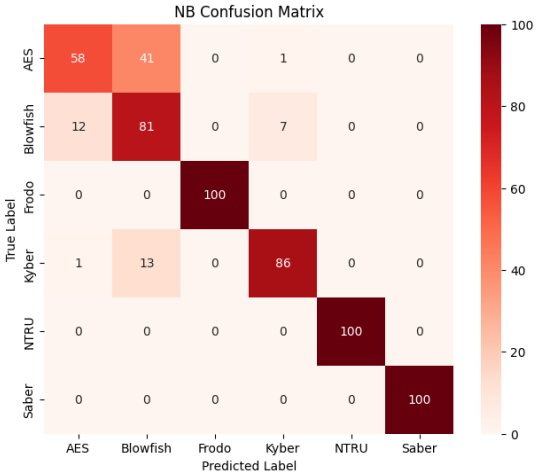


Figura 32 – Matriz de confusão - *cross-validation* - NB - 20KB

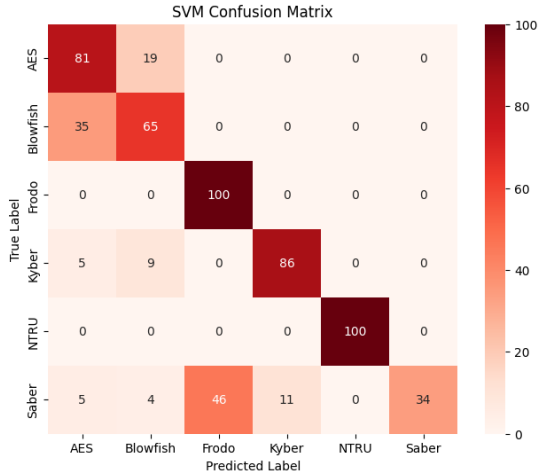


Figura 33 – Matriz de confusão - *cross-validation* - SVM - 20KB

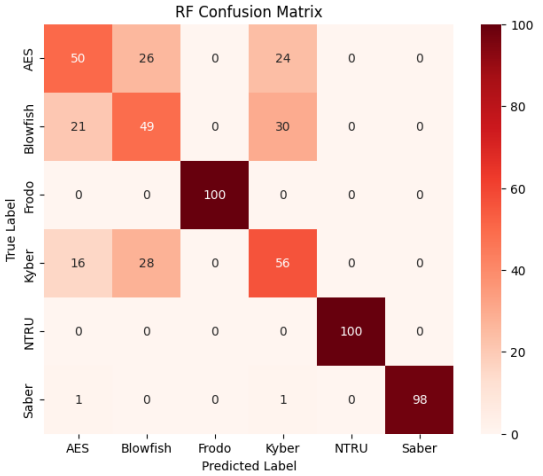


Figura 34 – Matriz de confusão - *cross-validation* - RF - 20KB

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 31, 32, 33 e 34, verifica-se que:

- **Para a classe AES:**
  - KNN classificou todas as 100 amostras corretamente, alcançando uma precisão, *recall* e *F1-score* de 100%.
  - NB acertou 58 amostras, com uma precisão de 82%, *recall* de 58% e *F1-score* de 68%.
  - SVM identificou corretamente 81 amostras, resultando em uma precisão de 64%, *recall* de 81% e *F1-score* de 72%.

  - RF acertou 50 amostras, com uma precisão de 57%, *recall* de 50% e *F1-score* de 53%.

- **Para a classe Blowfish:**

  - KNN classificou 99 amostras corretamente, obtendo precisão, *recall* e *F1-score* de 99%.
  - NB acertou 81 amostras, registrando em uma precisão de 60%, *recall* de 81% e *F1-score* de 69%.
  - SVM identificou corretamente 65 amostras, resultando numa precisão de 67%, *recall* de 65% e *F1-score* de 66%.
  - RF acertou 49 amostras, alcançando uma precisão de 48%, *recall* de 49% e *F1-score* de 48%.

- **Para as classes Frodo, NTRU e Saber:**

  - Todos os classificadores (KNN, NB, SVM e RF) identificaram as 100 amostras corretamente, alcançando precisão, *recall* e *F1-score* de 100%.
  - SVM para a classe Saber acertou 34 amostras, registrando um *recall* de 34% e *F1-score* de 51%.

- **Para a classe Kyber:**

  - KNN acertou 99 amostras, alcançando precisão de 100%, *recall* de 99% e *F1-score* de 99%.
  - NB classificou corretamente 86 amostras, registrando uma precisão de 91%, *recall* de 86% e *F1-score* de 89%.
  - SVM identificou corretamente 86 amostras, obtendo precisão de 89%, *recall* de 86% e *F1-score* de 87%.
  - RF acertou 56 amostras, resultando numa precisão de 50%, *recall* de 56% e *F1-score* de 53%.

4.1.4 Resultado Experimento ECB *Dataset* 60KB Estratégia *train-test split*

Ao analisar as matrizes de confusão geradas pelos classificadores KNN, NB, SVM e RF, apresentadas nas Figuras 35, 36, 37 e 38, observa-se que:

- **Para a classe AES:**

  - O RF classificou corretamente 18 amostras, obtendo precisão, *recall* e *F1-score* de 60%.

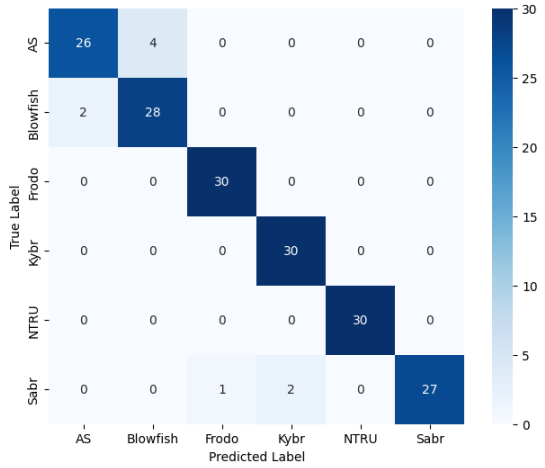


Figura 35 – Matriz de confusão - *train-test split* - KNN - 60KB

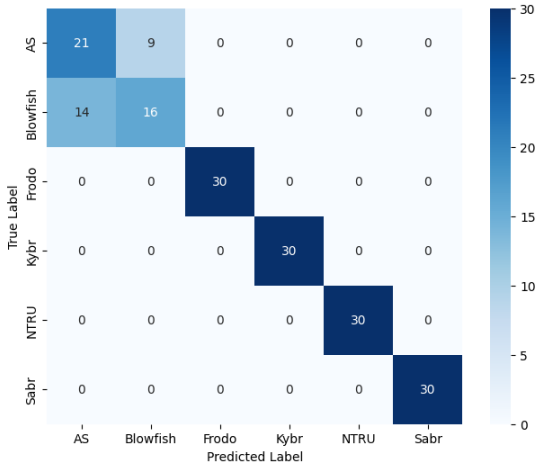


Figura 36 – Matriz de confusão - *train-test split* - NB - 60KB

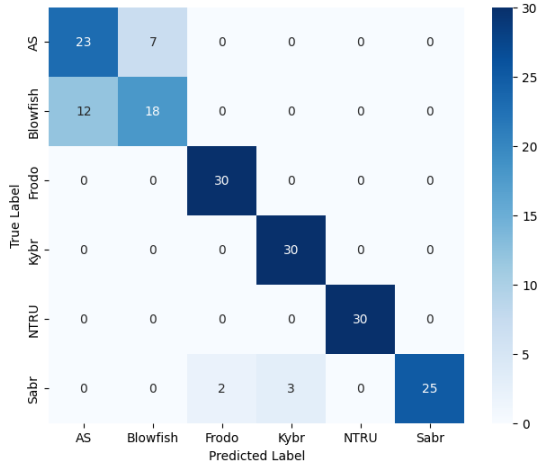


Figura 37 – Matriz de confusão - *train-test split* - SVM - 60KB

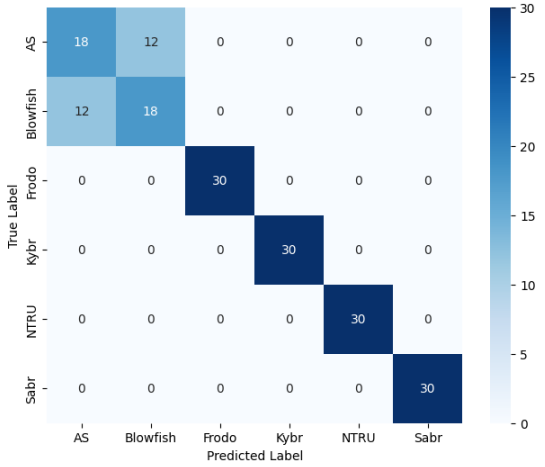


Figura 38 – Matriz de confusão - *train-test split* - RF - 60KB

  - O NB identificou 21 amostras, resultando em uma precisão de 60%, *recall* de 70% e *F1-score* de 65%.
  - O SVM acertou 23 amostras, com uma precisão de 66%, *recall* de 77% e *F1-score* de 71%.
  - O KNN classificou corretamente 26 amostras, registrando uma precisão de 93%, *recall* de 87% e *F1-score* de 90%.

- **Classe Blowfish:**
  - O RF identificou 18 amostras, com uma precisão, *recall* e *F1-score* de 60%.
  - O NB classificou corretamente 16 amostras, resultando em uma precisão de 64%, *recall* de 53% e *F1-score* de 58%.
  - O SVM acertou 18 amostras, alcançando uma precisão de 72%, *recall* de 60% e *F1-score* de 65%.

– O KNN classificou corretamente 28 amostras, obtendo uma precisão de 88%, *recall* de 93% e *F1-score* de 90%.

- **Classes Frodo, Kyber e NTRU:**
  - Todos os classificadores (RF, NB, SVM e KNN) identificaram as 30 amostras, alcançando uma precisão, *recall* e *F1-score* de 100%.
- **Classe Saber:**
  - O RF classificou corretamente as 30 amostras, com uma precisão, *recall* de 100% e *F1-score* de 100%.
  - O NB também identificou as 30 amostras, resultando em uma precisão, *recall* e *F1-score* de 100%.
  - O SVM identificou 25 amostras, alcançando uma precisão de 100%, *recall* de 83% e *F1-score* de 91%.
  - O KNN classificou corretamente 27 amostras, obtendo uma precisão de 100%, *recall* de 90% e *F1-score* de 95%.

4.1.5 Resultado Experimento ECB *Dataset* 60KB Estratégia *cross-validation*

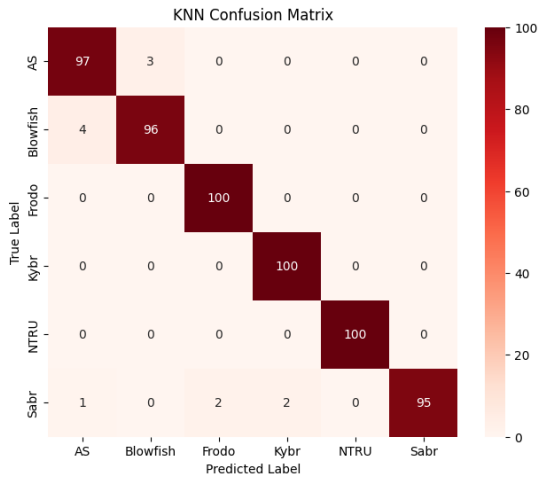


Figura 39 – Matriz de confusão - *cross-validation* - KNN - 60KB

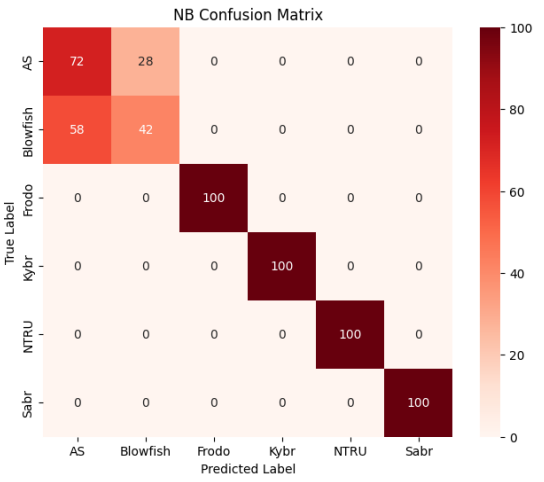


Figura 40 – Matriz de confusão - *cross-validation* - NB - 60KB

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 39, 40, 41 e 42, verifica-se que:

- **Para a classe AES:**
  - KNN classificou 97 amostras corretamente, alcançando uma precisão, *recall* e *F1-score* de 95%, 97% e 96%, respectivamente.

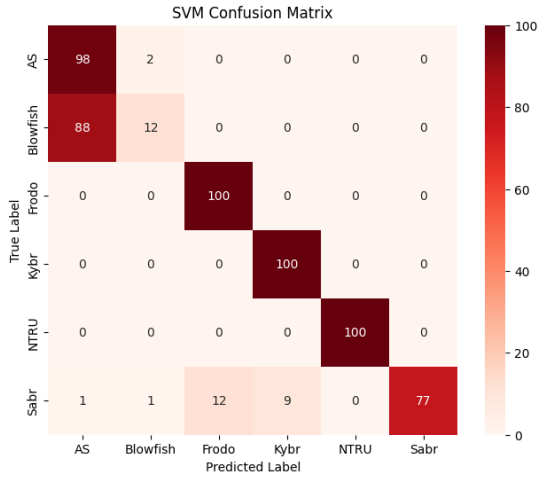


Figura 41 – Matriz de confusão - *cross-validation* - SVM - 60KB

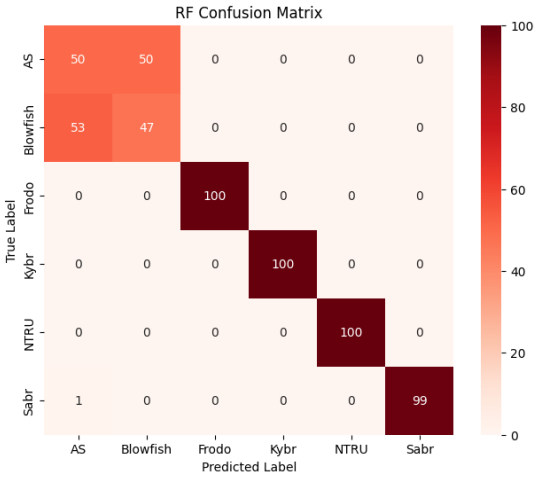


Figura 42 – Matriz de confusão - *cross-validation* - RF - 60KB

  - NB acertou 72 amostras, com uma precisão de 55%, *recall* de 72% e *F1-score* de 63%.
  - SVM identificou corretamente 98 amostras, resultando em uma precisão de 52%, *recall* de 98% e *F1-score* de 68%.
  - RF acertou 50 amostras, com uma precisão de 48%, *recall* de 50% e *F1-score* de 49%.

- **Para a classe Blowfish:**
  - KNN classificou 96 amostras corretamente, obtendo precisão, *recall* e *F1-score* de 97%, 96% e 96%, respectivamente.
  - NB acertou 42 amostras, com uma precisão de 60%, *recall* de 42% e *F1-score* de 49%.
  - SVM identificou corretamente 12 amostras, resultando numa precisão de 80%, *recall* de 12% e *F1-score* de 21%.
  - RF acertou 47 amostras, com uma precisão de 48%, *recall* de 47% e *F1-score* de 48%.

- **Para as classes Frodo, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram as 100 amostras corretamente, com precisão, *recall* e *F1-score* de 100% para as classes Frodo e NTRU, e precisão, *recall* e *F1-score* de 100% para KNN, NB e RF na classe Saber.
  - SVM para a classe Saber acertou 77 amostras, alcançando um *recall* de 77% e *F1-score* de 87%.

- **Para a classe Kyber:**
  - KNN acertou 100 amostras, com precisão de 98%, *recall* de 100% e *F1-score* de 99%.
  - NB classificou corretamente 100 amostras, com uma precisão de 100%, *recall* de 100% e *F1-score* de 100%.
  - SVM identificou corretamente 100 amostras, obtendo precisão de 92%, *recall* de 100% e *F1-score* de 96%.
  - RF acertou 100 amostras, resultando numa precisão de 100%, *recall* de 100% e *F1-score* de 100%.

4.1.6 Resultado Experimento ECB *Dataset* 100KB Estratégia *train-test split*

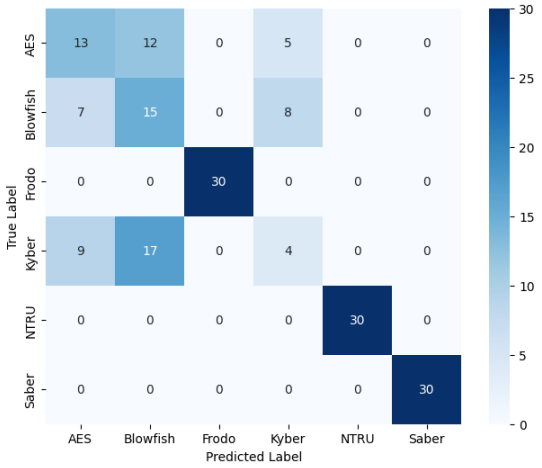


Figura 43 – Matriz de confusão - *train-test split* - KNN - 100KB

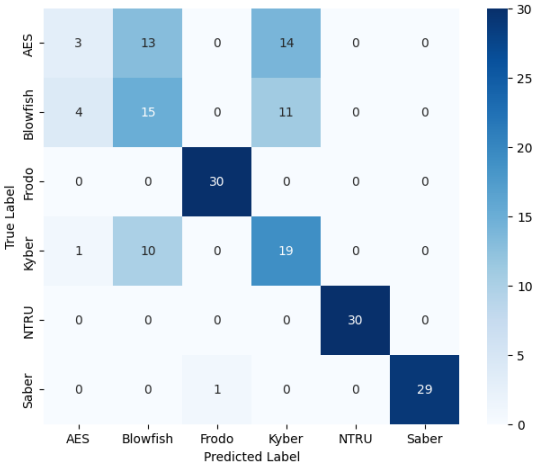


Figura 44 – Matriz de confusão - *train-test split* - SVM - 100KB

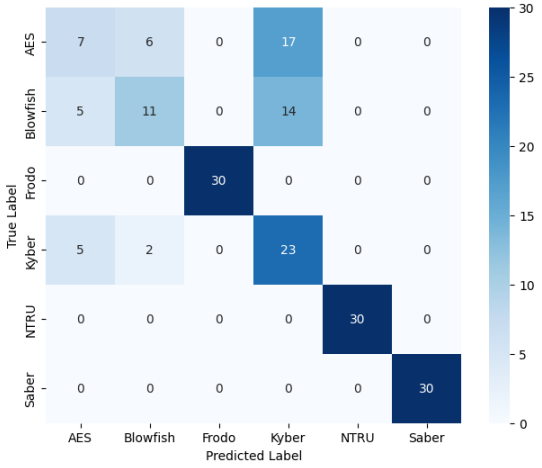


Figura 45 – Matriz de confusão - *train-test split* - NB - 100KB

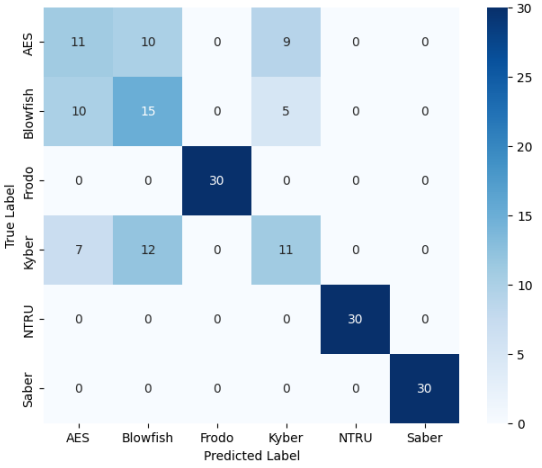


Figura 46 – Matriz de confusão - *train-test split* - RF - 100KB

As matrizes de confusão dos classificadores KNN, NB, SVM e RF, apresentadas respectivamente nas Figuras 43, 45, 44 e 46, revelam que:

- **Para a classe AES:**
  - O classificador KNN foi capaz de classificar corretamente 13 amostras, alcançando uma precisão de 45%, *recall* de 43% e *F1-score* de 44%.
  - O classificador NB acertou 7 amostras, resultando numa precisão de 41%, *recall* de 23% e *F1-score* de 30%.
  - O SVM classificou corretamente 3 amostras, com precisão de 38%, *recall* de 10% e *F1-score* de 16%, enquanto o RF obteve êxito ao classificar 11 amostras corretamente, atingindo uma precisão de 39%, *recall* de 37% e *F1-score* de 38%.
- **Para a classe Blowfish:**
  - KNN, SVM e RF acertaram 15 classificações, enquanto o NB classificou corretamente 11 amostras, resultando numa precisão de 58%, *recall* de 37% e *F1-score* de 45%.
- **Para a classe Kyber:**
  - KNN classificou corretamente 4 amostras, apresentando uma precisão de 24%, *recall* de 13% e *F1-score* de 17%.
  - NB acertou 23 amostras, obtendo uma precisão de 43%, *recall* de 77% e *F1-score* de 55%.
  - SVM também se saiu bem ao classificar corretamente 19 amostras, com uma precisão de 43%, *recall* de 63% e *F1-score* de 51%, enquanto o RF classificou corretamente 11 amostras, resultando numa precisão de 44%, *recall* de 37% e *F1-score* de 40%.
- **Para as classes Frodo, NTRU e Saber:**
  - Todos os classificadores, incluindo KNN, NB, SVM e RF, classificaram corretamente 30 amostras, atingindo 100% de precisão, *recall* e *F1-score*, exceto no SVM que classificou corretamente 29 amostras da classe Saber e obteve 100% de precisão, *recall* de 97% e *F1-score* de 98%.

4.1.7 Resultado Experimento ECB *Dataset* 100KB Estratégia *cross-validation*

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 47, 48, 49 e 50, verifica-se que:

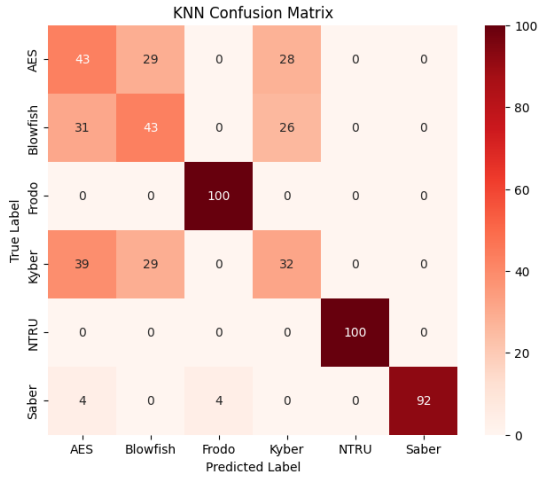


Figura 47 – Matriz de confusão - *cross-validation* - KNN - 100KB

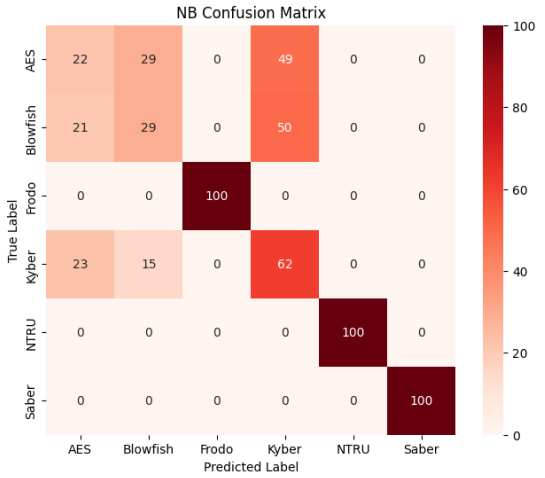


Figura 48 – Matriz de confusão - *cross-validation* - NB - 100KB

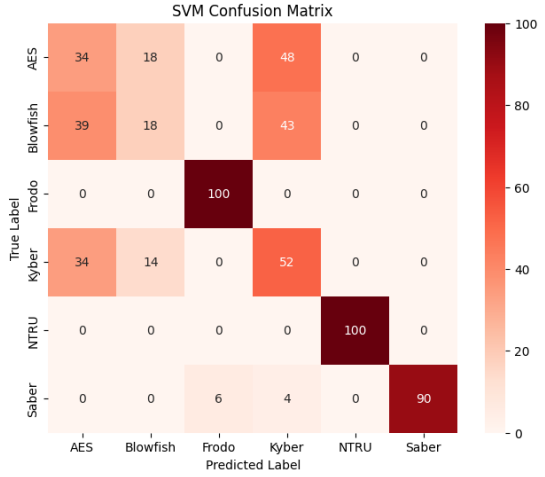


Figura 49 – Matriz de confusão - *cross-validation* - SVM - 100KB

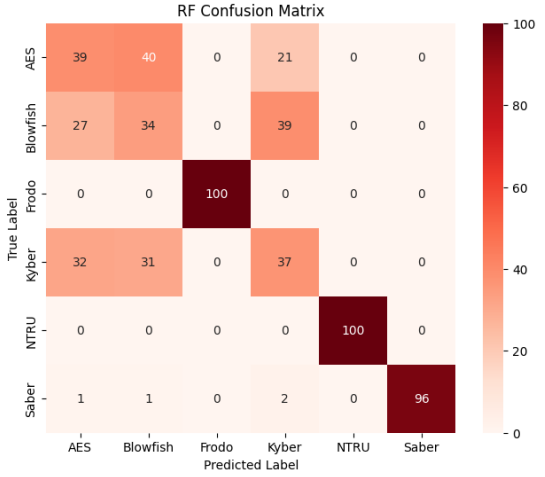


Figura 50 – Matriz de confusão - *cross-validation* - RF - 100KB

- **Para a classe AES:**
  - KNN acertou 43 amostras, alcançando uma precisão, *recall* e *F1-score* de 37%, 43% e 40%, respectivamente.
  - NB classificou corretamente 22 amostras, com uma precisão de 33%, *recall* de 22% e *F1-score* de 27%.
  - SVM identificou corretamente 34 amostras, resultando em uma precisão de 32%, *recall* de 34% e *F1-score* de 33%.
  - RF acertou 39 amostras, com uma precisão de 39%, *recall* de 39% e *F1-score* de 39%.
- **Para a classe Blowfish:**

  - KNN classificou 43 amostras corretamente, obtendo precisão, *recall* e *F1-score* de 43%.
  - NB acertou 29 amostras, com uma precisão de 40%, *recall* de 29% e *F1-score* de 34%.
  - SVM identificou corretamente 18 amostras, resultando numa precisão de 36%, *recall* de 18% e *F1-score* de 24%.
  - RF acertou 34 amostras, com uma precisão de 32%, *recall* de 34% e *F1-score* de 33%.

