Applications of Neural Network-Based AI in Cryptography

Abderrahmane Nitaj[1] and Tajjeeddine Rachidi[2]

[1] Normandie Univ, UNICAEN, CNRS, LMNO, 14000 Caen, France
abderrahmane.nitaj@unicaen.fr
[2] School of Science and Engineering, Alakhawayn University in Ifrane, Morocco
T.Rachidi@aui.ma

Abstract. Artificial intelligence (AI) is a modern technology that allows plenty of advantages in daily life, such as predicting weather, finding directions, classifying images and videos, even automatically generating code, text, and videos. Other essential technologies such as blockchain and cybersecurity also benefit from AI. As a core component used in blockchain and cybersecurity, cryptography can benefit from AI in order to enhance the confidentiality and integrity of cyberspace. In this paper, we review the algorithms underlying four prominent cryptographic cryptosystems, namely the Advanced Encryption Standard, the Rivest–Shamir–Adleman, Learning With Errors, and the Ascon family of cryptographic algorithms for authenticated encryption. Where possible, we pinpoint areas where AI can be used to help improve their security.

Keywords: Artificial Intelligence, Machine Learning, ANNs, Cryptography, AES, RSA, LWE, ASCON.

1 Introduction

In 1991, Rivest [3] presented a talk about relationships between machine learning and cryptography. Surprisingly, while artificial intelligence (AI) is being extensively developed in a number of applications, a very limited number of research studies have been done in the area of using AI in cryptography.

AI is the set of tools, methodologies, and implementations deployed to enable a digital computer or a robot to perform tasks that are usually associated with human intelligence [4,5]. In the last few decades, AI has exponentially developed in many sectors, such as big data, Internet of Things, robotics, banking, finance, healthcare, e-commerce, meteorology, education, facial recognition, information systems, autonomous driving, data security, etc. Machine learning (ML) is a subset of AI that enables smart machines and computers to learn without human intervention and gives them the ability to imitate human behavior. The goal of ML is to design algorithms that extract information from data in order to build models capable of deriving predictions and patterns. ML covers a range of methodologies and applications, such as facial recognition, natural language, online chatbots, medical imaging, diagnostics, self-driving of vehicles, etc.

On the other hand, cybersecurity is the set of various methods and tools deployed to protect electronic devices, such as information systems, servers, computers, networks, and data centers, from all kind of threats, vulnerabilities, and attacks. Often, an attack on an electronic device has severe impacts on the regular operations, or even worse, can completely destroy the stored data. Therefore, the goal of cybersecurity is to detect any attack, handle it, and recover the system after the accident. Cybersecurity includes the use of cryptography and cryptographic protocols for protecting data in transit and in storage.

The aim of this paper is to present an overview of the applications of AI and ML in cryptography with a focus on four prominent cryptosystems, namely, AES, RSA, LWE, and Ascon. For this, we start by providing an overview of AI and ML techniques before embarking on presenting the above-referenced cryptosystems and explaining how AI and ML can be applied to improve their security for the benefit of cybersecurity.

Cryptography is concerned with protecting information and communications transferred over public communication channels in the presence of adversaries. It allows only the recipient of a message to view its contents and is used in all domains where the security is a concern. To transmit a message or electronic data, two families of cryptography can be used: *symmetric* and *asymmetric cryptography*. In symmetric cryptography, also called secret key cryptography, the same key is used for both encryption and decryption. Typically, a message is encrypted using a secret key, and both the encrypted message and the secret key are sent to the recipient for decryption. In asymmetric cryptography, invented by Diffie and Hellman [6] in 1976, and mostly known as public key cryptography, two keys are involved: one is public, and one is private. In general, the two keys are related by a mathematical process with the idea that it is computationally infeasible to determine one key given the other one. To encrypt and send a message, the sender uses the public key of the recipient. To decrypt, the recipient uses his or her private key.

Cryptanalysis is the study of cryptographic schemes for vulnerabilities. Specifically, there are mainly two methods of deploying cryptanalysis: mathematical and side-channel. Mathematical cryptanalysis, or algebraic cryptanalysis, consists of breaking cryptographic schemes by scrutinizing their mathematical properties, while side-channel cryptanalysis consists of studying and manipulating the implementations in order to collect information on the keys or on the plaintext itself.

In symmetric cryptography, the security of any scheme is based on the robustness of its S-box, a nonlinear operator that is often related to a vectorial Boolean function with very good cryptographic properties, such as resistance to differential cryptanalysis; linear cryptanalysis; boomerang cryptanalysis; and a variety of other cryptographic criteria [7]. Artificial intelligence can be used to design S-boxes from vectorial Boolean functions and to study their cryptographic properties in order to select the most efficient and the most secure schemes.

In asymmetric cryptography, security is often based on a hard mathematical problem such as the integer factorization problem, the discrete logarithm problem, the Shortest Vector Problem (SVP), and the Closest Vector Problem (CVP) in a lattice.

Currently, the most widely used asymmetric cryptosystem is RSA, invented in 1978 by Rivest, Shamir, and Adleman [8]. RSA is used for encryption, signatures, and key distribution, and is a powerful tool to provide privacy and to ensure authenticity of emails, digital data, and payment systems. The mathematics behind RSA are based on the ring $\mathbb{Z}/N\mathbb{Z}$, where $N = pq$ is the product of two large prime numbers. In RSA, the public key is an integer e satisfying $\gcd(e, (p-1)(q-1)) = 1$, and the private key is the integer d satisfying $ed \equiv 1 \pmod{(p-1)(q-1)}$. To encrypt a message $1 < m < N$, one computes $c \equiv m^e \pmod{N}$, and to decrypt it, one computes $m \equiv c^d \pmod{N}$. Since its invention, RSA has been intensively analyzed for vulnerabilities [9,10,11,12].

A promising family of asymmetric cryptography has appeared with the Learning With Error (LWE) problem and its variants. LWE was proposed by Regev [13] in 2005. Several homomorphic encryption libraries, public key encryptions, and digital signature systems are based on LWE or on one of its variants [14]. In 2016, NIST initiated a process to select and standardize post-quantum cryptography standardization [15], and in 2022, it selected CRYSTALS–Kyber [16] for public-key encryption and CRYSTALS–Dilithium [17], Falcon [18], and SPHINCS+ [19] for digital signatures. Among the four selected algorithms for standardization, three are based on hard problems in lattices, namely CRYSTALS–Kyber, CRYSTALS–Dilithium, and Falcon. There are several variants of LWE, such as Polynomial-LWE [20], Ring-LWE [21], Module-LWE [22], and Continuous LWE [23]. The goal in LWE is to find a secret vector $s \in \mathbb{Z}_q^n$ given $m \geq n$ samples of the form $(a_i, \langle a_i, s\rangle + e_i)$, where $a_i \in \mathbb{Z}_q^n$ is uniformly generated, $e_i \in \mathbb{Z}_q^m$ is a small vector chosen according to a probability density, and $\langle a_i, s\rangle$ is the inner product of a_i and s. The security of LWE is based on several hard problems in lattices, specifically the Gap Shortest Vector Problem (GapSVP) and the Shortest Independent Vectors Problem (SIVP).

The explosion of the Internet of Things (IoT), characterized by low-energy and low-computation power devices, has prompted the need for efficient and strong lightweight ciphers for protecting the privacy and authenticity of data transmitted by these devices. The U.S. government's National Institute of Standards and Technology (NIST) recently selected the Ascon family of lightweight symmetric ciphers for authenticated encryption [24,25] to be used by IoT devices. The Ascon family ensures 128-bit security and uses a 320-bit permutation internally.

Common attacks on both symmetric and asymmetric cryptography are side-channel attacks, introduced by Kocher [26] in 1996. Side-channel attacks are used to retrieve private keys from electronic devices. There are various types of possible side-channel attacks depending on the cryptosystem and the device. This includes timing execution [26], power consumption [27], electromagnetic radiation [28], fault injection [29], and acoustic attack [30].

In symmetric and asymmetric cryptography, a plaintext M is encrypted by a nonlinear trapdoor function F together with a secret or a private key K so that $C = F(M, K)$. An algebraic attack consists of finding the plaintext M or the key K using publicly accessible Cs and, eventually, finding their known corresponding Ms. Moreover, for symmetric ciphers, an algebraic attack can be used to approximate the hole or a partial (reduced-round) encryption process by a linear function, which makes the cipher vulnerable.