- **Para as classes Frodo, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram as 100 amostras corretamente, com precisão, *recall* e *F1-score* de 100% para as classes Frodo e NTRU, e precisão, *recall* e *F1-score* de 100% para KNN, NB e RF na classe Saber.
  - SVM para a classe Saber acertou 90 amostras, alcançando um *recall* de 90% e *F1-score* de 95%.

- **Para a classe Kyber:**
  - KNN acertou 32 amostras, com precisão de 37%, *recall* de 32% e *F1-score* de 34%.
  - NB classificou corretamente 62 amostras, com uma precisão de 39%, *recall* de 62% e *F1-score* de 48%.
  - SVM identificou corretamente 52 amostras, obtendo precisão de 35%, *recall* de 52% e *F1-score* de 42%.
  - RF acertou 37 amostras, resultando numa precisão de 37%, *recall* de 37% e *F1-score* de 37%.

4.1.8 Considerações Experimento ECB

4.1.8.1 Considerações sobre os classificadores

O gráfico na Figura 51 ilustra a acurácia dos classificadores KNN, NB, SVM e RF nos *datasets* de 20KB, 60KB e 100KB na estratégia *train-test split*. No conjunto de 20KB, o KNN obteve 100% de acurácia, enquanto NB e RF atingiram 81% e o SVM, 79%. No conjunto de 60KB, o KNN alcançou 95% e os demais modelos melhoraram para 87%. No conjunto de 100KB, o KNN caiu para 68% e os outros ficaram entre 67% e 73%.

Na Figura 52, são apresentadas as acurácias dos mesmos classificadores usando a estratégia *cross-validation*. No conjunto de 20KB, o KNN atingiu 99,67%, enquanto NB e

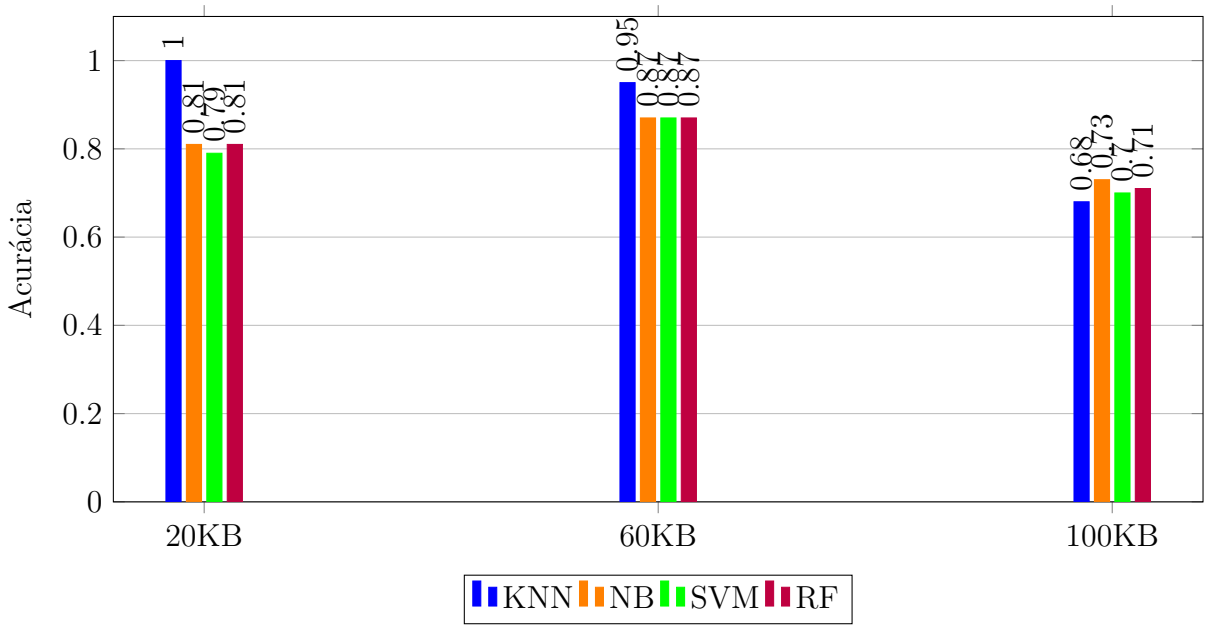


Figura 51 – Acurácia dos modelos por Dataset ECB - Estratégia *train-test split*

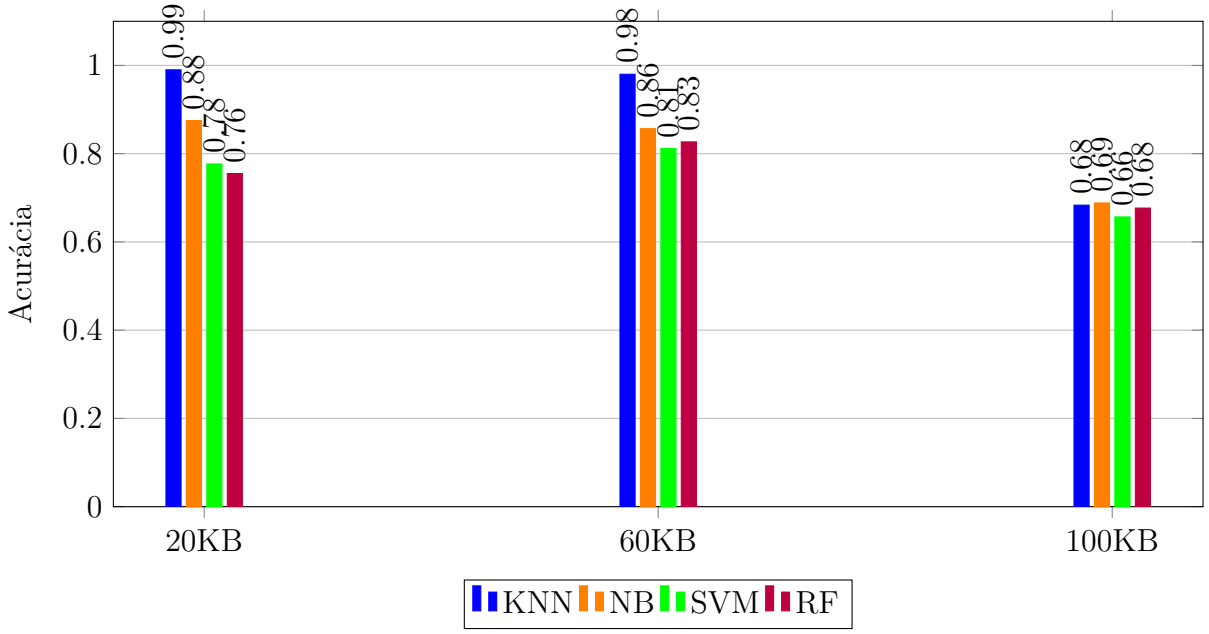


Figura 52 – Acurácia dos modelos por Dataset ECB - Estratégia *cross-validation*

RF registraram 87,50% e o SVM obteve 77,67%. No conjunto de 60KB, o KNN diminuiu para 98% e os outros modelos melhoraram para aproximadamente 86%, 81% e 83%. No conjunto de 100KB, o KNN caiu para 68,33%, enquanto os demais ficaram entre 65,67% e 68,83%.

Considerando os tamanhos das bases de dados analisadas, bem como os resultados obtidos na estratégia de *cross-validation* que validam os alcançados na estratégia de *train-test split*, verifica-se a ausência de vieses na divisão dos conjuntos de treinamento e teste, além de ratificar a estabilidade dos modelos preditivos utilizados durante o experimento.

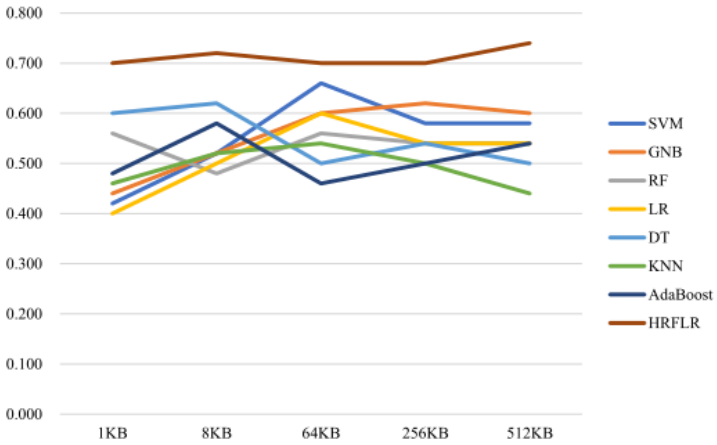
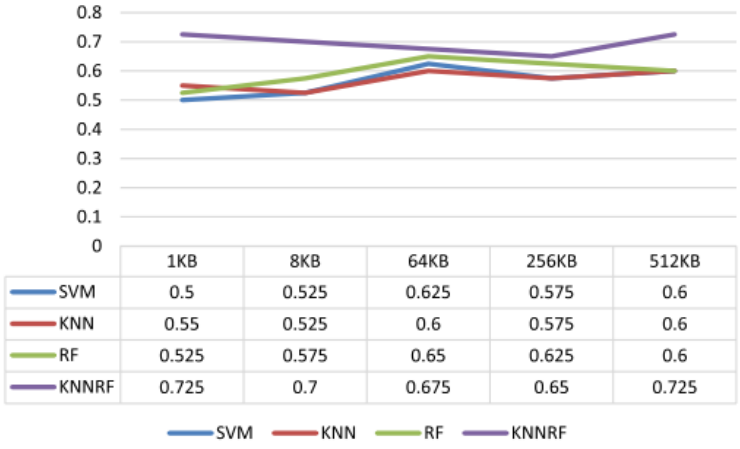


Fig. 7 The binary classification accuracy of the eight classifiers in ciphertext files with sizes from 1 KB to 512 KB

Figura 53 – Acurácia em *datasets* de diferentes tamanhos. Fonte: (11)



	1KB	8KB	64KB	256KB	512KB
SVM	0.5	0.525	0.625	0.575	0.6
KNN	0.55	0.525	0.6	0.575	0.6
RF	0.525	0.575	0.65	0.625	0.6
KNNRF	0.725	0.7	0.675	0.65	0.725

Figure 4 **Identification accuracy under different file sizes.**
Full-size DOI: 10.7717/peerj-cs.1110/fig-4

Figura 54 – Acurácia em *datasets* de diferentes tamanhos. Fonte: (12)

Esses resultados indicam que o KNN alcançou acurácia máxima (100%) e se destacou como o classificador mais eficaz. NB, SVM e RF mostraram desempenho mais consistente e alcançaram melhores taxas de identificação no conjunto de 60KB. Em ambas estratégias, todos os classificadores apresentaram quedas no *dataset* de 100KB, indicando que o aumento no tamanho dos dados nem sempre resulta em uma melhor identificação de criptossistemas. Esses resultados corroboram pesquisas anteriores (11, 12), como evidenciado nas Figuras 53 e 54, que também mostram essa tendência.

O cálculo da variância dos conjuntos de dados usando a função `VarianceThreshold` da biblioteca *Scikit-Learn* revelou que no *dataset* de 20KB, a variância média foi de 6654.23. Para o *dataset* de 60KB, a variância média foi de 13704.71 e para o *dataset* de 100KB, foi de 16456.00. Esses resultados confirmam a tendência observada, indicando que, neste experimento, quanto maior o tamanho do criptograma, maior é a amplitude dos valores p, indicando maior probabilidade de aleatoriedade nos dados.

Em todas as instâncias analisadas, a metodologia proposta obteve taxas de identificação variando de 100% a 65%, superando significativamente o índice aleatório de 16,67%. Essa abordagem foi capaz de identificar padrões binários únicos nos criptogramas, agindo como assinaturas das transformações dos algoritmos criptográficos. Essa presença permitiu que os algoritmos de Aprendizado de Máquina identificassem os criptossistemas por meio dos criptogramas, viabilizando o ataque de distinção.

4.1.8.2 Considerações sobre os criptogramas

De acordo com as matrizes de confusão nesta Seção, a maioria das amostras dos algoritmos pós-quânticos foi corretamente classificada e identificada. Em todos os cenários analisados, as taxas de identificação dessas amostras foram significativamente superiores ao acaso. Esses resultados ressaltam a vulnerabilidade dos criptossistemas de chave pública pós-quânticos, que possuem nível de indistinção IND-CPA, a ataques de distinção. A presença de padrões binários distintivos originados dos problemas matemáticos LWE, RLWE e MLWE, utilizados em suas formulações, torna esses algoritmos distinguíveis entre si. É relevante observar que esses algoritmos apresentaram um grau de indistinção inferior ao das cifras clássicas examinadas, as quais mostraram taxas de identificação inferiores.

As matrizes de confusão obtidas no experimento revelam que, em determinados cenários, os classificadores cometeram erros ao classificar amostras do Kyber, havendo uma tendência de confusão entre amostras do Kyber e amostras do AES e Blowfish, principalmente no *dataset* de 100KB. Ao analisar a estrutura do `Kyber.CPA.Enc`, apresentada no Capítulo 2, verifica-se que uma parte significativa da saída encriptada $c = (u, v)$ - onde $u := \text{Compress}_q(A^T r + e_1, d_u)$ e $v := \text{Compress}_q(t^T r + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v)$ - deriva da matriz aleatória A, do vetor r, e dos vetores de erro e_1 e e_2, todos gerados de forma aleatória.

Portanto, há a probabilidade de que os criptogramas provenientes desse algoritmo apresentem um maior grau de aleatoriedade devido a esses componentes aleatórios que compõem a saída c, dificultando assim a identificação de suas amostras em relação às demais pós-quânticas.

Apesar de serem diferentes dos criptossistemas estudados em pesquisas anteriores que utilizaram a NIST STS para criar vetores representativos, os criptossistemas analisados neste estudo apresentaram resultados compatíveis com pesquisas conduzidas por autores como Xia et al. (2020), Yuan et al. (2022) e Ayuan et al. (2022). Isso indica que os valores p refletem características estatísticas únicas dos criptossistemas, oferecendo uma maneira eficaz de caracterizá-los e distinguí-los. Esses valores evidenciam a presença de padrões nos criptogramas que podem ser explorados por técnicas criptoanalíticas.

A utilização dos 15 testes estatísticos da NIST STS, em contraste com métodos anteriores que usaram 10 ou 3 testes, teve um impacto positivo nos resultados. Os vetores representativos conseguiram capturar de forma mais adequada as características dos

criptogramas, resultando em uma melhoria significativa nas classificações.

4.2 Experimento - Modo CBC

4.2.1 Organização do experimento

De acordo com Stallings, Bressan e Barbosa(117), o modo de operação CBC, ilustrado no Figura 7, foi desenvolvido para mitigar vulnerabilidades de segurança do modo ECB. Ao utilizar o encadeamento dos blocos cifrados, a entrada da função de encriptação para cada bloco de texto claro não mantém um relacionamento fixo com o texto claro original e dessa forma, padrões repetitivos binários não são revelados.

Em consequência do encadeamento de blocos, várias pesquisas de identificação de padrões em criptogramas obtiveram resultados ligeiramente superiores ao índice aleatório. Tan, Deng e Zhang(104) alcançaram uma acurácia de 38% ao analisar amostras de 500 KB e 100 KB, geradas por chaves e IVs aleatórios, com um índice aleatório de 20% (AES, DES, 3DES, RC5 e Blowfish). Fan e Zhao(108) conseguiram 13.5% de acurácia com um índice aleatório de 12.5% (DES, 3DES, AES-128, AES-256, IDEA, SMS, Blowfish e Camellia-128) em amostras de 512KB. Hu e Zhao(109) obtiveram uma média de 12.64% de acurácia, ligeiramente superior ao índice aleatório de 12.5% (AES-128, AES-256, Blowfish-64, Camellia-128, DES-56, 3DES-56, IDEA-64 e SMS4-128) em um conjunto de amostras de 512KB. Utilizando a rede neural MultiLayer Perceptron (MLP), Yu e Shi(114) conseguiram 29.8% de acurácia, superando a probabilidade aleatória de 25% (DES, AES, 3DES e Blowfish) em amostras de 256 KB. Os resultados dessas pesquisas evidenciam que não foram identificados padrões nos criptogramas analisados, indicando a eficácia da eliminação de padrões binários no modo CBC.

Conduziu-se um experimento a fim de identificar os criptossistemas no modo CBC, considerando a eliminação de padrões binários decorrente do encadeamento de blocos. Semelhante ao experimento anterior, neste experimento também foram analisados *datasets* de 20KB, 60KB e 100KB.

Tomando como base o método apresentado por Rocha, Xexeo e Torres(126) e seguindo a metodologia ilustrada nas Figuras 55 e 56, extratos da base de dados foram criptografados pelos algoritmos AES, Blowfish, Frodo, Kyber, Saber e NTRU, com chaves e IVs aleatórios e não repetidos aplicados a cada extrato, resultando em vários textos cifrados. Essa abordagem foi adotada para eliminar vieses provenientes das chaves e IVs. o primeiro bloco produzido de cada texto cifrado gerado foi selecionado para análise.

Esses primeiros blocos, correspondentes a cada um dos cifradores analisados, foram concatenados em um único arquivo. Esse procedimento foi replicado 100 vezes para cada criptossistema analisado, gerando 100 diferentes arquivos de primeiros blocos concatenados

para cada algoritmo. Semelhante ao primeiro experimento, destaca-se que a mesma quantidade de amostras foi aplicada a todos os algoritmos criptográficos avaliados, garantindo assim uma comparação justa e eliminando possíveis viéses.

Em seguida, os 600 arquivos dos primeiros blocos concatenados foram submetidos a NIST STS, e os 41 valores p provenientes dos resultados dos 15 testes estatísticos, bem como a quantidade de bits 0s e 1s, foram utilizados para formar os vetores representativos, de modo que cada arquivo dos primeiros blocos concatenados foi representado por um vetor contendo 43 campos.

Similar ao primeiro experimento, para cada um dos três *datasets* do experimento, contendo 600 vetores representativos (100 para cada criptossistema), também foram empregadas as estratégias *train-test split* e *cross-validation*. Em um primeiro momento, adotou-se a estratégia *train-test split* e dividiu-se o conjunto de dados em dois grupos: um de treinamento, representando 70% das amostras totais (420 amostras) e outro de teste, com os 30% restantes (180 amostras), de acordo com as conclusões de Stapor et al.(122), que apontam para um equilíbrio entre viés e variância, indicando proporções próximas a 66% e 33% entre os conjuntos de treinamento e teste. Em uma segundo momento, para cada um dos três *datasets*, avaliou-se a estratégia *cross-validation* com o valor de K igual a 10.

É importante considerar que o método apresentado por Rocha, Xexeo e Torres(126), ao eliminar o encadeamento de blocos do modo CBC e selecionar apenas os primeiros blocos dos criptogramas, assemelha a análise da coleção de criptogramas gerados no modo CBC a uma análise da coleção de criptogramas gerados no modo ECB, de modo que existe a probabilidade dos resultados serem superiores ou semelhantes aos obtidos no experimento anterior. Outrossim, conforme apresentado no Capítulo 2, devido a essas construções distintas dos algoritmos pós-quânticos (Crystals-Kyber (*Module Learning With Errors* - MLWE), Saber (*Learning With Round* - MLWR) e Frodo (*Learning With Errors* - LWE)) que resulta em características únicas dos dados de entrada, existe a probabilidade de que os arquivos de primeiros blocos concatenados provenientes desses algoritmos também apresentem diferenças estatísticas, de modo que suas amostras sejam identificáveis por meio dos padrões binários existentes.

4.2.2 Resultado Experimento CBC *Dataset* 20KB Estratégia *train-test split*

Com base nas matrizes de confusão dos classificadores KNN, NB, SVM e RF, ilustradas nas Figuras 57, 58, 59 e 60, respectivamente, observa-se que:

- **Para a classe AES:**
  - O KNN apresentou um desempenho ótimo, identificando corretamente todas as

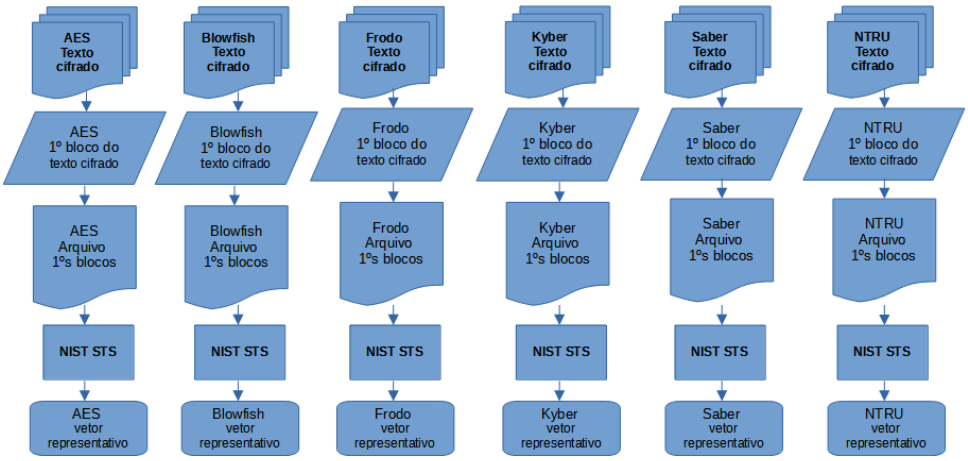


Figura 55 – Construção dos vetores representativos - modo CBC

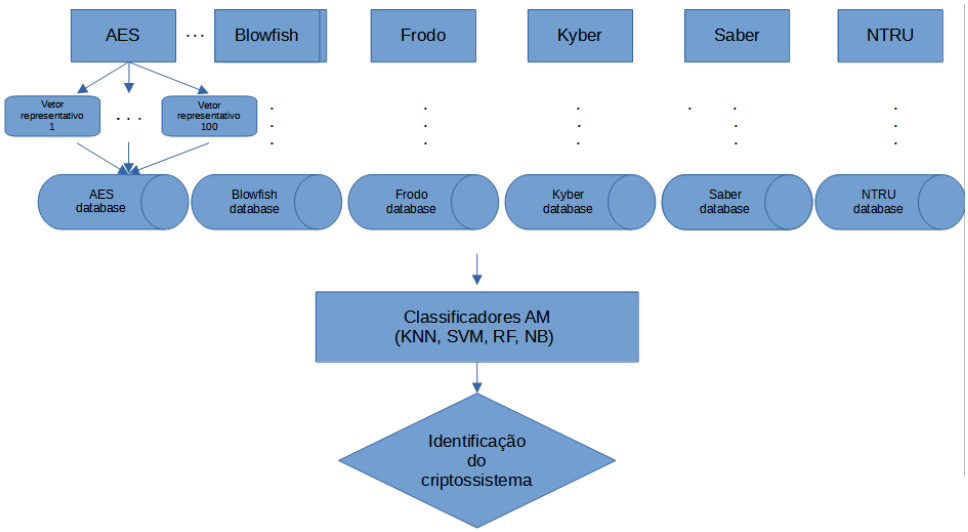


Figura 56 – Esquema de identificação - modo CBC

30 amostras, alcançando uma precisão, *recall* e *F1-score* de 100%.

– O NB classificou 29 amostras corretamente, com uma precisão, *recall* e *F1-score* de 97%.

– O RF classificou 16 amostras corretamente, resultando em uma precisão de 64%, *recall* de 53% e *F1-score* de 58%.

- **Para a classe Blowfish:**

  – O KNN classificou corretamente as 30 amostras e alcançou *recall* e *F1-score* de 100%.

  – O NB e SVM tiveram um desempenho oposto, classificando 6 e 27 amostras corretamente, respectivamente, com uma precisão de 86%, *recall* de 20% e *F1-score* de 32% para o NB e precisão de 75%, *recall* de 90% e *F1-score* de 82% para o SVM.