The rest of this paper is organized as follows. In Section 2, we review the main facts of artificial intelligence (AI) and machine learning (ML). In Section 3, we discuss the differences between AI and ML. In Section 4, we provide a list of possible applications of AI in cryptography. In Sections 5–8, we review the four prominent cryptosystems, namely AES, RSA, LWE, and Ascon, and present possible applications of AI to test and enhance their security. We conclude the paper in Section 9.

2 Artificial Intelligence and Machine Learning

Artificial intelligence (AI) is a subarea of computer science that concerns itself with building rational agents, i.e., agents that sense the world (i.e., read a percept), map the percept to some internal representation, and identify the best action to take among a set of possible actions given the percept. The selected action is the one that minimizes the agentâĂŹs objective function, then enacts the action and updates the internal representation of the world as a consequence of the action. There exist various types of agents: search agents, adversarial search agents, planning agents, logical agents, probabilistic agents, and learning agents. The latter are data-driven and use machine learning algorithms to predict the action to take as a function of the input percept from a collection of tuples (percept, action) [31].

Machine learning (ML) is a subarea of AI that concerns itself with learning agents and algorithms. There exist three (3) major classes of learning agents/algorithms:

1. Supervised learning algorithms. These use tuples (input vector; output vector, also called Label) to learn/approximate the output vector for an unseen given input vector.
2. Unsupervised learning algorithms. These do not make use of the label and use the input vector only to learn/infer/approximate the output vector for an unseen given input vector.
3. Reinforcement learning algorithms. This class progressively learns/infers/approximates the output vector for an unseen given input vector from the positive or negative feedback returned from the external world.

Recently, two more subclasses have come to light. These are:

1. Self-supervised learning algorithms. These algorithms mask parts of the input and try to learn it. In essence, these algorithms transform an unsupervised problem (i.e., a problem for which no labels exist) into a supervised problem by auto-generating the labels.

2. Imitation learning algorithms. These are very recent. In essence, imitation learning algorithms reproduce others' actions from observing the actions performed by other agents/machine learning algorithms [32].

A comprehensive review and taxonomy of artificial intelligence and machine learning techniques together with their disadvantages and challenges can be found in [33].

Artificial neural networks (ANNs) in particular have proven to be more powerful than others in many applications, including computer vision (CV), natural language processing (NLP), and autonomous driving (AD).

Artificial Neural Networks (ANNs) as a Non-Linear Approximation Function

Artificial neural network models are mainly characterized by their architecture, the number of densely connected hidden layers, the number of cells/perceptrons (see Figure 1) per layer, activation functions used in the neurons for firing, and the cost function used to train the network (or, similarly, to find the weights of the interconnections between layers) using gradient descent and back propagation for computing the gradient of the cost function in the weight space [34] (see Figure 2). ANNs can be divided into three major categories: those that perform input classification, sequence learning, or function approximation.

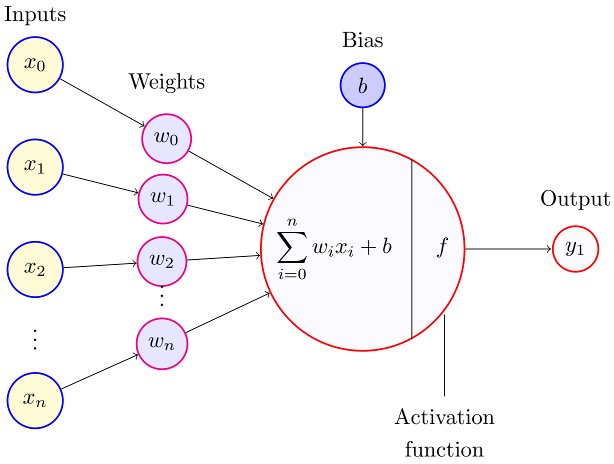

Fig. 1. basic unit of artificial neural networks: the perceptron.

Thanks to nonlinear activation functions such as the Sigmoid, Tanh or RELU functions, ANNs are great at approximating arbitrary nonlinear functions (often as piece-wise linear approximations).