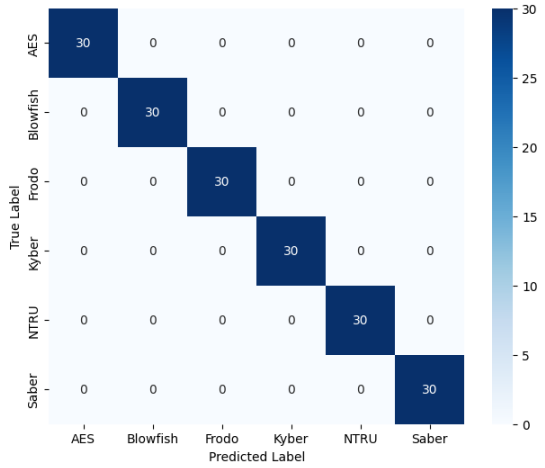


Figura 57 – Matriz de confusão - KNN - 20KB

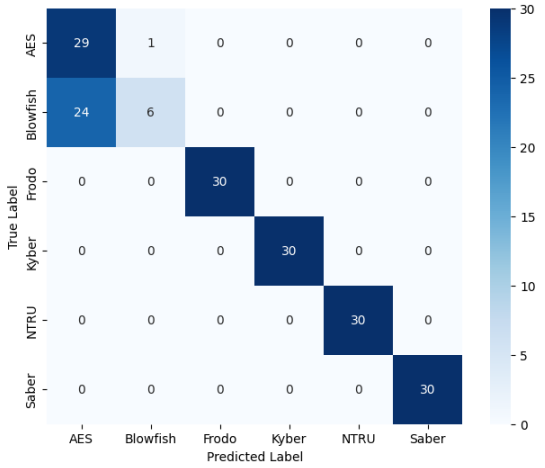


Figura 58 – Matriz de confusão - NB - 20KB

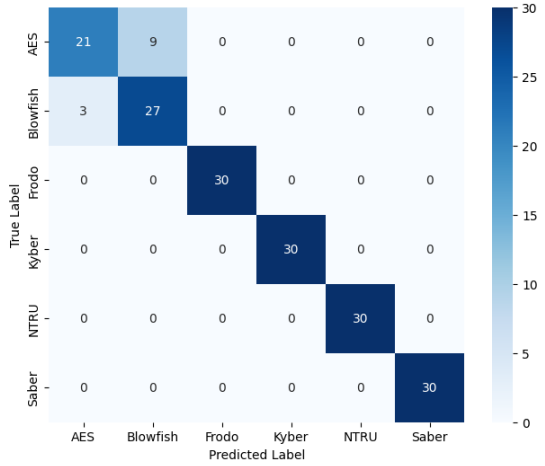


Figura 59 – Matriz de confusão - SVM - 20KB

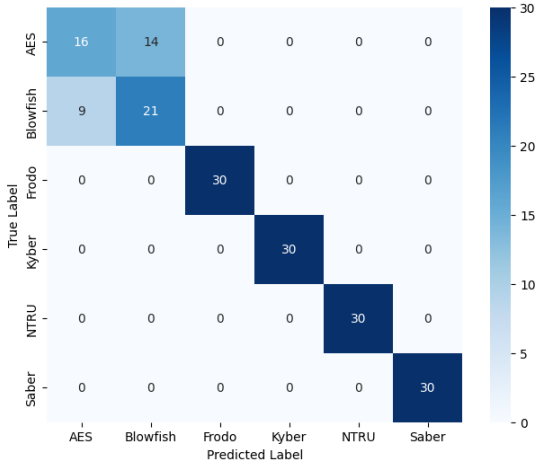


Figura 60 – Matriz de confusão - RF - 20KB

  - O RF também apresentou resultados semelhantes ao SVM nesta classe, classificando corretamente 21 amostras e obtendo precisão, *recall* e *F1-score* de 60%, 70% e 65%, respectivamente.

- **Para as classes Frodo, Kyber, NTRU e Saber:**

  - Todos os classificadores - KNN, NB, SVM e RF - identificaram corretamente todas as 30 amostras e alcançaram uma precisão, *recall* e *F1-score* de 100%.

4.2.3 Resultado Experimento CBC *Dataset* 20KB Estratégia *cross-validation*

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 61, 62, 63 e 64, verifica-se que:

- **Para a classe AES:**

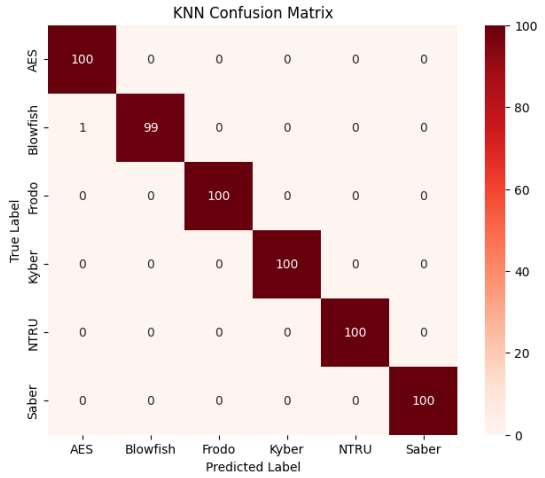


Figura 61 – Matriz de confusão - *cross-validation* - KNN - 20KB

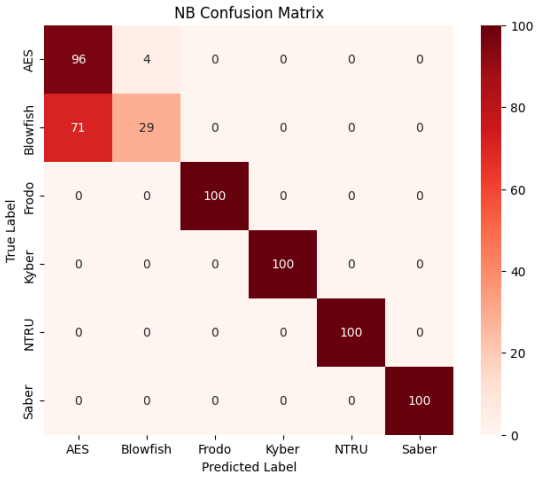


Figura 62 – Matriz de confusão - *cross-validation* - NB - 20KB

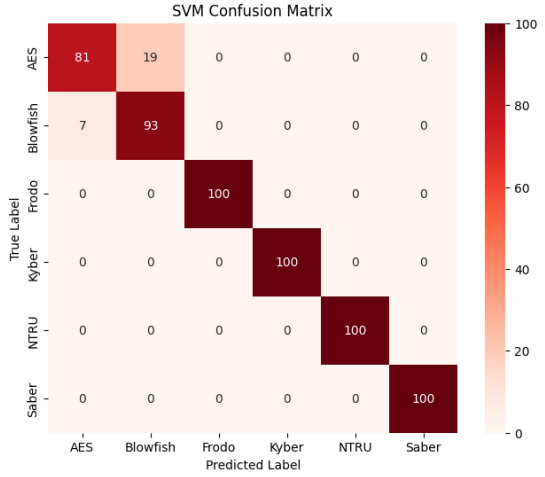


Figura 63 – Matriz de confusão - *cross-validation* - SVM - 20KB

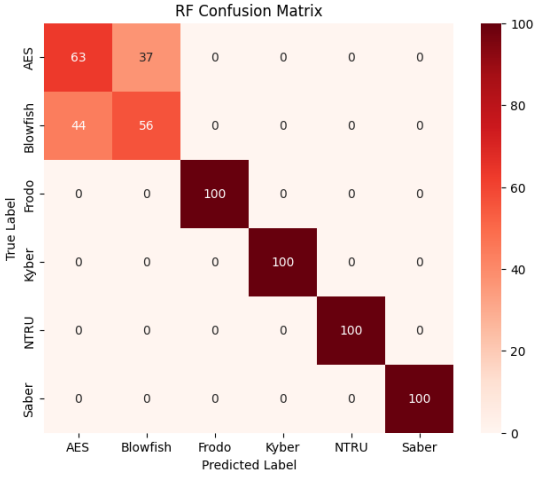


Figura 64 – Matriz de confusão - *cross-validation* - RF - 20KB

  - KNN classificou corretamente 100 amostras, alcançando uma precisão, *recall* e *F1-score* de 100%.
  - NB acertou 96 amostras, resultando em precisão de 57%, *recall* de 96% e *F1-score* de 72%.
  - SVM identificou corretamente 81 amostras, registrando em uma precisão de 92%, *recall* de 81% e *F1-score* de 86%.
  - RF acertou 63 amostras, obtendo precisão de 59%, *recall* de 63% e *F1-score* de 61%.

- **Para a classe Blowfish:**

  - KNN classificou corretamente 99 amostras, obtendo precisão, *recall* e *F1-score* de 100%, 99% e 99%, respectivamente.

- NB acertou 29 amostras, alcançando precisão de 88%, *recall* de 29% e *F1-score* de 44%.
- SVM identificou corretamente 93 amostras, resultando numa precisão de 83%, *recall* de 93% e *F1-score* de 88%.
- RF acertou 56 amostras, registrando precisão de 60%, *recall* de 56% e *F1-score* de 58%.

- **Para as classes Frodo, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram 100 amostras corretamente, resultando em precisão, *recall* e *F1-score* de 100% para todas as classes.

4.2.4 Resultado Experimento CBC *Dataset* 60KB Estratégia *train-test split*

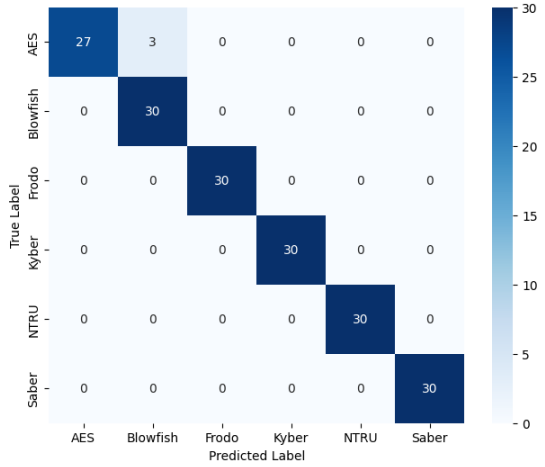


Figura 65 – Matriz de confusão - KNN - 60KB

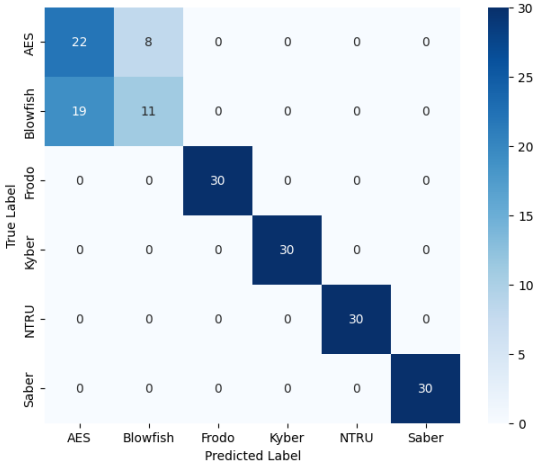


Figura 66 – Matriz de confusão - NB - 60KB

A análise das matrizes de confusão dos classificadores KNN, NB, SVM e RF, ilustradas nas Figuras 65, 66, 67 e 68, respectivamente, revela que:

- **Para a classe AES:**
  - KNN: Classificou corretamente 27 amostras, obtendo precisão de 100%, *recall* de 90% e *F1-score* de 95%.
  - NB: Classificou 22 amostras corretamente, alcançando precisão de 53%, *recall* de 73% e *F1-score* de 62%.
  - SVM: Acertou 13 amostras, obtendo uma precisão de 68%, *recall* de 43% e *F1-score* de 53%.

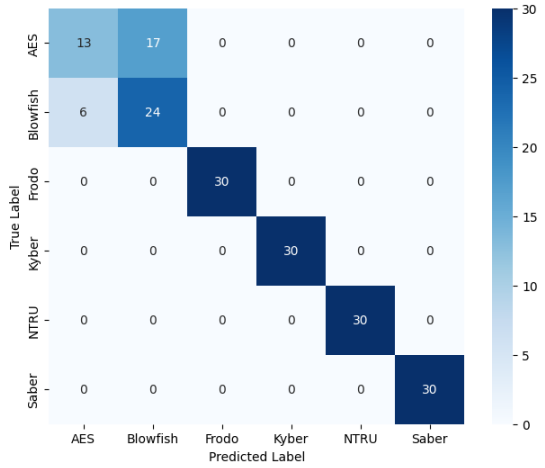


Figura 67 – Matriz de confusão - SVM - 60KB

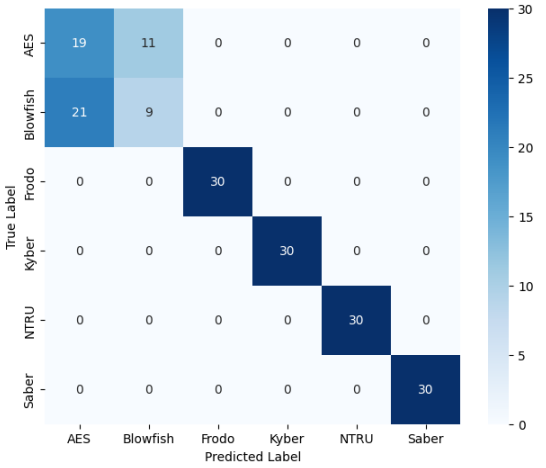


Figura 68 – Matriz de confusão - RF - 60KB

  - RF: Classificou 19 amostras corretamente, resultando em precisão de 47%, *recall* de 63% e *F1-score* de 54%.

- **Para a classe Blowfish:**
  - KNN: Classificou todas as 30 amostras corretamente, registrando em 100% de *recall* e *F1-score*.
  - NB: Classificou 11 amostras corretamente, resultando em 58% de precisão, *recall* de 37% e *F1-score* de 45%.
  - SVM: Classificou 24 amostras corretamente, alcançando 59% de precisão, *recall* de 80% e *F1-score* de 68%.
  - RF: Também classificou 11 amostras corretamente, resultando em uma precisão de 45%, *recall* de 30% e *F1-score* de 36%.

- **Para as classes Frodo, Kyber, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) identificaram corretamente todas as 30 amostras e obtiveram 100% de precisão, *recall* e *F1-score*.

4.2.5 Resultado Experimento CBC *Dataset* 60KB Estratégia *cross-validation*

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 69, 70, 71 e 72, verifica-se que:

- **Para a classe AES:**
  - KNN classificou corretamente 95 amostras e obteve precisão de 98%, *recall* de 95% e *F1-score* de 96%.

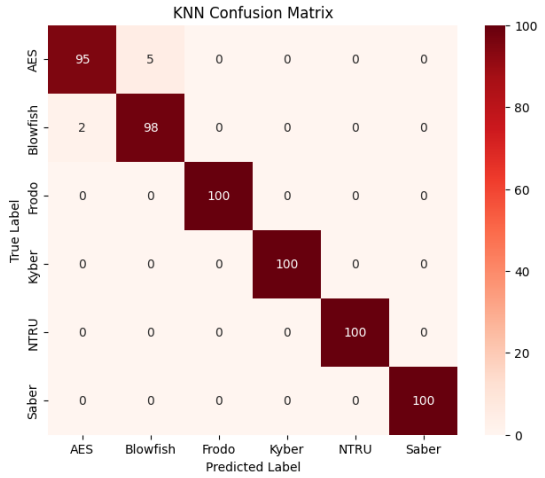


Figura 69 – Matriz de confusão - *cross-validation* - KNN - 60KB

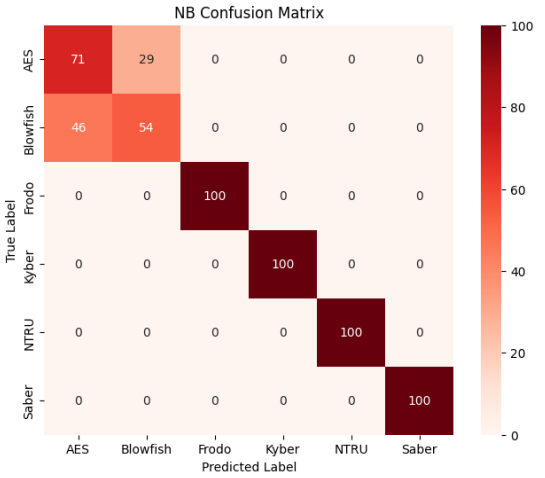


Figura 70 – Matriz de confusão - *cross-validation* - NB - 60KB

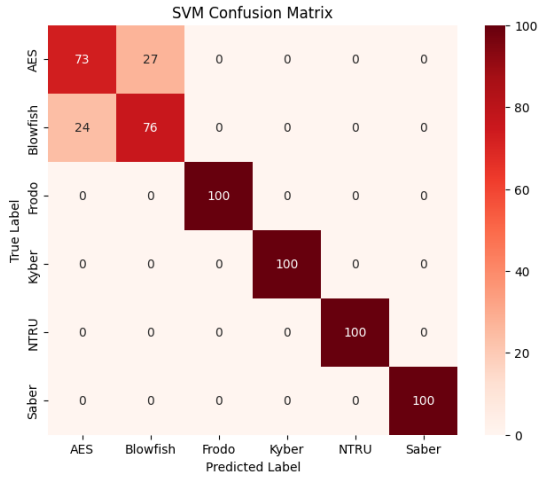


Figura 71 – Matriz de confusão - *cross-validation* - SVM - 60KB

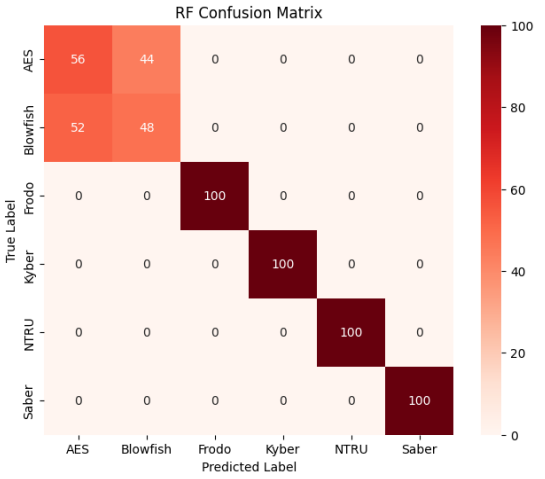


Figura 72 – Matriz de confusão - *cross-validation* - RF - 60KB

  - NB acertou 71 amostras, alcançando uma precisão de 61%, *recall* de 71% e *F1-score* de 65%.
  - SVM identificou corretamente 73 amostras, obtendo uma precisão de 75%, *recall* de 73% e *F1-score* de 74%.
  - RF acertou 56 amostras, resultando em uma precisão de 52%, *recall* de 56% e *F1-score* de 54%.

- **Para a classe Blowfish:**
  - KNN classificou corretamente 98 amostras, registrando uma precisão de 95%, *recall* de 98% e *F1-score* de 97%.
  - NB acertou 54 amostras, alcançando uma precisão de 65%, *recall* de 54% e *F1-score* de 59%.

  - SVM identificou corretamente 76 amostras, resultando numa precisão de 74%, *recall* de 76% e *F1-score* de 75%.
  - RF acertou 48 amostras e obteve precisão de 52%, *recall* de 48% e *F1-score* de 50%.

- **Para as classes Frodo, Kyber, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram 100 amostras corretamente, registrando precisão, *recall* e *F1-score* de 100% para todas as classes.

4.2.6 Resultado Experimento CBC *Dataset* 100KB Estratégia *train-test split*

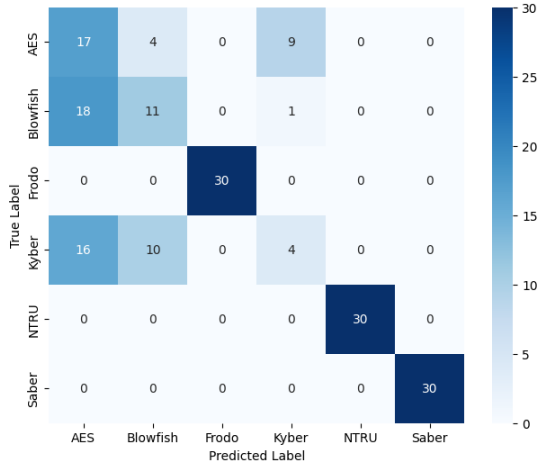


Figura 73 – Matriz de confusão - KNN - 100KB

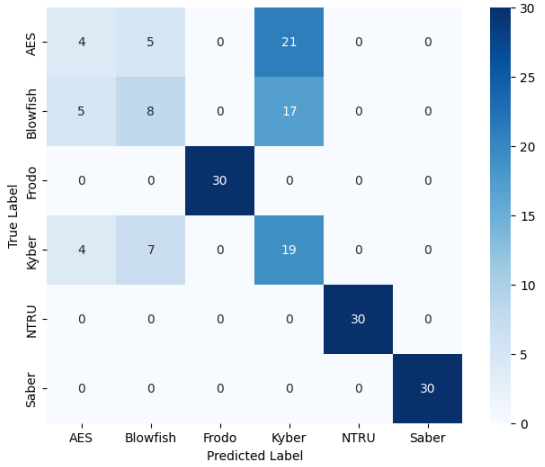


Figura 74 – Matriz de confusão - NB - 100KB

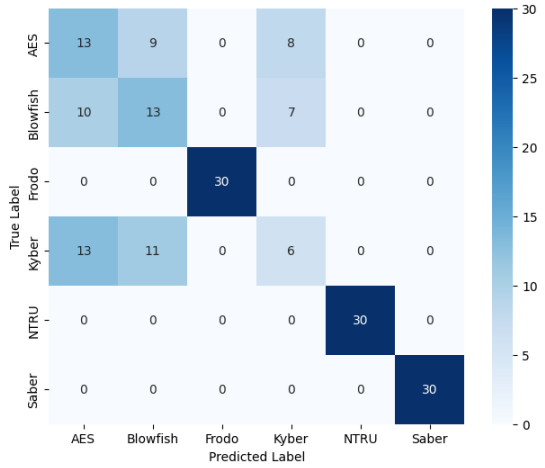


Figura 75 – Matriz de confusão - SVM - 100KB

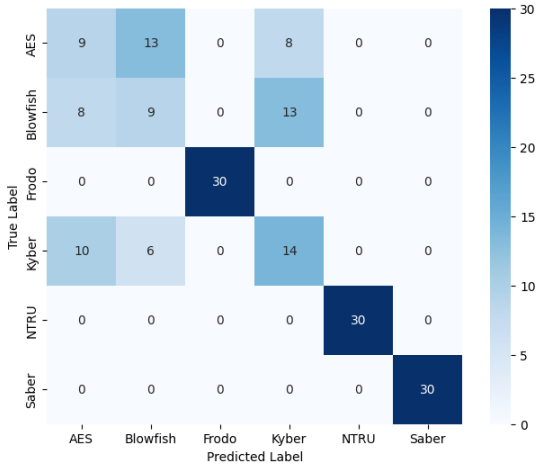


Figura 76 – Matriz de confusão - RF - 100KB

Analisando os gráficos de confusão dos classificadores KNN, NB, SVM e RF, mostrados nas Figuras 73, 74, 75 e 76, respectivamente, pode-se observar que:

- **Para a classe AES:**
  - O KNN classificou corretamente 17 amostras, alcançando precisão de 33%, *recall* de 57% e *F1-score* de 42%.
  - O NB identificou 4 amostras, registrando precisão de 31%, *recall* de 13% e *F1-score* de 19%.
  - O SVM efetuou a classificação correta em 13 amostras e obteve com precisão de 36% e *F1-score* de 39%.
  - O RF classificou corretamente 9 amostras, resultando em um *recall* de 30% e *F1-score* de 32%.
- **Para a classe Blowfish:**
  - O KNN efetuou a classificação correta de 11 amostras, alcançou precisão de 44%, *recall* de 37% e *F1-score* de 40%.
  - O NB identificou 8 amostras e obteve precisão de 40%, *recall* de 27% e *F1-score* de 32%.
  - O SVM efetuou a classificação correta de 13 amostras, registrando em umam precisão de 39%, *recall* de 43% e *F1-score* de 41%.
  - O RF identificou 9 amostras, resultando em um *recall* de 30% e *F1-score* de 31%.
- **Para a classe Kyber:**
  - O KNN efetuou a classificação precisa de 4 amostras, resultando em uma precisão de 29%, *recall* de 13% e *F1-score* de 18%.
  - O NB identificou 19 amostras e obteve precisão de 33% e *F1-score* de 44%.
  - O SVM efetuou a classificação correta em 6 amostras e alcançou precisão de 29%, *recall* de 20% e *F1-score* de 24%.
  - O RF conduziu a classificação precisa de 14 amostras, obtendo precisão de 40%.
- **Para as classes Frodo, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) classificaram corretamente todas as 30 amostras e alcançaram 100% de precisão, *recall* e *F1-score*.