In fact, an ANN with at least one hidden layer is a universal approximator, i.e., it can represent any function [35]. However, one hidden layer might need an exponential number of neurons, so often an architecture with many fully connected hidden layers (deep neural network) is preferred, as it is more compact. Arguably, the performance of the network increases with more hidden units and more hidden layers (see Figure 3).

Although there is no universal method to approximate any arbitrary given function, there are development procedures that consistently lead to successful approximations of nonlinear functions within a specific field. There is wealth of literature for such applications in all fields, including bioengineering, mechanics, agriculture, digital design, and, last but not least, intrusion detection and cybersecurity.

As arbitrary function approximators, ANNs lend themselves naturally to cryptanalysis techniques, such as known and chosen plaintext attacks, linear, nonlinear, and differential attacks. Furthermore, the feed-forward

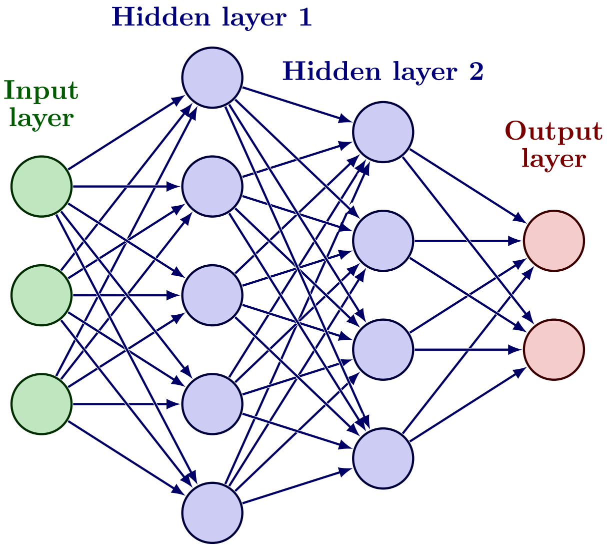

Fig. 2. A multi-layer perceptron forming a 4-layer neural net with 3 input units, 5 units in the first hidden layer, 4 units in the second hidden layer, and 2 output units.

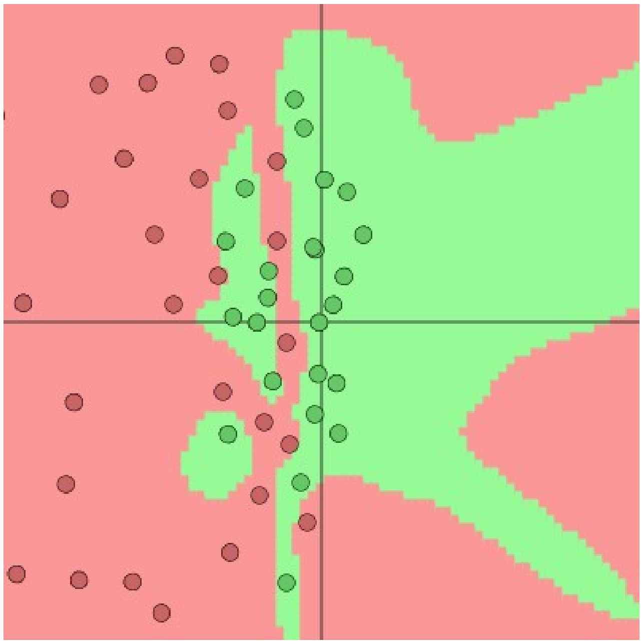

Fig. 3. Learning the nonlinear line that separates the green dots from the red dots using 2 hidden layers with 20 neurons each. Figure generated using [36].

of the data process through multiple layers of the neural networks has functional resemblance to multiple-round operations in a symmetric cipher, i.e., linear permutations, followed by nonlinear transformations. This also invites the attempt to leverage ANNs for the design of ciphers.

3 ANN Types and Their Domains of Application

Due to numerous advantages over other ML techniques, such as the ability to learn hierarchical features, the ability to handle multiple output, and the ability to deal with nonlinear data, which is clearly highlighted by the rapid development of foundation models (FMs) [1], ANN-based learning agents have overshadowed other types of intelligent agents, including the ones that use other machine learning techniques. They have become a synonym for AI. This said, and despite the aforementioned advantages, it is not guaranteed that, because deep ANNs demonstrate excellent performance for domain problems dealing with language, images, and videos, they will necessarily outperform other ML techniques in cryptography [37,38]. Unless otherwise specified, we will use the term AI to designate agents that use ANNs.

There are many types of neural networks, such as auto-encoders, convolutional neural networks (CNNs), long short-term memory networks (LSTMs), recurrent neural networks (RNNs), generative adversarial networks (GANs), and transformers. They stand apart from one other through the type of processing attached to individual layers and the architecture used to connect the hidden layers, among other things, as well as whether a cell uses its own output from previous excitation or not.

3.1 Convolutional Neural Networks (CNNs)

In addition to hidden layers, a CNN contains multiple convolution layers, which are responsible for the extraction of important features, such as images, from spatial data. The earlier layers are responsible for low-level details, and the later layers are responsible for more high-level features. As such, CNNs are well-suited for applications such as facial recognition, medical analysis, and image classification.

3.2 Recurrent Neural Networks (RNNs)

RNNs are used to predict the next item in sequential data, which can be videos or text. In RNNs, a neuron in a layer also receives a time-delayed input from its own previous instance prediction. This instance prediction is stored in the RNN cell, which is a second input for every prediction. RNNs are typically used in tasks such as text or speech generation, text translation, and sentiment analysis.

3.3 Autoencoders

An autoencoder is a type of artificial neural network used to learn efficient coding of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding function that transforms the input data into a low-dimension latent space representation, and a decoding function that recreates the input data from the latent space representation. Autoencoders are used in dimensionality reduction, image compression, image denoising, feature extraction, image generation using generative adversarial networks (GANs), sequence-to-sequence predictions, and recommendation systems.

3.4 Long Short-Term Memory Networks (LSTMs)

LSTMs use gates to control which output should be used or forgotten, including: input gate, output gate, and forget gate. LSTMs are best applied in speech recognition and text prediction.

3.5 Generative Adversarial Networks (GANs)

GANs learn to create new data instances that resemble the training data. For example, GANs can create images that look like photographs of human faces, even though the faces do not belong to any real person [39].

3.6 Transformers

The transformer model architecture drops recurrence and convolutions and uses an attention mechanism to connect an encoder network with a decoder network. Applications of this model include machine translation [40].

All of these variants of ANNs are mainly characterized by their architecture, the number of parameters/weights that make up the model and that have to be learned, and the training corpora used (Common Crawl, The Pile, MassiveText, Wikipedia, GitHub, books, articles, logs, etc.). As this paper is being written, the Megatron-Turing Natural Language Generation (MT-NLG), a transformer-based language generation model, uses 530 billion parameters, more than 7 times the average number of neurons in the adult human brain. These are also called foundation/base models since they can be adapted/fine-tuned for different tasks/contexts. Head-to-head comparison of existing (commercial and open source) large models (LMs) can be found in [41]. It is worth noting that the size of a model depends largely on the nature of the problem at hand. In many engineering domains, the models used are not as big as foundation models.

4 Possible Applications of AI in Cryptography

The combination of cryptography with artificial intelligence will be beneficial for the security of several applications. One of the goals of applying AI is to identify potential vulnerabilities of a cryptographic system.

4.1 Areas of Application

1. *In cybersecurity:* Cybersecurity can easily benefit from the applications of AI. By applying AI, it is possible to write and process software to detect and to defend a system against cyberattacks. The advantages of applying AI instead of traditional security systems is that AI provides fast solutions and better security.
2. *In blockchain:* Blockchain is a new technology with various industrial and economic applications. It plays a prominent role in many sectors, such as banking, cryptocurrencies, and data management. It achieves a complete independence from any central authority and guarantees secure communications thanks to advanced cryptographic techniques. AI can be used to analyze the security and the efficiency of blockchain applications in order to improve their practicability, security, and profitability.
3. *In symmetric cryptography:* AI can be deployed to analyze the security of a symmetric system defined by an S-box or a vectorial Boolean function by testing all possible cryptographic criteria, including bijectivity, nonlinearity, linear analysis, differential analysis, balancedness, correlation immunity, algebraic degree, side-channel analysis, strict avalanche criterion (SAC), bit independence criterion (BIC), and the NIST Statistical Test Suite [42], which is used to guarantee the quality of random number generators for cryptographic applications. Especially, the security of AES and Ascon can be much improved if tested with the help of AI.
4. *In asymmetric cryptography based on RSA:* AI can be used to generate safe primes for the RSA modulus, and to generate safe public and private keys by running the known attacks such as factorization, small private key attacks, partial key exposure attacks, and side-channel attacks.