4.2.7 Resultado Experimento CBC *Dataset* 100KB Estratégia *cross-validation*

Ao analisar as matrizes de confusão dos classificadores KNN, NB, SVM e RF, representadas nas Figuras 77, 78, 79 e 80, verifica-se que:

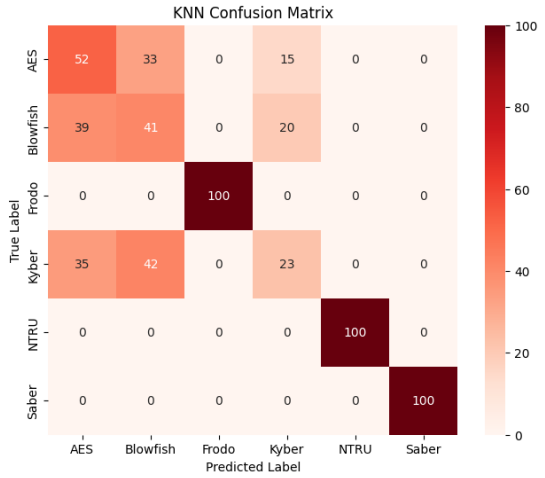


Figura 77 – Matriz de confusão - *cross-validation* - KNN - 100KB

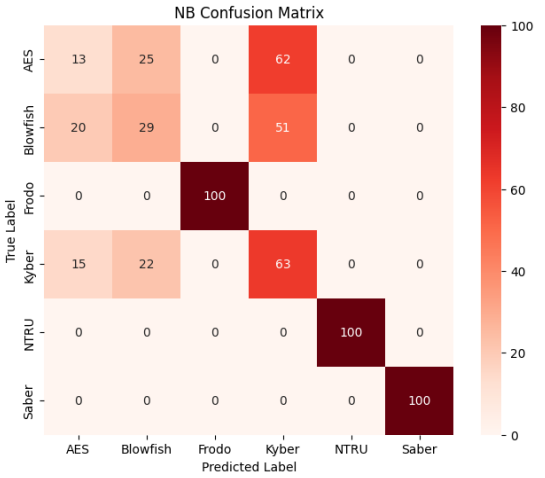


Figura 78 – Matriz de confusão - *cross-validation* - NB - 100KB

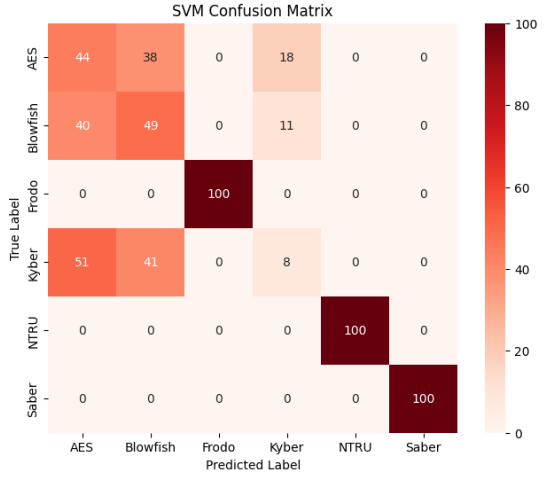


Figura 79 – Matriz de confusão - *cross-validation* - SVM - 100KB

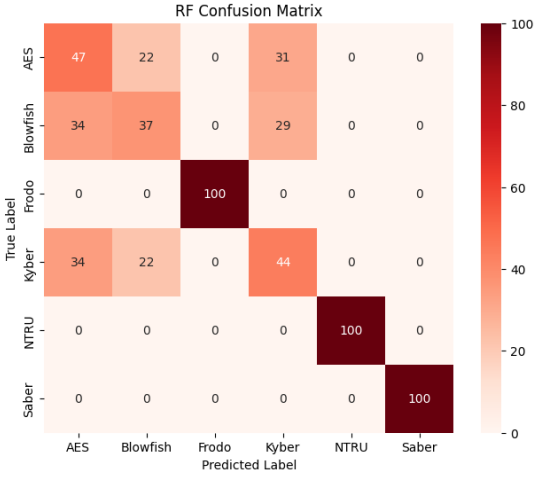


Figura 80 – Matriz de confusão - *cross-validation* - RF - 100KB

- **Para a classe AES:**
  - KNN classificou corretamente 52 amostras, alcançando precisão de 41%, *recall* de 52% e *F1-score* de 46%.
  - NB acertou 13 amostras, registrando precisão de 27%, *recall* de 13% e *F1-score* de 18%.
  - SVM identificou corretamente 44 amostras, obtendo uma precisão de 33%, *recall* de 44% e *F1-score* de 37%.
  - RF acertou 47 amostras, resultando em uma precisão de 41%, *recall* de 47% e *F1-score* de 44%.
- **Para a classe Blowfish:**

  - KNN classificou corretamente 41 amostras, obtendo precisão de 35%, *recall* de 41% e *F1-score* de 38%.
  - NB acertou 29 amostras, alcançando uma precisão de 38%, *recall* de 29% e *F1-score* de 33%.
  - SVM identificou corretamente 49 amostras, resultando numa precisão de 38%, *recall* de 49% e *F1-score* de 43%.
  - RF acertou 37 amostras, registrando precisão de 46%, *recall* de 37% e *F1-score* de 41%.

- **Para a classe Kyber:**
  - KNN classificou corretamente 23 amostras e obteve uma precisão de 40%, *recall* de 23% e *F1-score* de 29%.
  - NB acertou 63 amostras, resultando em uma precisão de 36%, *recall* de 63% e *F1-score* de 46%.
  - SVM identificou corretamente 8 amostras, registrando precisão de 22%, *recall* de 8% e *F1-score* de 12%.
  - RF acertou 44 amostras e alcançou uma precisão de 42%, *recall* de 44% e *F1-score* de 43%.

- **Para as classes Frodo, Kyber, NTRU e Saber:**
  - Todos os classificadores (KNN, NB, SVM e RF) acertaram as 100 amostras corretamente e obtiveram precisão, *recall* e *F1-score* de 100% para todas as classes.

4.2.8 Considerações Experimento CBC

4.2.8.1 Considerações sobre os classificadores

O gráfico apresentado na Figura 81 ilustra a acurácia dos classificadores KNN, NB, SVM e RF nos *datasets* de 20KB, 60KB e 100KB usando a estratégia *train-test split*. No conjunto de 20KB, o KNN alcançou a precisão ótima de 100%, enquanto NB e RF obtiveram 86 e 93%%, enquanto o SVM registrou 87%. No conjunto de 60KB, o KNN manteve um desempenho alto com acurácia de 98%, enquanto os outros modelos melhoraram para 85%, 87% e 82%. No conjunto de 100KB, todos os classificadores tiveram suas acurácias reduzidas para aproximadamente 68%.

No gráfico apresentado na Figura 82, são exibidas as acurácias dos mesmos classificadores utilizando a estratégia *cross-validation*. No conjunto de 20KB, o KNN alcançou uma taxa de acerto de 100%, enquanto NB, SVM e RF obtiveram 88%, 96% e 87%. No

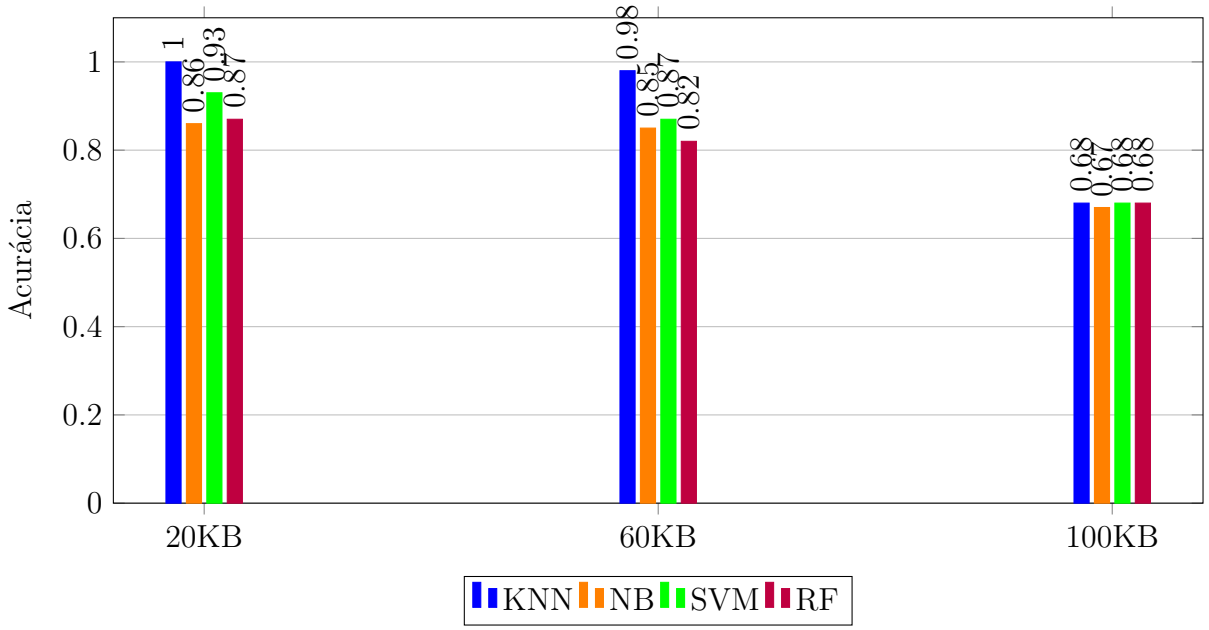


Figura 81 – Acurácia dos modelos por Dataset CBC - Estratégia *train-test split*

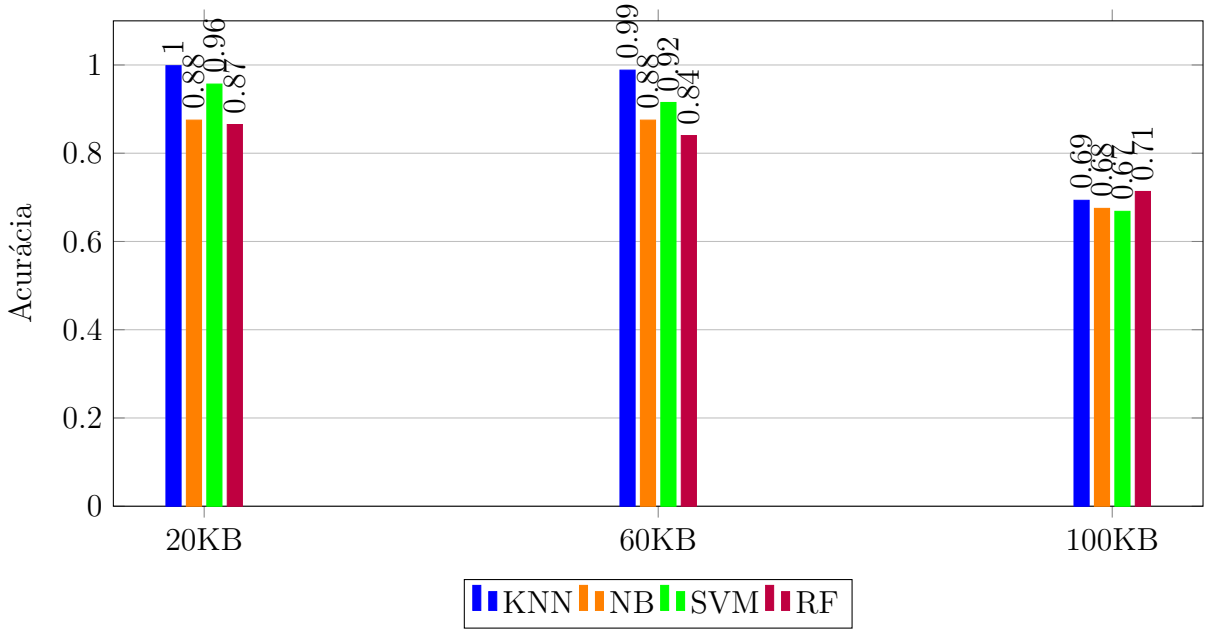


Figura 82 – Acurácia dos modelos por Dataset CBC - Estratégia *cross-validation*

conjunto de 60KB, a precisão do KNN diminuiu para 99%, enquanto os outros modelos melhoraram para aproximadamente 86%, 92% e 84%. No conjunto de 100KB, novamente, as acurácias de todos os classificadores reduziram para aproximadamente 69%. Estes resultados destacam o KNN como o classificador mais eficaz em ambas as estratégias.

Similar ao experimento anterior, ao considerar os tamanhos das bases de dados e os resultados da *cross-validation* que validam os da estratégia *train-test split*, confirma-se a ausência de viés na divisão dos conjuntos de treinamento e teste, ratificando a estabilidade dos modelos preditivos utilizados no experimento.

Nesta dissertação, foram analisados algoritmos diferentes daqueles examinados por Tan, Deng e Zhang(104), o único trabalho relacionado que analisou exclusivamente os primeiros blocos de bits gerados por criptogramas. Essa diferença impossibilita uma comparação direta entre os dois estudos. No entanto, ao comparar a eficácia geral obtida por ambas as pesquisas na única condição em que coincidem (amostras de 100 KB, com chaves e IVs aleatórios), observa-se que a acurácia média do método utilizado nesta pesquisa (67,75%) superou a de Tan, Deng e Zhang(104) (38,78%).

De modo mais amplo, nos três cenários analisados, a metodologia proposta obteve taxas de identificação variando de 100% a 68%, e todos os classificadores superaram consideravelmente o índice aleatório de 16.67%. O método proposto também superou as acurácias dos métodos de Fan e Zhao(108) (13.5% em amostras de 512 KB, com probabilidade de escolha aleatória de 12.5%), Hu e Zhao(109) (12.64% em amostras de 512 KB, com probabilidade de escolha aleatória de 12.5%), Yu e Shi(114) (29.8% em amostras de 256 KB, com probabilidade de escolha aleatória de 12.5%) e Mello e Xexeo(103) (50% em amostras variando entre 2 e 34 bits, com probabilidade de escolha aleatória de 14.28%).

A tendência novamente observada de que o aumento no tamanho dos dados contribui para uma maior aleatoriedade nos dados, reforçando a concepção de que, para fins de análise estatística, o aumento do tamanho no conjunto de dados nem sempre melhora a identificação de criptossistemas, descoberta inicialmente por Yuan et al.(11) e Yuan et al.(12), pode ser confirmada pelo cálculo da variância dos conjuntos de dados usando a função `VarianceThreshold` da biblioteca *Scikit-Learn*. Nos *datasets* de 20KB, 60KB e 100KB, a variância média foi de 5002,70, 14000,93 e 16813,27, respectivamente. Esses resultados, similares aos obtidos no experimento anterior, onde foram encontrados respectivamente 6654,23, 13704,71 e 16456,01, confirmam a tendência observada. Indicam que, neste experimento, quanto maior o tamanho do criptograma, maior é a amplitude dos valores p, evidenciando maior probabilidade de aleatoriedade nos dados. Esses resultados eram esperados, uma vez que a metodologia empregada neste experimento assemelha a análise no modo CBC e uma análise no modo ECB.

4.2.8.2 Considerações sobre os arquivos de primeiros blocos concatenados

De acordo com os resultados obtidos neste experimento, verifica-se que os classificadores foram capazes de identificar a presença de padrões binários únicos ou assinaturas nos primeiros blocos dos arquivos analisados. Com exceção das amostras Kyber no *dataset* de 100KB, os criptossistemas de chave pública pós-quânticos se mostraram completamente distinguíveis entre si, confirmando as descobertas do primeiro experimento e indicando novamente a vulnerabilidade dos criptossistemas de chave pública pós-quânticos, que possuem nível de distinguibilidade IND-CPA, a ataques de distinção.

Semelhante ao experimento anterior, a análise das matrizes de confusão no *dataset*

de 100KB revela que os classificadores novamente cometeram erros ao classificar amostras do Kyber, apresentando novamente tendência de confusão entre amostras do Kyber e amostras do AES e Blowfish. Existe a probabilidade de que esses erros decorram dos componentes aleatórios que compõem a saída c do `Kyber.CPA.PKE` ($c = (u, v)$ - onde $u := \text{Compress}_q(A^T r + e_1, d_u)$ e $v := \text{Compress}_q(t^T r + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v)$ - derivados da matriz aleatória A, do vetor r, e dos vetores de erro e_1 e e_2, todos gerados aleatoriamente), dificultando a identificação dessas amostras em comparação com as demais pós-quânticas.

A presença de padrões binários, provenientes das características únicas dos dados de entrada e derivadas dos problemas matemáticos LWE, RLWE e MLWE utilizados nas suas formulações, tornou os arquivos de primeiros blocos concatenados destes criptossistemas distinguíveis. Tal qual o experimento anterior, é relevante observar que esses algoritmos apresentaram um grau de indistinção inferior ao das cifras clássicas examinadas, as quais mostraram taxas de identificação inferiores.

Há que se observar também que a metodologia adotada neste experimento, que se concentra nos primeiros blocos concatenados, elimina o efeito do encadeamento de blocos do modo CBC e representa um avanço na identificação de cifras operando nesse modo, de modo que, em um cenário em que um grande conjunto de arquivos cifrados esteja disponível, torna-se possível identificar padrões nos textos cifrados e determinar o algoritmo gerador do criptograma.

4.3 Experimento - KEM

4.3.1 Organização do experimento

Os esquemas de criptografia de chave pública discutidos nas seções anteriores fazem parte de mecanismos projetados para efetuar o compartilhamento seguro de chaves de sessão simétricas em canais inseguros. Esses esquemas estão incluídos na categoria *Public-key Encryption and Key-establishment Algorithms* do concurso de padronização *Post-Quantum Cryptography* atualmente em andamento pelo NIST.

Cramer e Shoup(35) apresentaram os fundamentos do Mecanismo de Encapsulamento de Chave (KEM) e do Mecanismo de Encapsulamento de Dados (DEM), além de apresentar um KEM denominado CS-KEM e demonstrar sua segurança contra ataques do tipo *Chosen-Ciphertext* (CCA). Entretanto, na pesquisa posterior conduzida por Herranz, Hofheinz e Kiltz(127), observou-se que o KEM derivado do esquema híbrido de Kurosawa e Desmedt não atendia completamente aos requisitos de segurança contra ataques CCA, embora o próprio esquema PKE híbrido permaneça resiliente a esses tipos de ataques.

Tendo em vista a categoria *Public-key Encryption and Key-establishment Algorithms* do processo seletivo do NIST, este experimento teve como objetivo identificar padrões

nas chaves de sessão criptografadas e compartilhadas por esses mecanismos, visando determinar qual KEM as transmitiu. Para uma comparação justa, todas as implementações analisadas neste experimento - `FrodoKEM-640-AES`, `CRYSTALS-KYBER512`, `LightSaber` e `NTRUhps2048509` - atendem ao mesmo nível de segurança definido pelo NIST na FIPS203 (43), a saber, nível I. Assim como nos dois experimentos anteriores, também foram analisados conjuntos de dados de 20KB, 60KB e 100KB.

Seguindo a metodologia ilustrada na Figura 83, 100 arquivos de 20KB, 60KB e 100KB foram gerados, cada um contendo diversas chaves de sessão simétricas. Considerando que cada KEM tem como objetivo transmitir chaves de 256 bits, cada arquivo de 20KB, 60KB e 100KB continha, respectivamente, 78.125, 234.375 e 390.625 chaves de sessão concatenadas.

Em seguida, em cada um desses três cenários, os 100 arquivos contendo chaves de sessão concatenadas foram submetidos ao NIST STS. Semelhante aos experimentos anteriores, cada arquivo de chaves concatenadas foi representado por um vetor de 43 campos. Destes, 41 correspondiam aos valores p retornados pelos 15 testes estatísticos da bateria, enquanto os outros dois indicavam as quantidades de bits 0s e 1s.

A coleção dos 400 vetores representativos (100 vetores para cada criptossistema pós-quântico) foi reunida para formar o conjunto de dados do experimento. Semelhante aos dois primeiros experimentos, foram testadas as estratégias *train-test split* e *cross-validation*. Primeiramente, avaliou-se a estratégias *train-test split* e durante a fase de pré-processamento dos dados, cada conjunto de dados foi dividido em duas partes: a primeira com 280 amostras (conjunto de treinamento, correspondendo a 70% dos dados) e a segunda com 120 amostras (conjunto de teste, representando os 30% restantes). Os modelos treinados foram, então, avaliados no conjunto de teste, e seus resultados foram analisados utilizando as métricas de avaliação de desempenho. Posteriormente, para cada um dos três *datasets*, avaliou-se a estratégia *cross-validation* com o valor de K igual a 10.

Considerando os resultados do método proposto por Souza, Xexéo e Oliveira(128), que foi capaz de agrupar criptogramas gerados pelos algoritmos AES (com chaves de 128, 192 e 256 bits), DES (com chaves de 64 bits) e RSA (com chaves de 512 e 1024 bits), utilizando chaves aleatórias e distintas para cada um, bem como o fato de que as chaves de sessão criptografadas e compartilhadas nas implementações `FrodoKEM-640-AES` (129), `CRYSTALS-KYBER512` (130), `LightSaber` (131) e `NTRUhps2048509` (132), possuem 16, 32, 32 e 32 bytes, respectivamente, existe a probabilidade de os classificadores serem capazes de identificar as chaves de sessão do `FrodoKEM-640-AES` e não conseguirem distinguir as geradas pelos demais KEM.

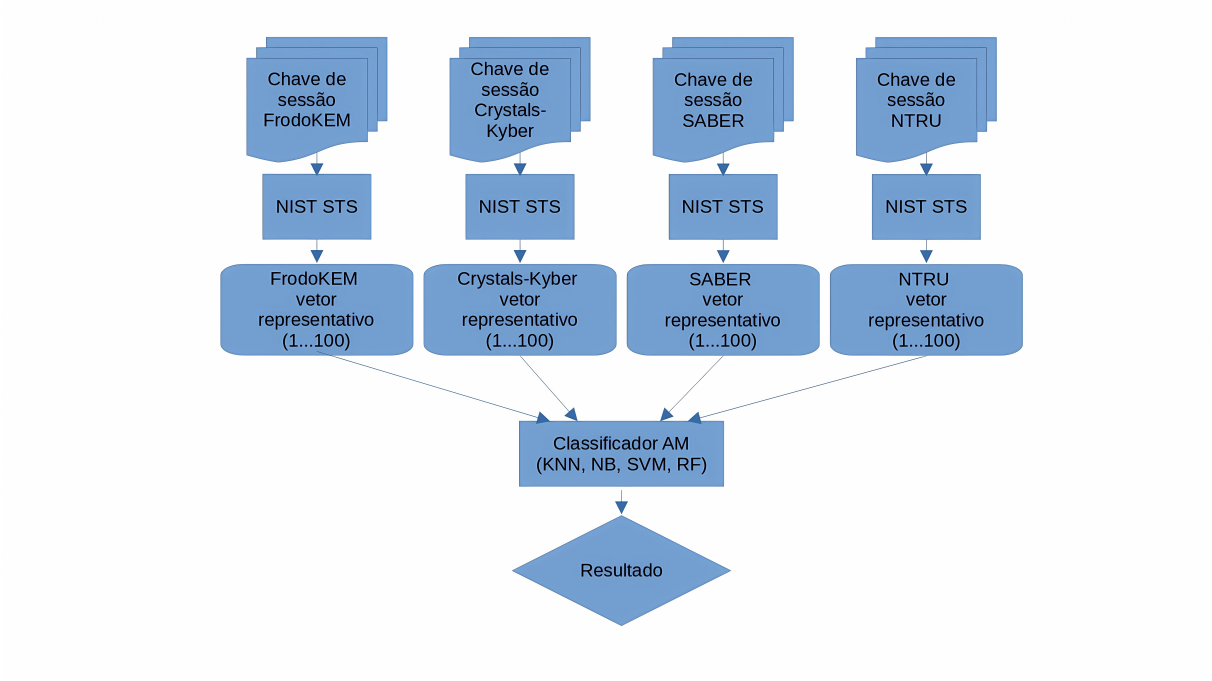


Figura 83 – Metodologia experimento - KEM

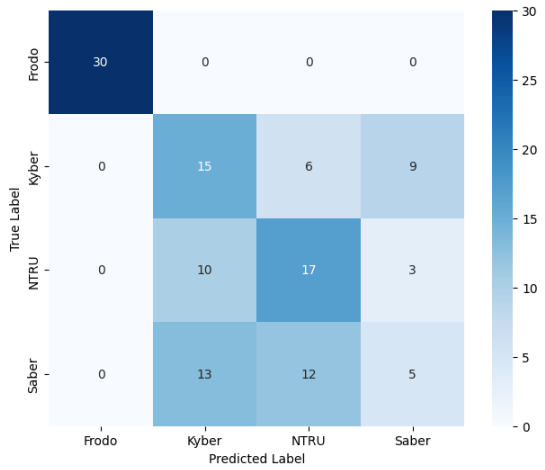


Figura 84 – Matriz de confusão - KNN - 20KB

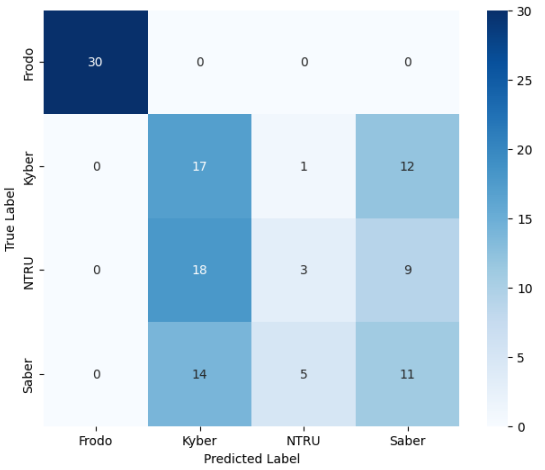


Figura 85 – Matriz de confusão - NB - 20KB

4.3.2 Resultado Experimento KEM *Dataset* 20KB Estratégia *train-test split*

A análise das matrizes de confusão ilustradas nas Figuras 84, 85, 86 e 87 revela que:

- **Para a classe Frodo:**
  - Todos os classificadores - KNN, NB, SVM e RF - classificaram corretamente todas as 30 amostras.
- **Para a classe Kyber:**

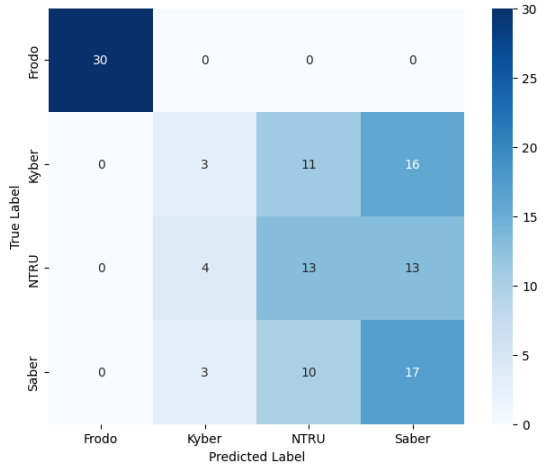


Figura 86 – Matriz de confusão - SVM - 20KB

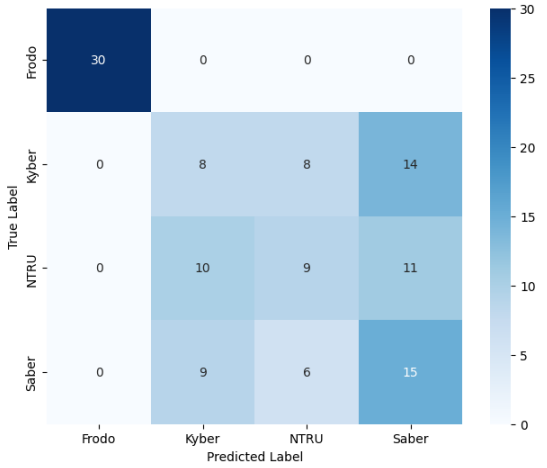


Figura 87 – Matriz de confusão - RF - 20KB

  - O KNN identificou 15 amostras e obteve precisão, *recall* e *F1-score* de 39%, 50% e 44%, respectivamente.
  - O NB identificou 17 amostras, alcançando uma precisão, *recall* e *F1-score* de 35%, 57% e 43%, respectivamente.
  - O SVM identificou 3 amostras corretamente, enquanto o RF classificou corretamente 8 amostras, registrando precisão, *recall* e *F1-score* de 30%, 27% e 28%, respectivamente.