5. *In asymmetric cryptography based on LWE:* Attacks on LWE and its variants are very limited because their security is based on the hardness of hard problems in lattices. Nevertheless, AI can be used to test the hardness of lattice problems with different parameters in order to guarantee the safety and the efficiency of the cryptosystem.

This said, AI itself can benefit from modern cryptographic techniques, such as homomorphic encryption, to resolve the privacy issue related to data used in learning without disclosing it [43].

Without much surprise, vanilla ANNs were applied very early in all areas of cryptology, including side-channel attacks [44], random number generation [45], recovery of plaintext [46], generation of ciphertext without the key [47], cryptanalysis of symmetric ciphers [48,49,50], and hash and message authentication functions [51]. Attempts have also be made to implement ciphers as ANNs [52].

The application of advanced ANNs such as deep, convolutional, and generative adversarial neural networks is also gaining in momentum. In this regard, Ref. [53] deployed a deep network for side-channel attacks on masked and unprotected AES implementations. Ref. [54] deployed a linear attack on round-reduced DES using deep learning with plain-cipher pairs. Ref. [55] posed the cryptanalysis of a cipher as a language translation problem to be solved using a GAN, which was adapted to handle discrete data. The GAN is trained to learn the mapping between plain and cipher text distributions without supervision. Ref. [56] used deep convolutional neural networks to exploit differential properties of round-reduced Speck cipher to perform a differential distinguishing attack that did not involve key search. Ref. [57] used a deep neural network to perform the known-plaintext attack on AES and its modes of operation to restore different bit lengths with probabilities. Ref. [58] used deep learning in side-channel attacks against a secure implementation of the RSA algorithm. Surprisingly, the applications of advanced ANNs to asymmetric encryption has yet to begin. Therefore, in this article, we attempt to pinpoint stages in prominent encryption algorithms, namely AES, RSA, and LWE, where the applications of advanced ANNs can help increase their security.

4.2 Datasets

Datasets are structured collections of data used to train a model for the nonlinear trapdoor function $C = F(K, M)$. They consist of pairs of collected (M, C) that are generated synthetically at the design phases of F. Typically, all combinations of Ms and their differences are generated and fed to F to obtain Cs, leading to a balanced dataset of pairs (M, C).

5 The Advanced Encryption Standard (AES)

The Advanced Encryption Standard [59], also known as the Rijndael algorithm, is a symmetric block cipher that was designed by Daemen and Rijmen [60] in 1999. It was adopted by the U.S. National Institute of Standards and Technology (NIST) in 2001 to supersede the Data Encryption Standard (DES) [61]. AES allows key lengths of size 128, 192, or 256 bits, with a block length of 128 bits. In AES, the encryption performs 10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key.

The encryption and the decryption in the AES algorithm start with two parameters: a block B of length 128 bits, and a key K of length 128, 192, or 256 bits (see Table 1). In all steps of the encryption and decryption in AES, the blocks $B = \{B_0, \ldots, B_{15}\}$ are represented by 4×4 square matrices of bytes called state arrays.