- **Para a classe NTRU:**
  - O KNN classificou corretamente 17 amostras, alcançando uma precisão, *recall* e *F1-score* de 49%, 57% e 52%, respectivamente.
  - O NB, embora tenha obtido uma acurácia de 51%, classificou 3 amostras corretamente, resultando em uma precisão, *recall* e *F1-score* de 33%, 10% e 15%, respectivamente.
  - O SVM identificou 13 amostras, resultando em uma precisão, *recall* e *F1-score* de 38%, 43% e 41%, respectivamente.
  - O RF identificou 9 amostras e alcançou precisão, *recall* e *F1-score* de 38%, 50% e 43%, respectivamente.

- **Para a classe Saber:**
  - O KNN acertou 5 amostras, alcançando uma precisão, *recall* e *F1-score* de 29%, 17% e 21%, respectivamente.
  - O NB classificou corretamente 11 amostras, resultando em uma precisão, *recall* e *F1-score* de 34%, 37% e 35%, respectivamente.

– O SVM identificou 17 amostras e obteve precisão, *recall* e *F1-score* de 37%, 57% e 45%, respectivamente.

– O RF acertou 15 amostras e registrou precisão, *recall* e *F1-score* de 38%, 50% e 43%, respectivamente.

4.3.3 Resultado Experimento KEM *Dataset* 20KB Estratégia *cross-validation*

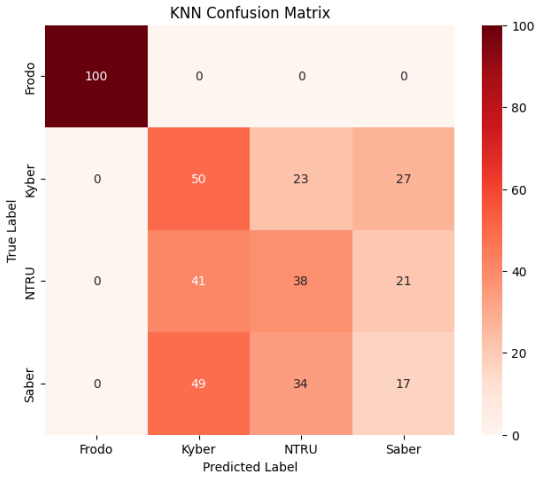


Figura 88 – Matriz de confusão - KNN - 20KB

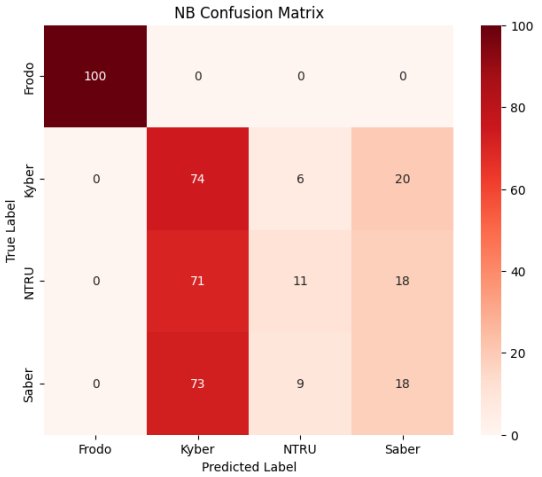


Figura 89 – Matriz de confusão - NB - 20KB

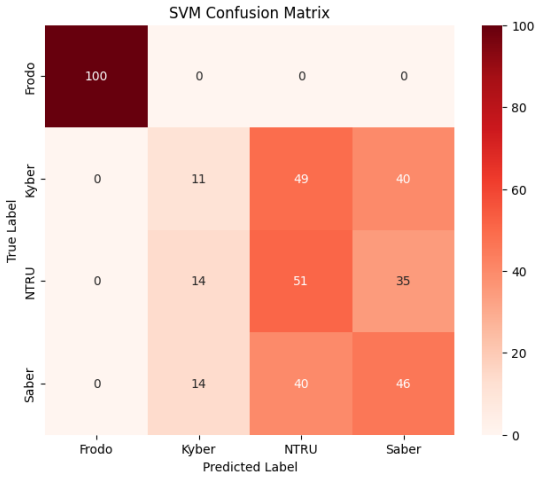


Figura 90 – Matriz de confusão - SVM - 20KB

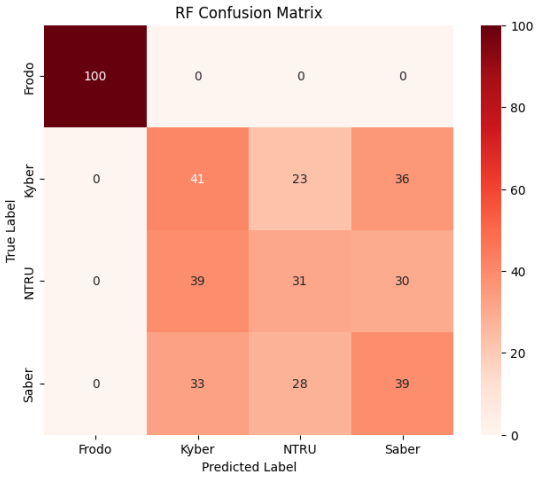


Figura 91 – Matriz de confusão - RF - 20KB

A análise das matrizes de confusão ilustradas nas Figuras 88, 89, 90 e 91 revela que:

- **Para a classe Frodo:**

  – Todos os algoritmos - KNN, NB, SVM e RF - classificaram corretamente as 100 amostras, resultando precisão, *recall* e *F1-score* de 100%.

- **Para a classe Kyber:**
  - KNN classificou corretamente 50 amostras, obtendo precisão de 36%, *recall* de 50% e *F1-score* de 42%.
  - NB acertou 74 amostras e registrou precisão de 34%, *recall* de 74% e *F1-score* de 47%.
  - SVM identificou corretamente 11 amostras, resultando em precisão de 28%, *recall* de 11% e *F1-score* de 16%.
  - RF acertou 41 amostras e obteve precisão de 36%, *recall* de 41% e *F1-score* de 38%.
- **Para a classe NTRU:**
  - KNN classificou corretamente 38 amostras, alcançando uma precisão de 40%, *recall* de 38% e *F1-score* de 39%.
  - NB acertou 11 amostras, resultando em uma precisão de 42%, *recall* de 11% e *F1-score* de 17%.
  - SVM identificou corretamente 51 amostras, obtendo uma precisão de 36%, *recall* de 51% e *F1-score* de 42%.
  - RF acertou 31 amostras, apresentando uma precisão de 38%, *recall* de 31% e *F1-score* de 34%.
- **Para a classe Saber:**
  - KNN classificou corretamente 17 amostras, obtendo precisão de 26%, *recall* de 17% e *F1-score* de 21%.
  - NB acertou 18 amostras, registrando uma precisão de 32%, *recall* de 18% e *F1-score* de 23%.
  - SVM identificou corretamente 46 amostras, resultando numa precisão de 38%, *recall* de 46% e *F1-score* de 42%.
  - RF acertou 39 amostras, alcançando precisão de 37%, *recall* de 39% e *F1-score* de 38%.

4.3.4 Resultado Experimento KEM *Dataset* 60KB Estratégia *train-test split*

A análise das matrizes de confusão ilustradas nas Figuras 92, 93, 94 e 95 revela que:

- **Para a classe Frodo:**

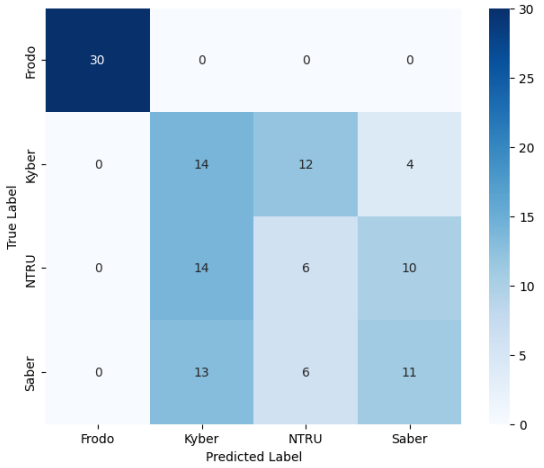


Figura 92 – Matriz de confusão - KNN - 60KB

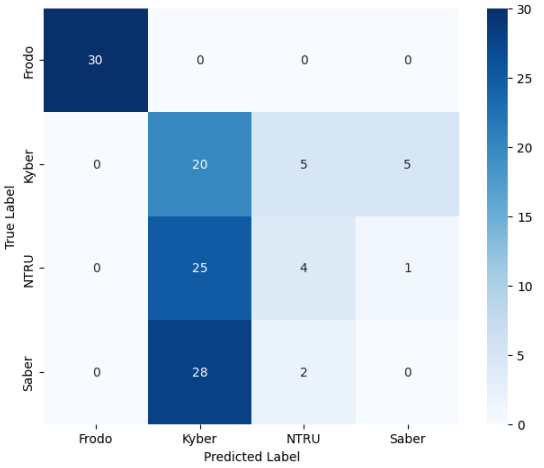


Figura 93 – Matriz de confusão - NB - 60KB

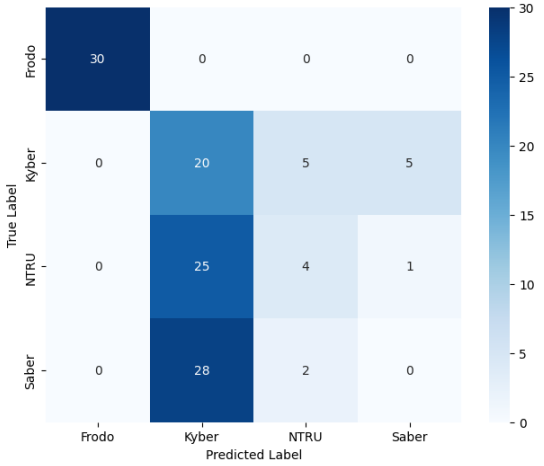


Figura 94 – Matriz de confusão - SVM - 60KB

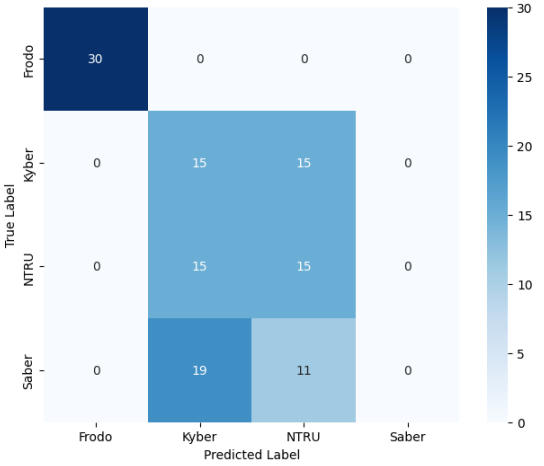


Figura 95 – Matriz de confusão - RF - 60KB

- Todos os classificadores - KNN, NB, SVM e RF - classificaram todas as 30 amostras corretamente.
- **Para a classe Kyber:**
  - KNN classificou corretamente 14 amostras, registrando precisão, *recall* e *F1-score* de 34%, 47% e 39%, respectivamente.
  - NB e SVM identificaram 20 amostras, alcançando em uma precisão, *recall* e *F1-score* de 27%, 67% e 39%, respectivamente.
  - RF acertou 8 amostras.
- **Para a classe NTRU:**
  - KNN identificaram 6 amostras, resultando em precisão, *recall* e *F1-score* de 25%, 20% e 22%, respectivamente.

  - NB e SVM acertaram 4 amostras.
  - RF classificaram corretamente 15 amostras e obteve precisão, *recall* e *F1-score* de 37%, 50% e 42%, respectivamente.

- **Para a classe Saber:**
  - KNN identificaram 11 amostras de Saber, alcançou *recall* de 37% e *F1-score* de 40%.
  - NB, SVM e RF não classificaram corretamente nenhuma amostra.

4.3.5 Resultado Experimento KEM *Dataset* 60KB Estratégia *cross-validation*

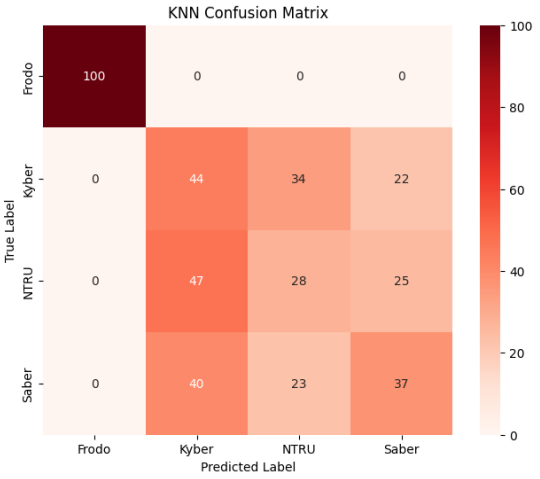


Figura 96 – Matriz de confusão - KNN - 60KB

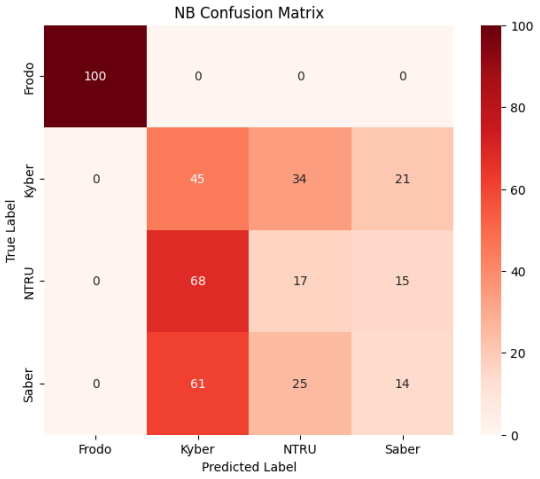


Figura 97 – Matriz de confusão - NB - 60KB

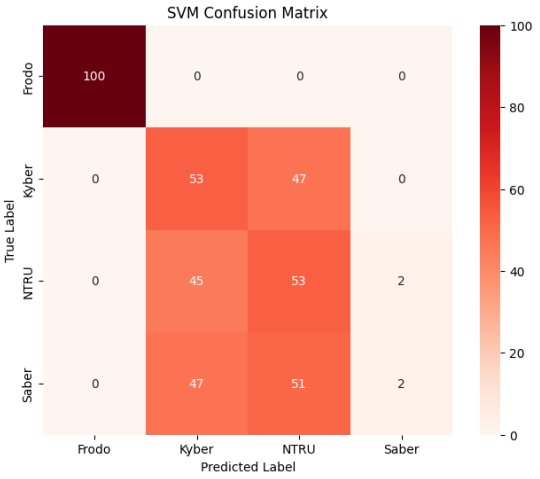


Figura 98 – Matriz de confusão - SVM - 60KB

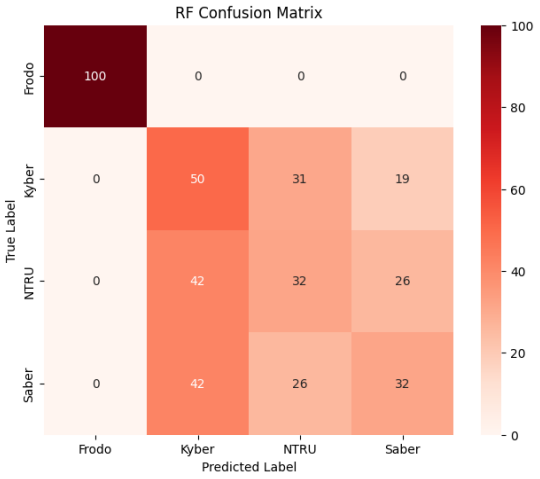


Figura 99 – Matriz de confusão - RF - 60KB

A análise das matrizes de confusão ilustradas nas Figuras 96, 97, 98 e 99 revela que:

- **Para a classe Frodo:**
  - Todos os algoritmos - KNN, NB, SVM e RF - classificaram corretamente as 100 amostras, resultando em precisão, *recall* e *F1-score* de 100%.
- **Para a classe Kyber:**
  - KNN classificou corretamente 44 amostras, obtendo precisão de 34%, *recall* de 44% e *F1-score* de 38%.
  - NB acertou 45 amostras, com uma precisão de 26%, *recall* de 45% e *F1-score* de 33%.
  - SVM identificou corretamente 53 amostras, resultando em precisão de 37%, *recall* de 53% e *F1-score* de 43%.
  - RF acertou 50 amostras, com uma precisão de 37%, *recall* de 50% e *F1-score* de 43%.
- **Para a classe NTRU:**
  - KNN classificou corretamente 28 amostras, alcançando uma precisão de 33%, *recall* de 28% e *F1-score* de 30%.
  - NB acertou 17 amostras, com uma precisão de 22%, *recall* de 17% e *F1-score* de 19%.
  - SVM identificou corretamente 53 amostras, obtendo uma precisão de 35%, *recall* de 53% e *F1-score* de 42%.
  - RF acertou 32 amostras, com uma precisão de 36%, *recall* de 32% e *F1-score* de 34%.
- **Para a classe Saber:**
  - KNN classificou corretamente 37 amostras, obtendo precisão de 44%, *recall* de 37% e *F1-score* de 40%.
  - NB acertou 14 amostras, com uma precisão de 28%, *recall* de 14% e *F1-score* de 19%.
  - SVM identificou corretamente 2 amostras, resultando em precisão de 50%, *recall* de 2% e *F1-score* de 4%.
  - RF acertou 26 amostras, alcançando precisão de 42%, *recall* de 32% e *F1-score* de 36%.

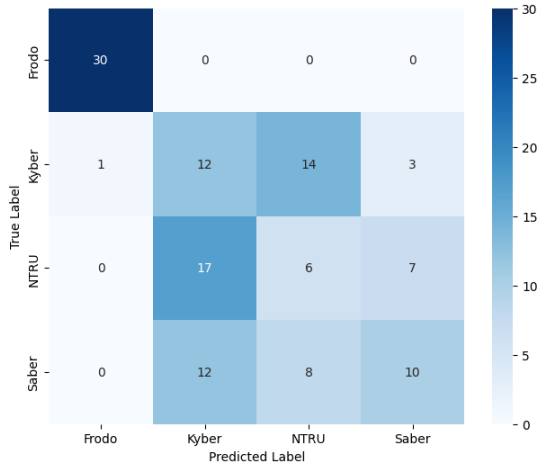


Figura 100 – Matriz de confusão - KNN - 100KB

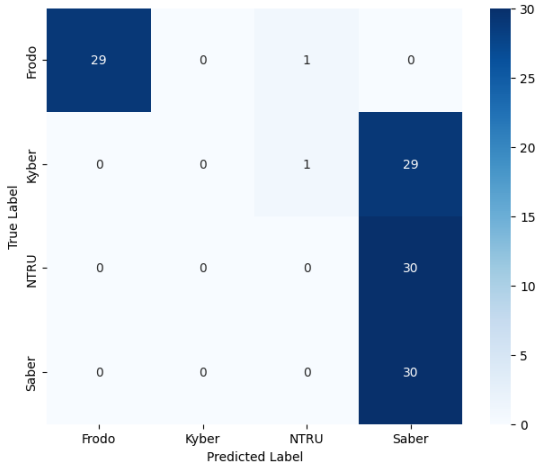


Figura 101 – Matriz de confusão - NB - 100KB

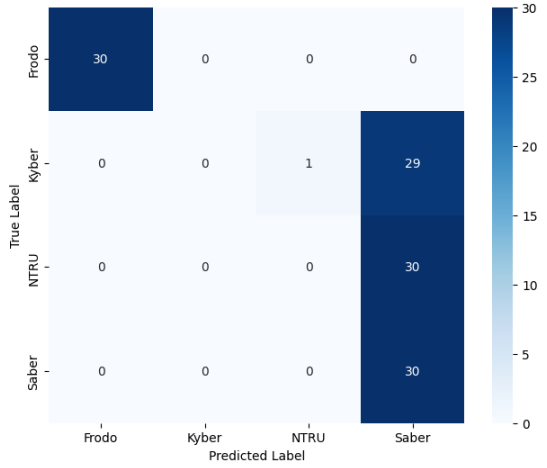


Figura 102 – Matriz de confusão - SVM - 100KB

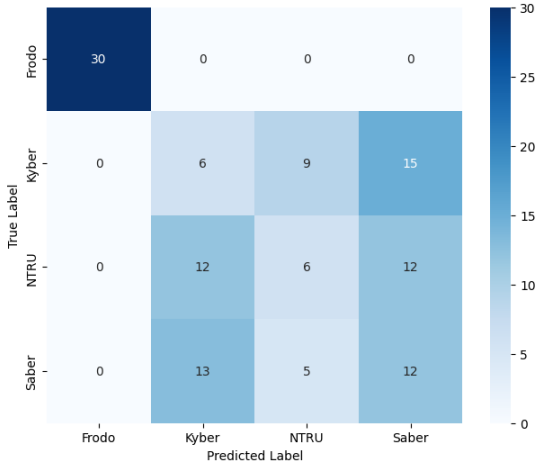


Figura 103 – Matriz de confusão - RF - 100KB

4.3.6 Resultado Experimento KEM *Dataset* 100KB Estratégia *train-test split*

A análise das matrizes de confusão apresentadas nas Figuras 100, 101, 102 e 103 revela que:

- **Para a classe Frodo:**
  - KNN, NB, SVM e RF classificaram corretamente todas as 30 amostras, obtendo precisão, *recall* e *F1-score* de 100%.
- **Para a classe Kyber:**
  - KNN acertou 12 amostras com precisão de 29%, resultando em um *recall* de 40% e *F1-score* de 34%.
  - RF classificou corretamente 6 amostras e obteve precisão de 19%, *recall* de 20% e *F1-score* de 20%.

  - NB e SVM não identificaram nenhuma amostra.

- **Para a classe NTRU:**
  - KNN acertou 6 amostras com precisão de 21%, resultando em *recall* de 20% e *F1-score* de 21%.
  - RF classificou corretamente 6 amostras, alcançando precisão de 30%, *recall* de 20% e *F1-score* de 24%.
  - NB e SVM não conseguiram identificar nenhuma amostra.

- **Para a classe Saber:**
  - KNN acertou 10 amostras com precisão de 50%, obtendo *recall* de 33% e *F1-score* de 40%.
  - RF identificaram 12 amostras com precisão de 31%, registrando *recall* de 40% e *F1-score* de 35%.
  - NB e SVM classificaram corretamente as 30 amostras e alcançaram *recall* de 100% e *F1-score* de 100%.

4.3.7 Resultado Experimento KEM *Dataset* 100KB Estratégia *cross-validation*

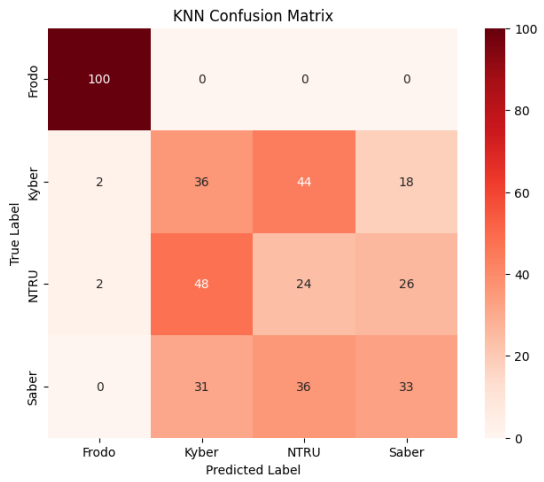


Figura 104 – Matriz de confusão - KNN - 100KB

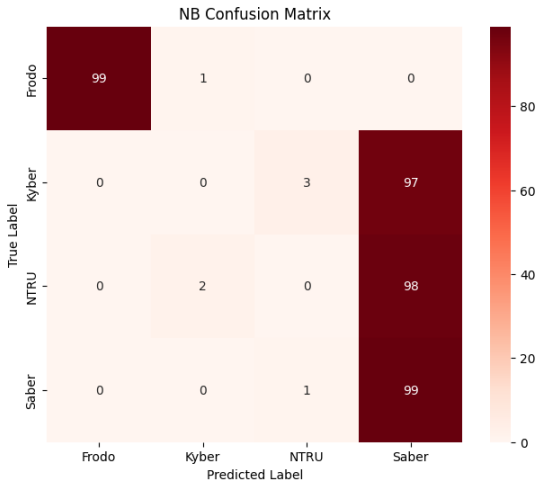


Figura 105 – Matriz de confusão - NB - 100KB

A análise das matrizes de confusão apresentadas nas Figuras 104, 105, 106 e 107 revela que:

- **Para a classe Frodo:**
  - Os classificadores KNN, SVM e RF acertaram as 100 amostras e obtiveram precisão de 100%, *recall* de 100% e *F1-score* de 100%.

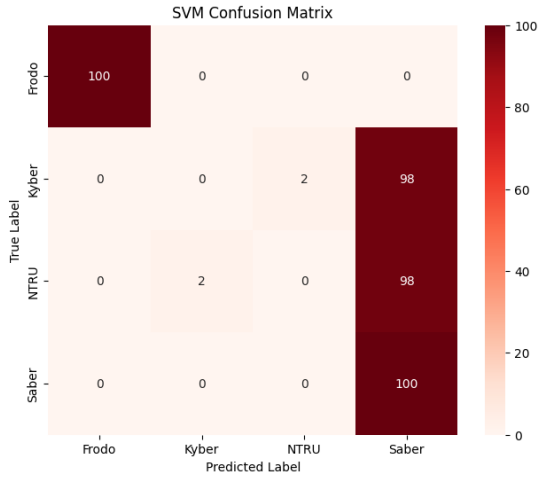


Figura 106 – Matriz de confusão - SVM - 100KB

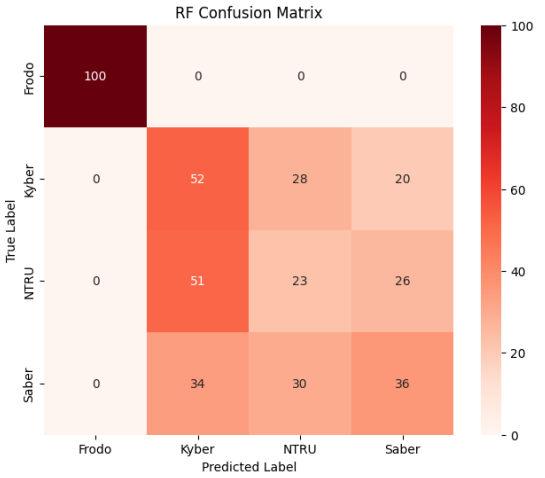


Figura 107 – Matriz de confusão - RF - 100KB

  - NB classificou 99 amostras corretamente, registrando precisão de 99%, *recall* de 99% e *F1-score* de 99%.

- **Para a classe Kyber:**
  - KNN identificou 36 amostras, obtendo precisão de 31%, *recall* de 36% e *F1-score* de 33%.
  - NB e SVM não acertaram nenhuma amostra, resultando em precisão, *recall* e *F1-score* de 0%.
  - RF classificou 52 amostras corretamente, alcançando precisão de 38%, *recall* de 52% e *F1-score* de 44%.

- **Para a classe NTRU:**
  - KNN identificou 24 amostras corretamente, obtendo precisão de 23%, *recall* de 24% e *F1-score* de 24%.
  - NB e SVM não classificou corretamente nenhuma amostra, resultando em precisão, *recall* e *F1-score* de 0%.
  - Modelo RF acertou 23 amostras, obtendo precisão de 28%, *recall* de 23% e *F1-score* de 25%.

- **Para a classe Saber:**
  - KNN acertou 33 amostras, registrando precisão de 43%, *recall* de 33% e *F1-score* de 37%.
  - NB identificou 99 amostras, obtendo precisão de 34%, *recall* de 99% e *F1-score* de 51%.
  - SVM classificou 100 amostras corretamente, obtendo *recall* de 100%.

– RF identificou 36 amostras corretamente, alcançando precisão de 44%, *recall* de 36% e *F1-score* de 40%.

4.3.8 Considerações Experimento KEM

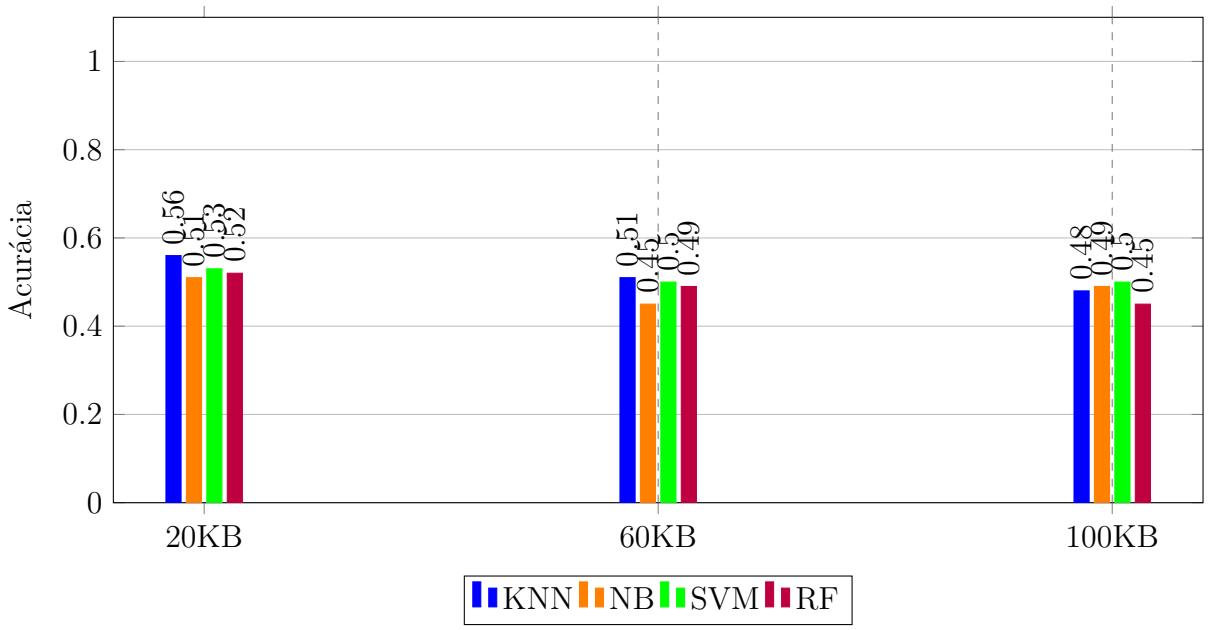


Figura 108 – Acurácia por Tamanho de Dataset - Estratégia *train-test split*

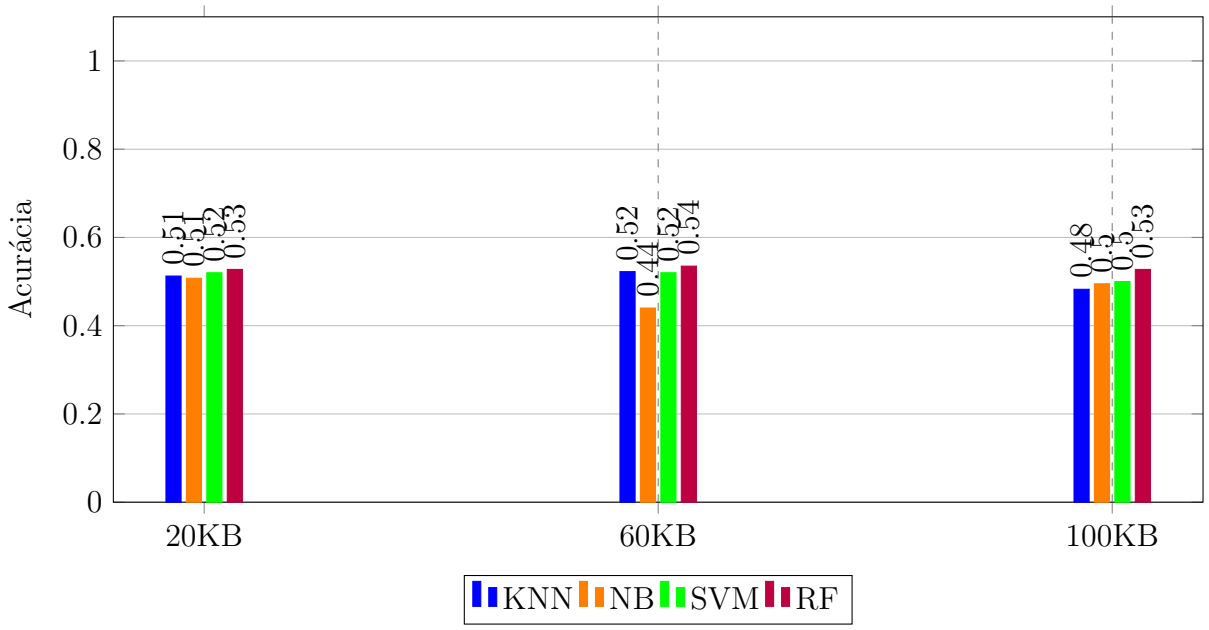


Figura 109 – Acurácia por Tamanho de Dataset - Estratégia *cross-validation*

4.3.8.1 Considerações sobre os classificadores

O gráfico na Figura 108 ilustra a acurácia dos classificadores KNN, NB, SVM e RF nos *datasets* de 20KB, 60KB e 100KB na estratégia *train-test split*. No conjunto de 20KB,

o KNN obteve 56% de acurácia, enquanto NB, RF e SVM atingiram 51%, 53% e 52%. No conjunto de 60KB, o KNN alcançou 51%, enquanto os demais registraram 45%, 50% e 49%. No conjunto de 100KB, apesar de ligeira queda, o desempenho dos classificadores se mostrou estável, onde o KNN caiu para 48% e os outros ficaram entre 49% e 45%.

O gráfico na Figura 109 representa a acurácia dos classificadores KNN, NB, SVM e RF nos conjuntos de dados de 20KB, 60KB e 100KB utilizando a estratégia *cross-validation*. No conjunto de 20KB, o KNN apresentou uma acurácia média de 51.25%, enquanto NB, RF e SVM alcançaram 50.75%, 52% e 52.75%, respectivamente. Para o dataset de 60KB, o KNN obteve uma acurácia média de 52.25%, enquanto os demais modelos registraram 44%, 52% e 53.5%. Já no dataset de 100KB, o KNN apresentou uma acurácia média de 48.25%, enquanto os outros modelos ficaram entre 49.5% e 52.75%.

Considerando que foram analisados quatro KEMs, observa-se que os resultados obtidos são aproximadamente o dobro do índice probabilístico aleatório (25%). A consistência desses resultados é verificada tanto na estratégia *train-test split* quanto na *cross-validation*, reafirmando a imparcialidade na divisão dos conjuntos de treinamento e teste. Essa concordância ratifica, assim, a estabilidade dos modelos preditivos empregados no experimento.

Na estratégia de *train-test split*, o KNN demonstrou ser o classificador mais eficaz, enquanto na estratégia de *cross-validation*, o RF mostrou-se ligeiramente superior, embora ambos tenham alcançado acurácias muito próximas. É importante salientar que os resultados satisfatórios se devem à distinção das amostras do Frodo em relação às dos demais KEM. Se a análise fosse restrita aos KEM participantes da terceira rodada do concurso do NIST (Crystals-Kyber, NTRU e Saber), as métricas de avaliação teriam se aproximado do índice aleatório (33,34%), conforme pode ser observado nos gráficos apresentados nas Figuras 110 e 111 e o ataque de distinção teria sido malsucedido.

A análise da variância nos conjuntos de dados, utilizando a função `VarianceThreshold` da biblioteca *Scikit-Learn*, revelou que para o *dataset* de 20KB, a variância foi 5650,30. No *dataset* de 60KB, a variância aumentou para 16949,61, e para o *dataset* de 100KB, a variância atingiu 27205,01. Esses resultados corroboram a tendência observada nos experimentos anteriores, indicando que, à medida que o tamanho do criptograma aumenta, a amplitude dos valores p também aumenta, sinalizando uma maior probabilidade de aleatoriedade nos dados.

Considerando que não foram identificadas pesquisas relacionadas envolvendo ataques de distinção em KEM clássicos ou pós-quânticos, não é possível estabelecer uma comparação entre os resultados apresentados nesta Subseção e os de outros trabalhos.

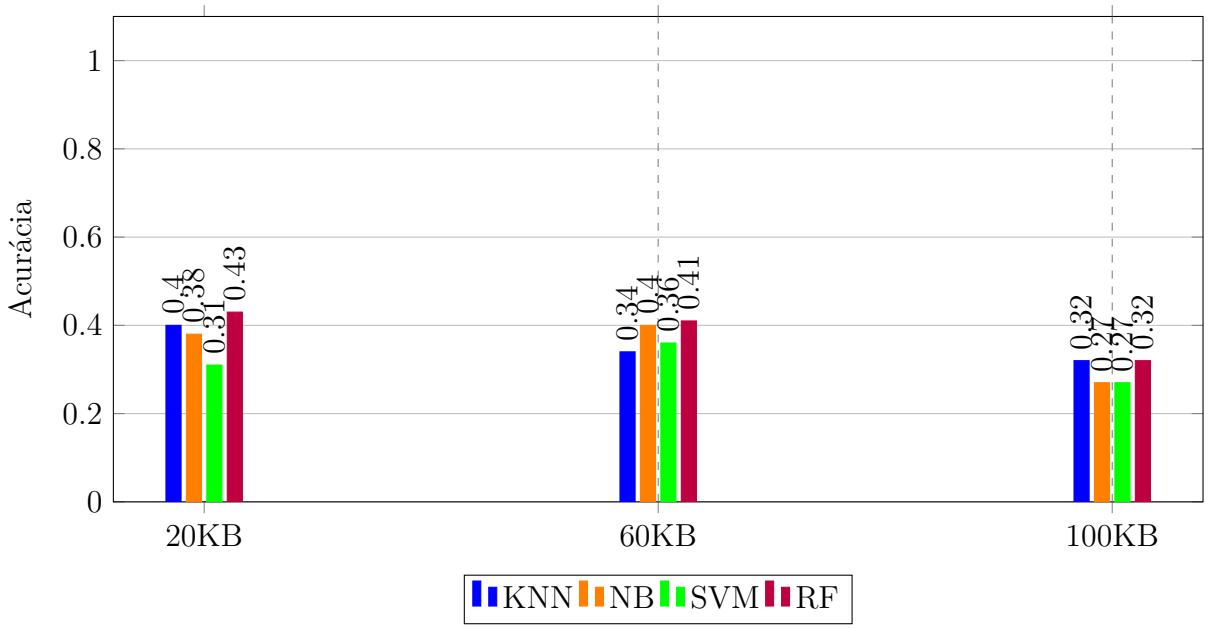


Figura 110 – Acurácia por Tamanho de Dataset (sem Frodo) - Estratégia *train-test split*

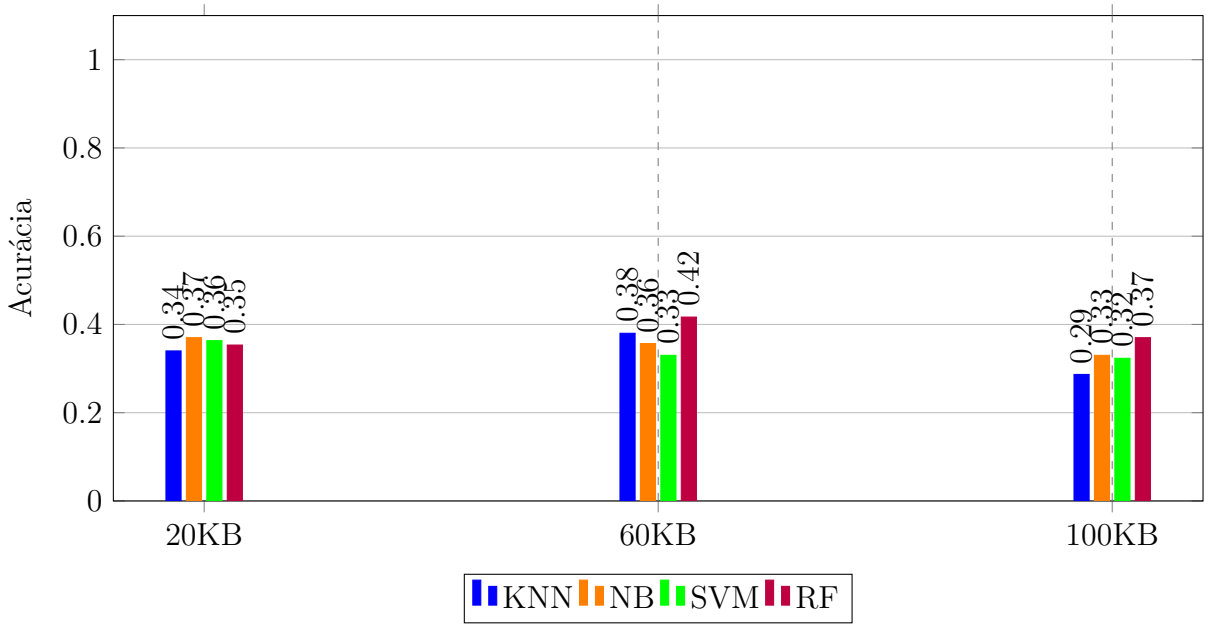


Figura 111 – Acurácia por Tamanho de Dataset (sem Frodo) - Estratégia *cross-validation*

4.3.8.2 Considerações sobre as chaves de sessão

Os resultados deste experimento indicam que as chaves de sessão criptografadas e compartilhadas pelo `FrodoKEM-640-AES` possuem padrões binários distintos que as tornam distinguíveis das chaves dos demais KEM. Por outro lado, as amostras dos outros mecanismos não puderam ser distinguidas entre si, tendo em vista que as acurácias obtidas pelos classificadores não foram satisfatoriamente superior ao índice de escolha aleatória. Esse resultado era esperado devido ao tamanho reduzido das chaves provenientes do `FrodoKEM-640-AES` (16 bytes), contrastando com o tamanho das demais chaves (32 bytes).

Esses resultados confirmam que o `FrodoKEM-640-AES` possui um nível de indistinguibilidade IND-CCA, enquanto os KEM `CRYSTALS-KYBER512`, `LightSaber` e `NTRUhps2048509`, possuem o nível mais rigoroso de indistinguibilidade (IND-CCA2).

Conclui-se que, nas condições deste experimento, o `FrodoKEM-640-AES` apresenta vulnerabilidades passíveis de serem exploradas em um ataque de distinção, uma vez que suas chaves de sessão compartilhadas podem ser identificadas ao trafegarem em um canal de comunicação inseguro. É igualmente relevante observar a resistência dos KEM `CRYSTALS-KYBER512`, `LightSaber` e `NTRUhps2048509` a ataques de distinção, tendo em vista que as características distintivas dos problemas matemáticos empregados na construção de seus esquemas de encriptação de chave pública, que possuem nível de indistinguibilidade IND-CPA, não se manifestam posteriormente nos KEM (IND-CCA2), de modo que, usando esta metodologia baseada em testes estatísticos, não foram identificados padrões que os tornem identificáveis.

5 CONSIDERAÇÕES FINAIS

Com base nos resultados apresentados no Capítulo 4, nas hipóteses delineadas no Capítulo 1, e considerando que durante a revisão bibliográfica não foram encontradas pesquisas relacionadas aos criptossistemas pós-quânticos, o que reforça a singularidade e inovação deste trabalho, a principal contribuição desta pesquisa consiste na identificação de padrões nos textos cifrados gerados pelos esquemas de encriptação de chave pública dos criptossistemas pós-quânticos Frodo, Crystals Kyber, NTRU e Saber. Conclui-se que a abordagem proposta permitiu a realização bem-sucedida de ataques de identificação ou distinção em textos cifrados produzidos por esses esquemas.

Para atender aos objetivos específicos estabelecidos no Capítulo 1, desenvolveu-se um método supervisionado que se baseia nos quinze testes estatísticos da suíte NIST STS SP 800-22, além de empregar classificadores de aprendizado de máquina. Esse método é aplicável tanto no modo ECB quanto no modo CBC.

A eficácia do método proposto se tornou evidente por meio da comparação dos resultados com o estado da arte atual. Os resultados obtidos superaram as taxas de identificação previamente relatadas na literatura para os algoritmos criptográficos clássicos AES e Blowfish. O método proposto conseguiu aprimorar as taxas de identificação desses algoritmos em comparação com pesquisas anteriores.

Além de consolidar a utilização dos valores p na composição dos vetores representativos dos criptogramas analisados, conforme abordado nos trabalhos relacionados, os resultados obtidos com essa metodologia foram notáveis no contexto do modo ECB. A pesquisa mencionada em (133) se destaca como um exemplo concreto desses resultados. Outra contribuição significativa refere-se à abordagem adotada para identificar padrões no experimento CBC, a qual também foi aplicada em (126). Ao concentrar-se nos primeiros blocos, mitigou-se a influência da aleatoriedade inerente ao encadeamento de blocos, o que possibilitou a identificação de padrões nos criptogramas gerados nesse modo.

Essa estratégia de identificação abriu caminho para o treinamento de algoritmos de aprendizado capazes de classificar novos criptogramas. No terceiro experimento, que envolveu o KEM, observou-se grande semelhança estatística nas chaves de sessão criptografadas e compartilhadas pelos KEM analisados (Crystals-Kyber, NTRU e Saber), com exceção do KEM Frodo. Embora não tenham sido alcançados níveis de distinção substancialmente superiores ao mero acaso, esse experimento confirmou o nível de indistiguibilidade IND CCA2 do KEM Frodo e IND CCA2 dos KEM Crystals-Kyber, NTRU e Saber.

Nos Experimentos ECB e CBC, o KNN destacou-se, obtendo as mais altas taxas de acurácia em todos os tamanhos de conjunto de dados (20KB, 60KB e 100KB) no ECB

e alcançando uma notável acurácia de 100% no dataset de 20KB no CBC. Enquanto isso, os modelos SVM, NB e RF mantiveram um desempenho consistente, atingindo seu ápice no dataset de 60KB no CBC. O dataset de 100KB revelou novamente que o aumento no tamanho nem sempre beneficia a identificação de criptossistemas.

No Experimento KEM, o KNN também se destacou em datasets menores, mas com taxas de acurácia mais modestas em comparação com os experimentos anteriores. Os modelos SVM, NB e RF mantiveram desempenhos satisfatórios, com taxas de acurácia variando de 52% a 53% no dataset de 20KB e de 45% a 50% no dataset de 100KB. A distinção do Frodo em relação aos outros KEMs contribuiu para as acurácias satisfatórias, e ao restringir a análise aos KEMs da terceira rodada do concurso do NIST (Crystals-Kyber, NTRU e Saber), as métricas se aproximaram do índice aleatório de 33,34%.

Essa pesquisa aponta o KNN como o modelo preferencial para identificar padrões em criptogramas gerados por criptossistemas pós-quânticos, mesmo quando esses criptogramas foram gerados em cenários com chaves e vetores de inicialização aleatórios, em amostras com pelo menos 20KB de tamanho.

O campo de estudo da identificação de padrões em criptogramas ainda apresenta diversas oportunidades de pesquisa. A seguir, enumeram-se possíveis trabalhos futuros:

1. Um problema identificado durante a revisão bibliográfica é a falta de uma base de dados de referência semelhante ao dataset Iris, amplamente utilizado em Aprendizado de Máquina. Portanto, sugerimos a adoção de uma base universal que possa ser cifrada por vários algoritmos, possibilitando a comparação dos resultados de pesquisa. Como mencionado no Capítulo 3, as pesquisas frequentemente utilizaram diferentes bases de dados (texto, áudio, imagens) para seus experimentos de identificação, muitas vezes sem detalhamento nos artigos. Essa base universal deve estar disponível em vários formatos de banco de dados para atender às necessidades dos pesquisadores;

2. Além do texto, outros tipos de dados não estruturados, como áudio e imagens, podem ser analisados para identificar padrões em criptogramas. Essa ampliação do escopo pode levar a descobertas relevantes;

3. Os esquemas de criptografia assimétrica analisados demonstraram um nível de indistinção IND-CPA e foram distinguidos nos dois primeiros experimentos. No entanto, os KEM não apresentaram distinção total. Com base nessas conclusões, recomendam-se modificações nos esquemas de chave pública para aumentar seus níveis de aleatoriedade, aproximando-os do nível de indistinção IND-CCA2. Além disso, outra abordagem que pode ser explorada em pesquisas futuras é a busca por métodos que identifiquem padrões nas chaves de sessão criptografadas e compartilhadas pelos KEMs, permitindo sua identificação com base nessas chaves;

4. A redução de dimensionalidade é um tema de grande relevância a ser explorado em pesquisas futuras. Embora esta pesquisa tenha analisado amostras com um tamanho mínimo de 20 KB, os resultados obtidos indicam que amostras de tamanhos menores podem possuir uma quantidade significativa de dados, permitindo a identificação de padrões por meio de análise estatística. Considerando que trabalhos relacionados analisaram amostras com 4 KB, sugere-se avaliar o impacto no desempenho dos classificadores quando a dimensão dos dados é reduzida. Estudos futuros podem investigar o desempenho de classificadores em amostras ainda menores do que 20KB, buscando identificar o ponto ideal de redução de dimensionalidade, pois é fundamental determinar o ponto ótimo em termos de identificação de criptossistemas. É plausível supor que o classificador KNN possa continuar se beneficiando com essa redução de dimensionalidade, mas é importante conduzir análises específicas para determinar o ponto ideal de redução de dimensionalidade e seu impacto nos resultados;

5. É necessário pesquisar novas formas de representar os criptogramas além das medidas de similaridade, frequência dos *bytes* dos criptogramas (dicionário) e valores p provenientes de testes estatísticos;

6. O processamento de linguagem natural (PLN), um ramo da inteligência artificial, utiliza o aprendizado de máquina para interpretar e processar texto e dados, incluindo tarefas como reconhecimento e geração de linguagem natural. Um sub campo do PLN, conhecido como Compreensão da Linguagem Natural (NLU, em inglês), foca em entender o verdadeiro significado do texto, realizando tarefas como categorização, arquivamento e análise de texto, o que possibilita a tomada de decisões com base nesse significado. O PLN tem amplas aplicações, incluindo a extração de *insights* de dados não estruturados em texto. Dada a capacidade dessas informações em aprofundar a compreensão dos dados, recomenda-se explorar o uso do PLN para identificar padrões e compreender criptogramas, independentemente de serem gerados por algoritmos clássicos ou pós-quânticos. As informações provenientes dessas ferramentas podem complementar abordagens tradicionais, devendo ser analisadas em conjunto com os criptogramas pelos classificadores de IA; e

7. Pesquisas futuras podem utilizar outras técnicas de aprendizado de máquina, além das abordagens tradicionais, para melhorar a identificação de padrões em criptogramas. Redes neurais profundas, aprendizado profundo e aprendizado por reforço são áreas a serem exploradas.