Table 1. Representation of the block and the subkey with bytes.

$$\begin{array}{|c|c|c|c|}\hline B_0 & B_1 & B_2 & B_3 \\ \hline B_4 & B_5 & B_6 & B_7 \\ \hline B_8 & B_9 & B_{10} & B_{11} \\ \hline B_{12} & B_{13} & B_{14} & B_{15} \\ \hline \end{array}\,, \quad \begin{array}{|c|c|c|c|}\hline K_0 & K_1 & K_2 & K_3 \\ \hline K_4 & K_5 & K_6 & K_7 \\ \hline K_8 & K_9 & K_{10} & K_{11} \\ \hline K_{12} & K_{13} & K_{14} & K_{15} \\ \hline \end{array}\,.$$

Table 2. AddRoundKey operation Xors state with key. .

$$\begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline K_0 & K_1 & K_2 & K_3 \\ \hline K_4 & K_5 & K_6 & K_7 \\ \hline K_8 & K_9 & K_{10} & K_{11} \\ \hline K_{12} & K_{13} & K_{14} & K_{15} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline S'_0 & S'_1 & S'_2 & S'_3 \\ \hline S'_4 & S'_5 & S'_6 & S'_7 \\ \hline S'_8 & S'_9 & S'_{10} & S'_{11} \\ \hline S'_{12} & S'_{13} & S'_{14} & S'_{15} \\ \hline \end{array}\,.$$

5.1 The Encryption Process of AES

At the beginning of the encryption, each key K is expanded into $n+1$ subkeys by an algorithm called *key expansion*, where $n \in \{10, 12, 14\}$ is the number of rounds. The encryption phase starts with the initial round by XORing the plaintext with the first subkey. Then, the rounds are composed of four algorithms, namely AddRoundKey, SubBytes, ShiftRows, and MixColumns, so that a round R_i with $0 \le i \le n$ is in the form:

$$R_i = \begin{cases} \texttt{AddRoundKey}(\text{plaintext, subkey}_0) & \text{if } i = 0, \\ \texttt{AddRoundKey} \circ \texttt{MixColumns} \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & \text{if } 0 < i < n, \\ \texttt{AddRoundKey} \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & \text{if } i = n. \end{cases}$$

The four algorithms can be summarized as follows:

- *AddRoundKey:* The subkey for the round is bitwise XORed with the state array computed in the previous step. In the first round, the state array is the input block, and in the last round, the resulting state array is the ciphertext (see Table 2).
- *SubBytes:* The SubBytes transformation is a byte substitution that operates on each byte of the state using a substitution table called S-box (see Table 3). Algebraically, each byte x is transformed into a list of 8 bits,

  $$\overline{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8),$$

  and is transformed via the rule T,

  $$T(\overline{x}) = \begin{cases} c & \text{if } x = 0 \\ M\left(\overline{x}^{-1}\right) + c & \text{if } x \neq 0, \end{cases}$$

where

$$c = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \qquad M = \begin{bmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \end{bmatrix},$$

and $\overline{x}^{-1}$ is the inverse of $\overline{x}$ in the finite field $\mathbb{F}_{2^8}$ modulo the polynomial $x^8 + x^4 + x^3 + x + 1$.

Table 3. SubBytes operation yielding a new state vector.

$$\text{Transformation } T \text{ on } \begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline S'_0 & S'_1 & S'_2 & S'_3 \\ \hline S'_4 & S'_5 & S'_6 & S'_7 \\ \hline S'_8 & S'_9 & S'_{10} & S'_{11} \\ \hline S'_{12} & S'_{13} & S'_{14} & S'_{15} \\ \hline \end{array}.$$

- *ShiftRows:* In this transformation, the bytes of the first row in the state array remain unchanged, and the bytes of rows 2, 3, and 4 are cyclically shifted left by 1, 2, and 3 cases, respectively (see Table 4).

Table 4. ShiftRows operation yielding a new state.

$$\begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_5 & S_6 & S_7 & S_4 \\ \hline S_{10} & S_{11} & S_8 & S_9 \\ \hline S_{15} & S_{12} & S_{13} & S_{14} \\ \hline \end{array}.$$

- *MixColumns:* In this transformation, each column is multiplied by a fixed matrix,as in Table 5.

Table 5. MixColumns operation yielding a new state.

$$\begin{array}{|c|c|c|c|}\hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline S'_0 & S'_1 & S'_2 & S'_3 \\ \hline S'_4 & S'_5 & S'_6 & S'_7 \\ \hline S'_8 & S'_9 & S'_{10} & S'_{11} \\ \hline S'_{12} & S'_{13} & S'_{14} & S'_{15} \\ \hline \end{array}.$$

In MixColumns, the operations are performed in $\mathbb{F}_{2^8}$ modulo the polynomial $x^8 + x^4 + x^3 + x + 1$.