6 APÊNDICE A

Neste Apêndice, encontram-se disponíveis as Tabelas 10, 11, 12, 13 e 14, mencionadas no Capítulo 3. Essas tabelas representam uma compilação minuciosa dos trabalhos relacionados discutidos nesta dissertação, considerando os critérios presentes na Tabela 9.

Tabela 10 – Trabalhos relacionados - 1ª tabela

Pesquisa	Abordagem	Algoritmos de aprendizado de máquina	Hiperparâmetros	Medida de avaliação de desempenho
Dileep e Sekhar(18)	Supervisionada	KNN e SVM	KNN (k = 5, 15 e 25) e SVM (*kernels* linear, polinomial, sigmoide e gaussiano)	Acurácia
Khadivi e Momtazpour(89)	Não há	Não há	Não há	Variância
Khadivi e Momtazpour(90)	Supervisionada	SVM, KNN e LDA	SVM (*kernel* RBF), KNN ((k = 20) e LDA (distância euclidiana)	Acurácia
Sharif, Kuncheva e Mansoor(91)	Supervisionada	NB, SVM, MLP, IBL, Bagging, AdaBoost, RoFo e DT (C4.5)	Não informado	Acurácia
Manjula e Anitha(92)	Supervisionada	DT (C4.5)	Não informado	Acurácia
Souza, Carvalho e Xexeo(93)	Não supervisionada	Hierarchical Clustering	Aglomerativo *bottom-up*	Precisão e *recall*
Lazarin(94)	Não supervisionada	Não há	Não há	Precisão e *recall*
Chou, Lin e Cheng(95)	Supervisionada	SVM	*kernel* gaussiano	Acurácia
Mishra e Bhattacharjya(96)	Supervisionada	DT (C4.5)	*kernel* gaussiano	Acurácia
Tan e Ji(17)	Supervisionada	SVM	Não informado	Acurácia
Souza e Tomlinson(98)	Não supervisionada	Redes neurais	400 neurônios, 10 épocas, taxa de aprendizado de 0.01 e tamanho de vizinhança igual a 1	Quantidade de *clusters*
Lomte e Shinde(99)	Não supervisionada	Redes neurais	400 neurônios, 10 épocas, taxa de aprendizado de 0.09 e tamanho de vizinhança igual a 400	Quantidade de *clusters*
Mello e Xexeo(100)	Supervisionada	C4.5, PART, FT, Complement-NB, MLP e WiSARD	Não informado	Acurácia
Barbosa, Vidal e Mello(101)	Supervisionada	J48, FT, PART, Complement-NB e MLP	Não informado	Acurácia
Barbosa et al.(102)	Supervisionada	J48, FT, PART, Complement-NB e MLP	Não informado	Acurácia
Mello e Xexeo(103)	Supervisionada	C4.5, PART, FT, Complement-NB, MLP e WiSARD	Não informado	Acurácia
Tan, Deng e Zhang(104)	Supervisionada	SVM	Não informado	Acurácia
Pradeepthi, Tiwari e Saxena(105)	Não supervisionada	Expectation-Maximization	Não informado	Quantidade de *clusters*
Fan e Zhao(108)	Supervisionada	RF, LR e SVM	Não informado	Acurácia
Hu e Zhao(109)	Supervisionada	RF	Não informado	Acurácia
Zhang, Zhao e Fan(110)	Supervisionada	RF e MLP	Não informado	Acurácia, precisão, *recall* e *F1-score*
Kavitha et al.(113)	Supervisionada	SVM, Bagging, AdaBoost, Redes Neurais, NB, IBL, RoFo e DT	Não informado	Acurácia
Xia, Li e Chen(28)	Supervisionada	Redes neurais	*ReLU* como função de ativação e *Conv1D* como camada básica de convolução.	Acurácia, precisão e *recall*
Yuan et al.(11)	Supervisionada	RF, SVM, KNN e HRFSVM	Não informado	Acurácia, precisão, *recall* e *F1-score*
Yuan et al.(12)	Supervisionada	RF, SVM, KNN e HKNNRF	Não informado	Acurácia, precisão, *recall* e *F1-score*
Yu e Shi(114)	Supervisionada	MLP	Não informado	Acurácia, precisão, *recall* e *F1-score*

Tabela 11 – Trabalhos relacionados - 2ª tabela

Algoritmo de AM	**Pesquisa**
AdaBoost	(91) (113)
Bagging	(91) (113)
Complement-NB	(100) (101) (102) (103)
DT (C4.5)	(91) (92) (96) (100) (101) (102) (103) (113)
Expectation-Maximization	(105)
FT	(100) (101) (102) (103)
Hierarchical Clustering	(93)
IBL	(91) (113)
KNN	(18) (90) (11) (12)
LDA	(90)
LR	(108)
MLP	(91) (100) (101) (102) (103) (110) (114)
NB	(91) (113)
PART	(100) (101) (102) (103)
Redes neurais	(98) (99) (113) (28)
RF	(108) (109) (110) (11) (12)
RoFo	(91) (113)
SVM	(18) (90) (91) (95) (17) (104) (108) (113) (11) (12)
WiSARD	(100) (103)

Abordagem	**Pesquisa**
Não-supervisionada	(93) (94) (98) (99) (105)
Supervisionada	(18) (90) (91) (92) (95) (96) (17) (100) (101) (102) (103) (104) (108) (109) (110) (113) (28) (11) (12) (114)

Tabela 12 – Trabalhos relacionados - 3ª tabela

Pesquisa	Algoritmos de criptografia	Modo de operação	Chave(s)	Vetore(s) de inicialização	Quantidade de amostras	Tamanhos das amostras	Acurácia	Tipo de criptografia
(18)	DES, 3DES, Blowfish, AES, RC5	DES nos modos ECB e CBC; demais no modo ECB.	Distintas	Não informado	100	4000 bits	95%	Clássica
(89)	AES, Substituição clássico	ECB	Não informado	Não há	118	320 bytes	100%	Clássica
(90)	AES	Não informado	Não informado	Não informado	6400	Não informado	65%	Clássica
(91)	DES, IDEA, AES e RC2	ECB	Distintas	Não há	240	512 KB	53,33%	Clássica
(92)	RC2, RC4, DES, 3DES, Blowfish, ECC, IDEA, RSA e AES	Não informado	Não informado	Não informado	2600	1 KB a 3 MB	75%	Clássica
(93)	MARS, RC6, Rijndael, Serpent e Twofish	ECB e CBC	Iguais	Não informado	1350	128 bits	Não informado	Clássica
(94)	DES, Blowfish, Camellia, AES, SEED, RC4, CAST, RC2 e 3DES	CBC	Distintas	Não informado	500	32 KB	72%	Clássica
(95)	AES e DES	ECB e CBC	Iguais	Distintos	3000	Não informado	100%; 52,55%	Clássica
(96)	AES, DES e Blowfish	ECB	Distintas	Não há	1000, 200, 700 e 2000	28, 256, 512 e 1024 bits	83%, 64%, 87,3% e 89,1%	Clássica
(17)	AES, Blowfish, 3DES, RC5 e DES	ECB	Iguais, distintas	Não há	1100	0.8 KB, 4 KB, 20 KB, 100 KB e 500 KB	90%	Clássica
(98)	MARS, RC6, Rijndael, Serpent e Twofish	ECB	Iguais	Não há	30	8192 bytes	100%	Clássica
(99)	MARS, RC6, Rijndael, Serpent e Twofish	Não informado	Iguais	Não há	45	8192 bytes	100%	Clássica
(100)	RC4, Blowfish, DES, Rijdael, RSA, Serpent e Twofish	ECB	Distintas	Não há	600	2 bits até 34 bits	100%	Clássica
(101)	DES, Blowfish, RC4 e RSA	Não informado	Não informado	Não informado	200	4 bits a 16 bits	100%	Clássica
(102)	DES, Blowfish, RC4 e RSA	ECB	Distintas	Não há	500	4 bits a 16 bits	100%	Clássica
(103)	DES, Blowfish, RSA, ARC4, Rijndael, Serpent e Twofish	ECB e CBC	Distintas	Não informado	600	2 bits até 34 bits	100%; 50%	Clássica
(104)	AES, DES, 3DES, RC5 e Blowfish	CBC	Iguais, distintas	Iguais, distintos	200	4 KB, 20 KB, 100 KB e 500 KB	97%	Clássica
(105)	DES e L-Block	Não informado	Não informado	Não informado	50	Não informado	100%	Clássica
(108)	DES, 3DES, AES-128, AES-256, IDEA, SMS4, Blowfish e Camellia	ECB e CBC	Iguais	Não informado	16016	512 KB	75%; 13,5%	Clássica
(109)	AES-128, AES-256, Blowfish, Camelia, DES, 3DES, IDEA e SMS4	ECB e CBC	Iguais	Não informado	16016	512 KB	87,9%; 12,64%	Clássica
(110)	AES-128, AES-256, Blowfish, Camellia, DES, 3DES, IDEA e SMS4	ECB e CBC	Distintas	Distintos	2000	512 KB	84,5%, 14%	Clássica
(113)	DES, IDEA, AES (128, 192 e 256 bits) e RC2 (42, 84 e 128 bits)	ECB	Distintas	Não há	200	512 KB	54,32%	Clássica
(28)	AES, Kasumi, 3DES, Present, RSA e ElGamal	CBC	Distintas	Não informado	1000	1,1 MB	95,72%	Clássica
(11)	AES, DES, Blowfish, Camellia, RC2 e RC6	ECB	Iguais	Não há	2500	1 KB, 8 KB, 64 KB, 256 KB e 512 KB	72,5%	Clássica
(12)	AES, 3DES, Blowfish, Cast e RC2	ECB	Iguais	Não há	2500	1 KB, 8 KB, 64 KB, 256 KB e 512 KB	74%	Clássica
(114)	DES, 3DES, AES e Blowfish	ECB e CBC	Iguais	Não informado	4000	256 KB	42%, 29,6%	Clássica

Tabela 13 – Trabalhos relacionados - 4ª tabela

Pesquisa	Algoritmo de criptografia	Modo de representação dos vetores	Tipo de cifra	Tipo de criptografia
Dileep e Sekhar(18)	DES, 3DES, Blowfish, AES e RC5	*bag-of-words*	Cifra de bloco	Simétrica
Khadivi e Momtazpour(89)	AES e Substituição clássica	*data mining*	Cifra de bloco, cifra de fluxo	Simétrica
Khadivi e Momtazpour(90)	AES	*bag-of-words*	Cifra de bloco	Simétrica
Sharif, Kuncheva e Mansoor(91)	DES, IDEA, AES e RC2	*bag-of-words*	Cifra de bloco	Simétrica
Manjula e Anitha(92)	RC2, RC4, DES, 3DES, Blowfish, ECC, IDEA, RSA e AES	Índices estatísticos	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Souza, Carvalho e Xexeo(93)	MARS, RC6, Rijndael, Serpent e Twofish	*bag-of-words*	Cifra de bloco	Simétrica
Lazarin(94)	DES, Blowfish, Camellia, AES, SEED, RC4, CAST, RC2 e 3DES	Índices estatísticos, distância de Hamming	Cifra de bloco, cifra de fluxo	Simétrica
Chou, Lin e Cheng(95)	AES e DES	Índices estatísticos	Cifra de bloco	Simétrica
Mishra e Bhattacharjya(96)	AES, DES e Blowfish	*bag-of-words*	Cifra de bloco	Simétrica
Tan e Ji(17)	AES, Blowfish, 3DES, RC5 e DES	*bag-of-words*	Cifra de bloco	Simétrica
Souza e Tomlinson(98)	MARS, RC6, Rijndael, Serpent e Twofish	*bag-of-words*	Cifra de bloco	Simétrica
Lomte e Shinde(99)	MARS, RC6, Rijndael, Serpent e Twofish	*bag-of-words*	Cifra de bloco	Simétrica
Mello e Xexeo(100)	RC4, Blowfish, DES, Rijdael, RSA, Serpent e Twofish	Histogramas	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Barbosa, Vidal e Mello(101)	DES, Blowfish, RC4 e RSA	Histogramas	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Barbosa et al.(102)	DES, Blowfish, RC4 e RSA	Histogramas	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Mello e Xexeo(103)	DES, Blowfish, RSA, ARC4, Rijndael, Serpent e Twofish	Histogramas	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Tan, Deng e Zhang(104)	AES, DES, 3DES, RC5 e Blowfish	Não informado	Cifra de bloco	Simétrica
Pradeepthi, Tiwari e Saxena(105)	DES e L-Block	Arquivos diferença (operação XOR)	Cifra de bloco	Simétrica
Fan e Zhao(108)	DES, 3DES, AES-128, AES-256, IDEA, SMS4, Blowfish e Camellia	*bag-of-words*	Cifra de bloco	Simétrica
Hu e Zhao(109)	AES-128, AES-256, Blowfish, Camelia, DES, 3DES, IDEA e SMS4	*bag-of-words*	Cifra de bloco	Simétrica
Zhang, Zhao e Fan(110)	AES-128, AES-256, Blowfish, Camellia, DES, 3DES, IDEA e SMS4	*bag-of-words*	Cifra de bloco	Simétrica
Kavitha et al.(113)	DES, IDEA, AES (128, 192 e 256 bits) e RC2 (42, 84 e 128 bits)	*bag-of-words*	Cifra de bloco	Simétrica
Xia, Li e Chen(28)	AES, Kasumi, 3DES, Present, RSA e ElGamal	Testes estatísticos do NIST	Cifra de bloco, cifra de fluxo	Simétrica, assimétrica
Yuan et al.(11)	AES, DES, Blowfish, Camellia, RC2 e RC6	Testes estatísticos do NIST	Cifra de bloco	Simétrica
Yuan et al.(12)	AES, 3DES, Blowfish, Cast e RC2	Testes estatísticos do NIST	Cifra de bloco	Simétrica
Yu e Shi(114)	AES, Blowfish, DES e 3DES	Testes estatísticos do NIST	Cifra de bloco	Simétrica

Tabela 14 – Trabalhos relacionados - 5ª tabela

Algoritmo de criptografia	**Pesquisa**
AES	(18) (89) (90) (91) (92) (94) (95) (96) (17) (104) (108) (109) (110) (113) (28) (11) (12) (114)
Blowfish	(18) (92) (94) (96) (17) (101) (100) (102) (103) (104) (105) (108) (109) (110) (11) (12) (114)
CAST	(94) (12)
Camellia	(94) (108) (109) (110) (11)
DES	(18) (91) (92) (94) (95) (96) (17) (100) (101) (102) (103) (104) (109) (108) (110) (113) (11) (114)
3DES	(18) (92) (94) (17) (104) (109) (108) (110) (28) (12) (114)
El Gamal	(28)
Kasumi	(28)
L-Block	(105)
MARS	(93) (98) (99)
Present	(28)
RC2	(91) (92) (94) (113) (11) (12)
RC4	(92) (94) (100) (101) (102) (103)
RC5	(18) (17) (104)
RC6	(93) (98) (99) (11)
Rijndael	(93) (98) (99) (100) (103)
Serpent	(93) (98) (99) (100) (103)
SEED	(94)
SMS4	(109) (110)
Substituição clássico	(89)
Twofish	(93) (98) (99) (100) (103)
IDEA	(91) (92) (109) (108) (110) (113)
ECC	(92)
RSA	(92) (100) (101) (102) (103) (28)

7 APÊNDICE B

Neste apêndice, encontram-se reunidos todos os resultados mencionados no capítulo 4, organizados em tabelas para uma visualização mais fácil e acessível.

Tabela 15 – Resultados consolidados Experimento ECB - Estratégia *train-test split*

Dataset (KB)	**Classe**	**Total de Amostras**	**Total Classificado Corretamente**	**Precisão (%)**	**Recall (%)**	**F1-Score (%)**
20	AES	30	30 (KNN, NB, SVM, RF)	100	100	100
	Blowfish	30	30 (KNN, NB, SVM, RF)	100	100	100
	Frodo e NTRU	30	30 (KNN, NB, SVM, RF)	100	100	100
	Kyber	30	30 (KNN, NB, SVM, RF)	100	100	100
	Saber	30	30 (KNN, NB, RF)	100	100	100
			7 (SVM)	23	23	38
60	AES	30	26 (KNN)	93	87	90
			21 (NB)	60	70	65
			23 (SVM)	66	77	71
			18 (RF)	60	70	65
	Blowfish	30	28 (KNN)	88	93	90
			16 (NB)	64	53	58
			18 (SVM)	72	60	65
			18 (RF)	60	70	65
	Frodo, NTRU, Kyber	30	30 (KNN, NB, SVM, RF)	100	100	100
	Saber		30 (RF, NB)	100	100	100
			27 (KNN)	90	90	90
			25 (SVM)	100	83	91
100	AES	30	13 (KNN)	45	43	44
			7 (NB)	41	23	30
			3 (SVM)	38	10	16
			11 (RF)	39	37	38
	Blowfish	30	15 (KNN, SVM, RF)	50	50	50
			11 (NB)	58	37	45
	Kyber	30	4 (KNN)	24	13	17
			23 (NB)	43	77	55
			19 (SVM)	43	63	51
			11 (RF)	44	37	40
	Frodo, NTRU	30	30 (KNN, NB, SVM, RF)	100	100	100
		30	29 (KNN, NB, RF)	100	97	98
	Saber		29 (SVM)	100	97	98

Tabela 16 – Resultados consolidados Experimento ECB - Estratégia *cross-validation*

Dataset (KB)	Classe	Total de Amostras	Total Classificado Corretamente	Precisão (%)	Recall (%)	F1-Score (%)
20	AES	100	100 (KNN)	100	100	100
			58 (NB)	82	58	68
			81 (SVM)	64	81	72
			50 (RF)	57	50	53
	Blowfish	100	99 (KNN)	99	99	99
			81 (NB)	60	81	69
			65 (SVM)	67	65	66
			49 (RF)	48	49	48
	Frodo, NTRU, Saber	100	100 (KNN, NB, SVM, RF)	100	100	100
			34 (SVM para Saber)	34	34	51
	Kyber	100	99 (KNN)	100	99	99
			86 (NB)	91	86	89
			86 (SVM)	89	86	87
			56 (RF)	50	56	53
60	AES	100	98 (KNN)	95	97	96
			72 (NB)	55	72	63
			98 (SVM)	52	98	68
			50 (RF)	48	50	49
	Blowfish	100	96 (KNN)	97	96	96
			42 (NB)	60	42	49
			12 (SVM)	80	12	21
			47 (RF)	48	47	48
	Frodo, NTRU	100	100 (KNN, NB, SVM, RF)	100	100	100
	Saber	100	100 (KNN, NB, RF)	100	100	100
			77 (SVM)	-	77	87
	Kyber	100	100 (KNN, NB, SVM, RF)	100	100	100
100	AES	100	52 (KNN)	41	52	46
			13 (NB)	27	13	18
			44 (SVM)	33	44	37
			47 (RF)	41	47	44
	Blowfish	100	41 (KNN)	35	41	38
			29 (NB)	38	29	33
			49 (SVM)	38	49	43
			37 (RF)	46	37	41
	Kyber	100	23 (KNN)	40	23	29
			63 (NB)	36	63	46
			8 (SVM)	22	8	12
			44 (RF)	42	44	43
	Frodo, NTRU, Saber	100	100 (KNN, NB, SVM, RF)	100	100	100
			34 (SVM para Saber)	34	34	51

Tabela 17 – Resultados consolidados Experimento CBC - Estratégia *train-test split*

Dataset (KB)	Classe	Total de Amostras	Total Classificado Corretamente	Precisão (%)	Recall (%)	F1-Score (%)
20	AES	30	30 (KNN)	100	100	100
			29 (NB)	97	98	97
			16 (RF)	64	53	58
	Blowfish	30	30 (KNN)	100	100	100
			6 (NB)	86	20	32
			21 (RF)	60	70	65
	Frodo, Kyber, NTRU, Saber	30	30 (KNN, NB, SVM, RF)	100	100	100
60	AES	30	27 (KNN)	100	90	95
			22 (NB)	53	73	62
			13 (SVM)	68	43	53
			19 (RF)	47	63	54
	Blowfish	30	30 (KNN)	100	100	100
			11 (NB)	58	37	45
			24 (SVM)	59	80	68
			11 (RF)	45	30	36
	Frodo, Kyber, NTRU, Saber	30	30 (KNN, NB, SVM, RF)	100	100	100
100	AES	30	17 (KNN)	33	57	42
			4 (NB)	31	13	19
			13 (SVM)	36	43	39
			9 (RF)	30	30	32
	Blowfish	30	11 (KNN)	44	37	40
			8 (NB)	40	27	32
			13 (SVM)	39	43	41
			9 (RF)	30	30	31
	Kyber	30	4 (KNN)	29	13	18
			19 (NB)	33	0	44
			6 (SVM)	29	20	24
			14 (RF)	40	40	40
	Frodo, NTRU, Saber	30	30 (KNN, NB, SVM, RF)	100	100	100

Tabela 18 – Resultados consolidados Experimento CBC - Estratégia *cross-validation*

Dataset (KB)	Classe	Total de Amostras	Total Classificado Corretamente	Precisão (%)	Recall (%)	F1-Score (%)
20	AES	30	30 (KNN)	100	100	100
			96 (NB)	57	96	72
			81 (SVM)	92	81	86
			63 (RF)	59	63	61
	Blowfish	30	99 (KNN)	100	99	99
			29 (NB)	88	29	44
			93 (SVM)	83	93	88
			56 (RF)	60	56	58
	Frodo, Kyber, NTRU, Saber	100	100 (KNN, NB, SVM, RF)	100	100	100
60	AES	30	95 (KNN)	98	95	96
			71 (NB)	61	71	65
			73 (SVM)	75	73	74
			56 (RF)	52	56	54
	Blowfish	30	98 (KNN)	95	98	97
			54 (NB)	65	54	59
			76 (SVM)	74	76	75
			48 (RF)	52	48	50
	Frodo, Kyber, NTRU, Saber	100	100 (KNN, NB, SVM, RF)	100	100	100
100	AES	30	52 (KNN)	41	52	46
			13 (NB)	27	13	18
			44 (SVM)	33	44	37
			47 (RF)	41	47	44
	Blowfish	30	41 (KNN)	35	41	38
			29 (NB)	38	29	33
			49 (SVM)	38	49	43
			37 (RF)	46	37	41
	Kyber	30	23 (KNN)	40	23	29
			63 (NB)	36	63	46
			8 (SVM)	22	8	12
			44 (RF)	42	44	43
	Frodo, NTRU, Saber	100	100 (KNN, NB, SVM, RF)	100	100	100

Tabela 19 – Resultados consolidados Experimento KEM - Estratégia *train-test split*

Dataset (KB)	Classe	Total de Amostras	Total Classificado Corretamente	Precisão (%)	Recall (%)	F1-Score (%)
20	Kyber	100	23 (KNN)	40	23	29
			63 (NB)	36	63	46
			8 (SVM)	22	8	12
			44 (RF)	42	44	43
	Frodo	30	30 (KNN, NB, SVM, RF)	100	100	100
	NTRU	100	17 (KNN)	49	57	52
			3 (NB)	33	10	15
			13 (SVM)	38	43	41
			9 (RF)	38	50	43
	Saber	100	5 (KNN)	29	17	21
			11 (NB)	34	37	35
			17 (SVM)	37	57	45
			15 (RF)	38	50	43
60	Kyber	100	23 (KNN)	40	23	29
			63 (NB)	36	63	46
			8 (SVM)	22	8	12
			44 (RF)	42	44	43
	Frodo	30	30 (KNN, NB, SVM, RF)	100	100	100
	NTRU	100	17 (KNN)	49	57	52
			3 (NB)	33	10	15
			13 (SVM)	38	43	41
			9 (RF)	38	50	43
	Saber	100	5 (KNN)	29	17	21
			11 (NB)	34	37	35
			17 (SVM)	37	57	45
			15 (RF)	38	50	43
100	Kyber	100	12 (KNN)	29	40	34
			6 (RF)	19	20	20
	Frodo	30	30 (KNN, NB, SVM, RF)	100	100	100
	NTRU	100	6 (KNN)	21	20	21
			6 (RF)	30	20	24
	Saber	30	10 (KNN)	50	33	40
			12 (RF)	31	40	35
			30 (NB, SVM)	100	100	100

Tabela 20 – Resultados consolidados Experimento KEM - Estratégia *cross-validation*

Dataset (KB)	**Classe**	**Total de Amostras**	**Total Classificado Corretamente**	**Precisão (%)**	**Recall (%)**	**F1-Score (%)**
20	Kyber	100	50 (KNN)	36	50	42
			74 (NB)	34	74	47
			11 (SVM)	28	11	16
			41 (RF)	36	41	38
	Frodo	100	100 (KNN, NB, SVM, RF)	100	100	100
	NTRU	100	38 (KNN)	40	38	39
			11 (NB)	42	11	17
			51 (SVM)	36	51	42
			31 (RF)	38	31	34
	Saber	100	17 (KNN)	26	17	21
			18 (NB)	32	18	23
			46 (SVM)	38	46	42
			39 (RF)	37	39	38
60	Kyber	100	44 (KNN)	34	44	38
			45 (NB)	26	45	33
			53 (SVM)	37	53	43
			50 (RF)	37	50	43
	Frodo	100	100 (KNN, NB, SVM, RF)	100	100	100
	NTRU	100	28 (KNN)	33	28	30
			17 (NB)	22	17	19
			53 (SVM)	35	53	42
			32 (RF)	36	32	34
	Saber	100	37 (KNN)	44	37	40
			14 (NB)	28	14	19
			2 (SVM)	50	2	4
			26 (RF)	42	26	36
100	Kyber	100	36 (KNN)	31	36	33
			0 (NB, SVM)	0	0	0
			52 (RF)	38	52	44
	Frodo	100	100 (KNN, SVM, RF)	100	100	100
			99 (NB)	99	99	99
	NTRU	100	24 (KNN)	23	24	24
			0 (NB, SVM)	0	0	0
			23 (RF)	28	23	25
	Saber	100	33 (KNN)	43	33	37
			99 (NB)	34	99	51
			100 (SVM)	-	100	-
			36 (RF)	44	36	40

REFERÊNCIAS

1 STALLINGS, W. *Criptografia e segurança de redes. Princípios e práticas, ch. 6.* [S.l.]: Pearson Prentice Hall, 2006.

2 BOS, J.; DUCAS, L.; KILTZ, E.; LEPOINT, T.; LYUBASHEVSKY, V.; SCHANCK, J. M.; SCHWABE, P.; SEILER, G.; STEHLÉ, D. Crystals-kyber: a cca-secure module-lattice-based kem. In: IEEE. *2018 IEEE European Symposium on Security and Privacy (EuroS&P).* [S.l.], 2018. p. 353–367.

3 NGO, K.; DUBROVA, E.; JOHANSSON, T. Breaking masked and shuffled cca secure saber kem by power analysis. In: *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security.* [S.l.: s.n.], 2021. p. 51–61.

4 JR, M. F.; KIPPEN, H.; KWONG, A.; DANG, T.; LICHTINGER, J.; DACHMAN-SOLED, D.; GENKIN, D.; NELSON, A.; PERLNER, R.; YERUKHIMOVICH, A. et al. When frodo flips: End-to-end key recovery on frodokem via rowhammer. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* [S.l.: s.n.], 2022. p. 979–993.

5 LIU, B.; WU, H. Efficient architecture and implementation for ntruencrypt system. In: IEEE. *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS).* [S.l.], 2015. p. 1–4.

6 MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, v. 1, n. 1, p. 32, 2003.

7 CHANDRA, M. A.; BEDI, S. Survey on svm and their application in image classification. *International Journal of Information Technology*, Springer, v. 13, p. 1–11, 2021.

8 HUANG, S.; DU, Q.; ZHANG, H. A hybrid svm approach for classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 40, n. 11, p. 2659–2666, 2002.

9 MELGANI, F.; BRUZZONE, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, v. 42, n. 8, p. 1778–1790, 2004.

10 TELOKEN, A. et al. Estudo comparativo entre os algoritmos de mineração de dados random forest e j48 na tomada de decisão. *Simpósio de Pesquisa e Desenvolvimento em Computação*, v. 2, n. 1, 2016.

11 YUAN, K.; HUANG, Y.; LI, J.; JIA, C.; YU, D. A block cipher algorithm identification scheme based on hybrid random forest and logistic regression model. *Neural Processing Letters*, Springer, p. 1–19, 2022.

12 YUAN, K.; YU, D.; FENG, J.; YANG, L.; JIA, C.; HUANG, Y. A block cipher algorithm identification scheme based on hybrid k-nearest neighbor and random forest algorithm. *PeerJ Computer Science*, PeerJ Inc., v. 8, p. e1110, 2022.

13 DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. *IEEE transactions on Information Theory*, IEEE, v. 22, n. 6, p. 644–654, 1976.

14 DIFFIE, W.; HELLMAN, M. E. *Diffie-Hellman Key Exchange*. [S.l.]: WIKIPEDIA, 1976.

15 RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, v. 21, n. 2, p. 120–126, Feb. 1978.

16 SCHNEIER, B. *Applied cryptography: protocols, algorithms, and source code in C*. [S.l.]: john wiley & sons, 2007.

17 TAN, C.; JI, Q. An approach to identifying cryptographic algorithm from ciphertext. In: IEEE. *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. [S.l.], 2016. p. 19–23.

18 DILEEP, A. D.; SEKHAR, C. C. Identification of block ciphers using support vector machines. In: IEEE. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. [S.l.], 2006. p. 2696–2701.

19 PFLEEGER, S. L. *Engenharia de software: teoria e prática*. [S.l.]: Prentice Hall, 2004.

20 KNUDSEN, L. R.; MEIER, W. Correlations in rc6 with a reduced number of rounds. In: SPRINGER. *Fast Software Encryption: 7th International Workshop, FSE 2000 New York, NY, USA, April 10–12, 2000 Proceedings 7*. [S.l.], 2001. p. 94–108.

21 NAGIREDDY, S. *A Pattern Recognition Approach to Block Cipher Identification. Master of Science Dissertation–Indian Institute of Technology Madras*. 2008.

22 DING, J.; GOWER, J. E.; STINSON, D. R. Post-quantum cryptography. *Cryptography and Communications*, Springer, v. 6, n. 1, p. 1–3, 2014.

23 ALAGIC, G.; APON, D.; COOPER, D.; DANG, Q.; DANG, T.; KELSEY, J.; LICHTINGER, J.; MILLER, C.; MOODY, D.; PERALTA, R. et al. Status report on the third round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2022.

24 BERNSTEIN, D. J.; HÜLSING, A. T.; LANGE, T. Post-quantum cryptography-integration study. ENISA, 2022.

25 CACRNET. *The Chinese Academy of Cyberspace Research Network*. Disponível em: <https://www.cacrnet.org.cn/site/content/838.html>.

26 PACHECO, R.; BRAGA, D.; PASSOS, I.; ARAÚJO, T.; LAGROTA, V.; COUTINHO, M. libharpia: a new cryptographic library for brazilian elections. In: SBC. *Anais do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2022. p. 250–263.

27 MOSCA, M. Post-quantum cryptography: From the underground to the spotlight. *Notices of the American Mathematical Society*, JSTOR, v. 62, n. 5, p. 526–529, 2015.

28 XIA, R.; LI, M.; CHEN, S. Cryptographic algorithms identification based on deep learning. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 11, n. 6, p. 2495–2503, 2020.

29 YU, X.; SHI, K. Block ciphers identification scheme based on randomness test. In: SPIE. *6th International Workshop on Advanced Algorithms and Control Engineering (IWAACE 2022)*. [S.l.], 2022. v. 12350, p. 375–380.

30 RUKHIN, A.; SOTO, J.; NECHVATAL, J.; SMID, M.; BARKER, E.; LEIGH, S.; LEVENSON, M.; VANGEL, M.; BANKS, D.; HECKERT, A. et al. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. [S.l.], 2001.

31 TECNOLóGICA, I. *IBM apresenta computador quântico com 433 qubits, o maior do mundo*. 2021. <https://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=ibm-apresenta-computador-quantico-433-qubits-maior-mundo&id=020150221109>. Accessed on: 2023-05-08.

32 YAN, B.; TAN, Z.; WEI, S.; JIANG, H.; WANG, W.; WANG, H.; LUO, L.; DUAN, Q.; LIU, Y.; SHI, W. et al. Factoring integers with sublinear resources on a superconducting quantum processor. *arXiv preprint arXiv:2212.12372*, 2022.

33 KÖLBL, S. *Design and analysis of cryptographic algorithms*. Tese (Doutorado) — Ph. D Thesis. Lyngby, 2017.

34 WAZLAWICK, R. S. *Metodologia de pesquisa para ciência da computação*. [S.l.]: Elsevier, 2009. v. 2.

35 CRAMER, R.; SHOUP, V. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, SIAM, v. 33, n. 1, p. 167–226, 2003.

36 CARVALHO, C. d. *O uso de técnicas de recuperação de informações em criptoanálise*. [S.l.]: Instituto Militar de Engenharia, 2006.

37 DWORKIN, M. *Recommendation for block cipher modes of operation. methods and techniques*. [S.l.], 2001.

38 SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, SIAM, v. 41, n. 2, p. 303–332, 1999.

39 BOS, J. W.; COSTELLO, C.; DUCAS, L.; MIRONOV, I.; NAEHRIG, M.; NIKOLAENKO, V.; RAGHUNATHAN, A. FrodoKEM: Learning with errors key encapsulation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, v. 2017, n. 4, p. 46–69, 2017. Disponível em: <https://tches.iacr.org/index.php/TCHES/article/view/7425>.

40 HOFFSTEIN, J. Ntru: a new high speed public key cryptosystem. *presented at the rump session of Crypto 96*, 1996.

41 D'ANVERS, J.-P.; KARMAKAR, A.; ROY, S. S.; VERCAUTEREN, F. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: SPRINGER. *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10*. [S.l.], 2018. p. 282–305.

42 REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 56, n. 6, p. 1–40, 2009.

43 STANDARDS, N. I. of; TECHNOLOGY. *FIPS 203 (Draft) - Module-Lattice-based Key-Encapsulation Mechanism Standard*. 2023. Disponível em: <https://doi.org/10.6028/NIST.FIPS.203.ipd>.

44 COSTA, S. I.; OGGIER, F.; CAMPELLO, A.; BELFIORE, J.-C.; VITERBO, E. *Lattices applied to coding for reliable and secure communications*. [S.l.]: Springer, 2017.

45 BOLLAUF, M. F. *Códigos, reticulados e aplicações em criptografia*. Tese (Doutorado) — Master's thesis, Universidade Estadual de Campinas, Campinas-SP, Brasil, 2015.

46 AJTAI, M. Generating hard instances of lattice problems. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1996. p. 99–108.

47 HARRIGAN, S. Lattice-based cryptography and the learning with errors problem.

48 BRAKERSKI, Z.; LANGLOIS, A.; PEIKERT, C.; REGEV, O.; STEHLÉ, D. Classical hardness of learning with errors. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 2013. p. 575–584.

49 LYUBASHEVSKY, V.; PEIKERT, C.; REGEV, O. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 60, n. 6, p. 1–35, 2013.

50 LANGLOIS, A.; STEHLÉ, D. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, Springer, v. 75, n. 3, p. 565–599, 2015.

51 YAGISAWA, M. Fully homomorphic encryption without bootstrapping. *Cryptology ePrint Archive*, 2015.

52 MALIK, M.; DUTTA, M.; GRANJAL, J. A survey of key bootstrapping protocols based on public key cryptography in the internet of things. *IEEE Access*, IEEE, v. 7, p. 27443–27464, 2019.

53 BANERJEE, A.; PEIKERT, C.; ROSEN, A. Pseudorandom functions and lattices. In: SPRINGER. *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*. [S.l.], 2012. p. 719–737.

54 BSI, C. M. Cryptographic mechanisms: Recommendations and key lengths. *BSI—Technical Guideline*, 2020.

55 TORRE, M. Á. González de la; ENCINAS, L. H.; QUEIRUGA-DIOS, A. About the frodokem lattice-based algorithm. 2022.

56 NIELSEN, J.; RIJMEN, V. *EUROCRYPT 2018, Part I. LNCS, vol. 10820*. [S.l.]: Springer, Cham, 2018.

57 RESENDE, A.; COSTA, V. da. Utilização de testes estatísticos para verificação de eficácia de algoritmos criptográficos. In: *IX Encontro Anual de Computação - ENACOMP*. Catalão-GO, Brasil: [s.n.], 2011.

58 L'ECUYER, P. Uniform random number generation. *Handbooks in operations research and management science*, Elsevier, v. 13, p. 55–81, 2006.

59 GUTMANN, P. Software generation of random numbers for cryptographic purposes. In: *Proceedings of the 1998 Usenix Security Symposium*. [S.l.: s.n.], 1998. p. 243–257.

60 DAVIS, D.; IHAKA, R.; FENSTERMACHER, P. Cryptographic randomness from air turbulence in disk drives. In: SPRINGER. *Advances in Cryptology—CRYPTO'94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings 14*. [S.l.], 1994. p. 114–120.

61 ZIMMERMANN, P. *PGP source code and internals*. [S.l.]: Mit Press, 1995.

62 PRESS, W. H.; FLANNERY, B. P.; TEUKOLSKY, S. A.; VETTERLING, W. T. *Numerical recipes in C: The art of scientific computing*. 2nd. ed. [S.l.]: Cambridge Univ Press, 1992.

63 MARSAGLIA, G. A current view of random number generators. In: ELSEVIER SCIENCE PUBLISHERS, NORTH-HOLLAND. *Computer Science and Statistics, Sixteenth Symposium on the Interface*. Amsterdam, 1985. p. 3–10.

64 VIEIRA, C. E. C.; SOUZA, R. d. C. e; RIBEIRO, C. C. *Um estudo comparativo entre três geradores de números aleatórios*. [S.l.]: PUC, 2004.

65 KIM, S. H.; HAN, D.; LEE, D. H. Predictability of android openssl's pseudo random number generator. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. [S.l.: s.n.], 2013. p. 659–668.

66 MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw Hill, 1997.

67 FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. d. L. F. d. *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. [S.l.]: Bookman Editora, 2011.

68 CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. Semi-supervised learning. *Chapelle, O. and Schölkopf, B. and Zien, A.(2006). Semi-supervised learning.*, v. 2, p. 1–49, 2006.

69 CHEESEMAN, P.; STUTZ, J.; HANSON, R.; TAYLOR, W. Autoclass iii. *Research Institute for Advanced Computer Science, NASA Ames Research Centre, USA*, 1990.

70 SOLANKI, V.; TANWAR, S. K. Machine learning for cryptanalysis. *Security and Communication Networks*, v. 2018, 2018.

71 AHA, D. W.; KIBLER, D. *Instance-based learning algorithms*. [S.l.]: Morgan Kaufmann, 1991.

72 VAPNIK, V. N. *The nature of statistical learning theory*. [S.l.]: Springer, 1995.

73 GUYON, I.; WESTON, J.; BARNHILL, S.; VAPNIK, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, v. 46, p. 389–422, 2002.

74 PRAJAPATI, G. L.; PATLE, A. On performing classification using svm with radial basis and polynomial kernel functions. In: *Third International Conference on Emerging Trends in Engineering and Technology*. [S.l.: s.n.], 2010. p. 512–515.

75 KUO, B.-C.; HO, H.-H.; LI, C.-H.; HUNG, C.-C.; TAUR, J.-S. A kernel-based feature selection method for svm with rbf kernel for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, v. 7, n. 1, p. 317–326, 2014.

76 CHANG, C.-C.; LIN, C.-J. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM New York, NY, USA, v. 2, n. 3, p. 27, 2011.

77 SCHÖLKOPF, B.; SMOLA, A. J. Advances in kernel methods: Support vector machines. *MIT press Cambridge*, v. 5, n. 8, p. 16, 1999.

78 BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, p. 123–140, 1996.

79 BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, p. 5–32, 2001.

80 NETO, C. G. Potencial de técnicas de mineraçao de dados para o mapeamento de áreas cafeeiras. *Instituto Nacional de Pesquisas Espaciais*, 2014.

81 AZAR, A. T.; ELSHAZLY, H. I.; HASSANIEN, A. E.; ELKORANY, A. M. A random forest classifier for lymph diseases. *Computer methods and programs in biomedicine*, Elsevier, v. 113, n. 2, p. 465–473, 2014.

82 LEITE, D. R. A.; MORAES, R. M. de; LOPES, L. W. Método de aprendizagem de máquina para classificação da intensidade do desvio vocal utilizando random forest. *Journal of Health Informatics*, v. 12, 2020.

83 SCHNEIER, B. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. [S.l.]: Addison-Wesley, 1996.

84 STINSON, D. R. *Cryptography: Theory and Practice*. 3. ed. [S.l.]: CRC Press, 2006.

85 STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 5. ed. [S.l.]: Prentice Hall, 2010.

86 FERGUSON, N.; SCHNEIER, B.; KOHNO, T. *Cryptography Engineering: Design Principles and Practical Applications*. [S.l.]: Wiley, 2010.

87 MENEZES, A. J.; OORSCHOT, P. C. van; VANSTONE, S. A. *Handbook of Applied Cryptography*. [S.l.]: CRC Press, 1996.

88 SOUZA, W. d. Identificação de padrões em criptogramas usando técnicas de classificação de textos. *Rio de Janeiro: Instituto Militar de Engenharia*, 2007.

89 KHADIVI, P.; MOMTAZPOUR, M. Application of data mining in cryptanalysis. In: IEEE. *2009 9th International Symposium on Communications and Information Technology*. [S.l.], 2009. p. 358–363.

90 KHADIVI, P.; MOMTAZPOUR, M. Cipher-text classification with data mining. In: IEEE. *2010 IEEE 4th International Symposium on Advanced Networks and Telecommunication Systems*. [S.l.], 2010. p. 64–66.

91 SHARIF, S. O.; KUNCHEVA, L.; MANSOOR, S. Classifying encryption algorithms using pattern recognition techniques. In: IEEE. *2010 IEEE International Conference on Information Theory and Information Security*. [S.l.], 2010. p. 1168–1172.

92 MANJULA, R.; ANITHA, R. Identification of encryption algorithm using decision tree. In: SPRINGER. *International Conference on Computer Science and Information Technology*. [S.l.], 2011. p. 237–246.

93 SOUZA, W. A. R. de; CARVALHO, L. A. V. de; XEXEO, J. A. M. Identification of n block ciphers. *IEEE Latin America Transactions*, IEEE, v. 9, n. 2, p. 184–191, 2011.

94 LAZARIN, N. M. Método não supervisionado de reconhecimento de padrões criptográficos. 2012.

95 CHOU, J.-W.; LIN, S.-D.; CHENG, C.-M. On the effectiveness of using state-of-the-art machine learning techniques to launch cryptographic distinguishing attacks. In: *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*. [S.l.: s.n.], 2012. p. 105–110.

96 MISHRA, S.; BHATTACHARJYA, A. Pattern analysis of cipher text: a combined approach. In: IEEE. *2013 International Conference on Recent Trends in Information Technology (ICRTIT)*. [S.l.], 2013. p. 393–398.

97 GRAY, R. M. *Entropy and information theory*. [S.l.]: Springer Science & Business Media, 2011.

98 SOUZA, W. A. D.; TOMLINSON, A. A distinguishing attack with a neural network. In: IEEE. *2013 IEEE 13th International Conference on Data Mining Workshops*. [S.l.], 2013. p. 154–161.

99 LOMTE, V. M.; SHINDE, A. D. Review of a new distinguishing attack using block cipher with a neural network. *International Journal of Science and Research*, v. 3, n. 8, p. 2012–2015, 2014.

100 MELLO, F. L.; XEXEO, J. A. M. Cryptographic algorithm identification using machine learning and massive processing. *IEEE Latin America Transactions*, IEEE, v. 14, n. 11, p. 4585–4590, 2016.

101 BARBOSA, F.; VIDAL, A.; MELLO, F. Machine learning for cryptographic algorithm identification. *Journal of Information Security and Cryptography (Enigma)*, v. 3, n. 1, p. 3–8, 2016.

102 BARBOSA, F. M.; VIDAL, A. R. S. F.; ALMEIDA, H. L. S.; MELLO, F. L. de. Machine learning applied to the recognition of cryptographic algorithms used for multimedia encryption. *IEEE Latin America Transactions*, IEEE, v. 15, n. 7, p. 1301–1305, 2017.

103 MELLO, F. L. de; XEXEO, J. A. Identifying encryption algorithms in ecb and cbc modes using computational intelligence. *J. Univers. Comput. Sci.*, v. 24, n. 1, p. 25–42, 2018.

104 TAN, C.; DENG, X.; ZHANG, L. Identification of block ciphers under cbc mode. *Procedia Computer Science*, Elsevier, v. 131, p. 65–71, 2018.

105 PRADEEPTHI, K.; TIWARI, V.; SAXENA, A. Machine learning approach for analysing encrypted data. In: IEEE. *2018 Tenth International Conference on Advanced Computing (ICoAC)*. [S.l.], 2018. p. 70–73.

106 PANAHI, P.; BAYILMIŞ, C.; ÇAVUŞOĞLU, U.; KAÇAR, S. Performance evaluation of l-block algorithm for iot applications. *algorithms*, v. 14, p. 15, 2018.

107 JIN, X.; HAN, J. Expectation maximization clustering. In: *Encyclopedia of Machine Learning and Data Mining*. [S.l.]: Springer, 2016. p. 1–2.

108 FAN, S.; ZHAO, Y. Analysis of cryptosystem recognition scheme based on euclidean distance feature extraction in three machine learning classifiers. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2019. v. 1314, n. 1, p. 012184.

109 HU, X.; ZHAO, Y. Block ciphers classification based on random forest. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2019. v. 1168, n. 3, p. 032015.

110 ZHANG, W.; ZHAO, Y.; FAN, S. Cryptosystem identification scheme based on ascii code statistics. *Security and Communication Networks*, Hindawi Limited, v. 2020, p. 1–10, 2020.

111 GRIFFIN, G.; HOLUB, A.; PERONA, P. Caltech-256 object category dataset. California Institute of Technology, 2007.

112 PANG, E. Understanding ascii and unicode character encoding. *ACM Inroads*, ACM, v. 5, n. 1, p. 74–78, 2014.

113 KAVITHA, T.; RAJITHA, O.; THEJASWI, K.; MUPPALANENI, N. B. Classification of encryption algorithms based on ciphertext using pattern recognition techniques. In: SPRINGER. *Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2018)*. [S.l.], 2020. p. 540–545.

114 YU, X.; SHI, K. Block ciphers identification scheme based on randomness test. In: SPIE. *6th International Workshop on Advanced Algorithms and Control Engineering (IWAACE 2022)*. [S.l.], 2022. v. 12350, p. 375–380.

115 YUAN, K.; HUANG, Y.; DU, Z.; LI, J.; JIA, C. A multi-layer composite identification scheme of cryptographic algorithm based on hybrid random forest and logistic regression model. *Complex & Intelligent Systems*, Springer, p. 1–17, 2023.

116 BRASIL, S. B. do. Bíblia sagrada: nova tradução na linguagem de hoje. *Barueri: SBB*, 2000.

117 STALLINGS, W.; BRESSAN, G.; BARBOSA, A. *Criptografia e segurança de redes* . [S.l.]: Pearson Educación, 2008.

118 MARTIN, G. R. R. *Fogo & Sangue*. São Paulo, Brasil: Editora Suma, 2018. ISBN 978-8556510869.

119 DAMAS, L. *Linguagem C* . [S.l.]: Grupo Gen-LTC, 2016.

120 SOARES, L. E.; PIMENTEL, R.; BATISTA, A.; FERRAZ, C. *Elite da tropa*. [S.l.]: Nova Fronteira, 2006.

121 SOARES, L. E.; PIMENTEL, R.; BATISTA, A.; FERRAZ, C. *Elite da tropa 2*. [S.l.]: Nova Fronteira, 2011.

122 STAPOR, K.; KSIENIEWICZ, P.; GARCÍA, S.; WOŹNIAK, M. How to design the fair experimental classifier evaluation. *Applied Soft Computing*, Elsevier, v. 104, p. 107219, 2021.

123 BORRA, S.; CIACCIO, A. D. Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational statistics & data analysis*, Elsevier, v. 54, n. 12, p. 2976–2989, 2010.

124 KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. v. 14, 03 2001.

125 KIM, J.-H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational statistics & data analysis*, Elsevier, v. 53, n. 11, p. 3735–3745, 2009.

126 ROCHA, B. D. S.; XEXEO, J. A. M.; TORRES, R. H. Artificial intelligence applied to the identification of block ciphers under cbc mode. *International Journal of Computer Applications*, Foundation of Computer Science (FCS), NY, USA, New York, USA, v. 185, n. 34, p. 1–8, Sep 2023. ISSN 0975-8887. Disponível em: <http://www.ijcaonline.org/archives/volume185/number34/32907-2023923114>.

127 HERRANZ, J.; HOFHEINZ, D.; KILTZ, E. The kurosawa-desmedt key encapsulation is not chosen-ciphertext secure. *Cryptology ePrint Archive*, 2006.

128 SOUZA, W. A. de; XEXÉO, J. A.; OLIVEIRA, C. Método de agrupamento de criptogramas em função das chaves de cifrar. *IV WAAMD*, p. 19, 2008.

129 NAEHRIG, M.; ALKIM, E.; BOS, J.; DUCAS, L.; EASTERBROOK, K.; LAMACCHIA, B.; LONGA, P.; MIRONOV, I.; NIKOLAENKO, V.; PEIKERT, C. et al. Frodokem: Learning with errors key encapsulation–algorithm specifications and supporting documentation. *NIST Technical Report*, 2019.

130 ROBERTO, A.; JOPPE, B.; LÉO, D. et al. Crystals-kyber: algorithm specification and supporting documentation. *Submission to the NIST Post-Quantum Cryptography Standardization Project*, 2019.

131 VERCAUTEREN, I. F. *SABER: Mod-LWR Based KEM (Round 2 Submission)*. 2020.

132 CHEN, C.; DANBA, O.; HOFFSTEIN, J.; HULSING, A.; RIJNEVELD, J.; SCHANCK, J. M.; SCHWABE, P.; WHYTE, W.; ZHANG, Z. Ntru: algorithm specifications and supporting documentation (2019). *URL: https://csrc. nist. gov/projects/post-quantum-cryptography/round-2-submissions. Citations in this document*, v. 1, 2019.

133 ROCHA, B. S.; XEXEO, J. A. M.; TORRES, R. H. Post-quantum cryptographic algorithm identification using machine learning. *Journal of Information Security and Cryptography (Enigma)*, v. 9, n. 1, p. 1–8, 2022.