5.2 The Decryption Process in AES

The decryption process in AES is performed by applying the inverse of the algorithms used in the encryption process. If n is the number of rounds in the encryption process, then there are $m = n$ rounds in the decryption process. The decryption starts by XORing the ciphertext with the last subkey of the key expansion. For $0 \leq i \leq m$, the the inverse round $InvR_i$ is composed of four algorithms as follows:

$$InvR_i =$$
$$\begin{cases} \texttt{InvAddRoundKey}(\text{ciphertext, subkey}_m) & \text{if } i = 0, \\ \texttt{InvMixColumns} \circ \texttt{InvAddRoundKey} \circ \texttt{InvSubBytes} & \\ \quad \circ \texttt{InvShiftRows} & \text{if } 0 < i < m, \\ \texttt{InvAddRoundKey} \circ \texttt{InvSubBytes} \circ \texttt{InvShiftRows} & \text{if } i = m. \end{cases}$$

The algorithms can be summarized as follows.

- *InvAddRoundKey:* As in the AddRoundKey algorithm, the subkey for the round is bitwise XORed, with the state array computed in the previous step. In the first round, the state array is the ciphertext block, and in the last round, the resultant state array is the plaintext.
- *InvSubBytes:* In this operation, the inverse S-box replaces each byte of the state with another byte by a substitution method. Specifically, each byte y is transformed into a list of 8 bits,

$$\overline{y} = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8),$$

  and is transformed via the rule T',

$$T'(\overline{y}) = \begin{cases} 0 & \text{if } y = c \\ \frac{1}{M^{-1}(y+c)} & \text{if } y \neq c, \end{cases}$$

  where

$$c = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \qquad M^{-1} = \begin{bmatrix} 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \\ 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \end{bmatrix}.$$

  The inverses are computed in the finite field $\mathbb{F}_{2^8}$ modulo the polynomial $x^8 + x^4 + x^3 + x + 1$.
- *InvShiftRows:* In this transformation, the bytes of the first row in the state array remain unchanged, and the bytes of rows 2, 3, and 4 are cyclically shifted right by 1, 2, and 3 cases, respectively (see Table 6).

Table 6. InvShiftRows transforms the state on the left to the state on the right.

$$.\quad \begin{array}{|c|c|c|c|} \hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} \quad \longrightarrow \quad \begin{array}{|c|c|c|c|} \hline S_0 & S_1 & S_2 & S_3 \\ \hline S_7 & S_4 & S_5 & S_6 \\ \hline S_{10} & S_{11} & S_8 & S_9 \\ \hline S_{13} & S_{14} & S_{15} & S_{12} \\ \hline \end{array} \quad .$$

- *InvMixColumns:* In this transformation, each column is multiplied by a fixed matrix, as in Table 7.

Table 7. InvMixColumns multiplies the state with the given matrix.

$$.\begin{array}{|c|c|c|c|}\hline 0e & 0b & 0d & 09 \\ \hline 09 & 0e & 0b & 0d \\ \hline 0d & 09 & 0e & 0b \\ \hline 0b & 0d & 09 & 0e \\ \hline \end{array} \times \begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline S'_0 & S'_1 & S'_2 & S'_3 \\ \hline S'_4 & S'_5 & S'_6 & S'_7 \\ \hline S'_8 & S'_9 & S'_{10} & S'_{11} \\ \hline S'_{12} & S'_{13} & S'_{14} & S'_{15} \\ \hline \end{array}.$$

In InvMixColumns, the operations are performed in $\mathbb{F}_{2^8}$ modulo the polynomial $x^8 + x^4 + x^3 + x + 1$.

5.3 Main Attacks on AES

The goal of the attacks on an asymmetric cryptosystem is to find good properties inside the cipher that allow for retrieval of partial or total information on the secret key. In addition to the exhaustive attack, the two prominent attacks are the linear cryptanalysis and the differential cryptanalysis.

- *Exhaustive search attack.* Brute force attacks, or exhaustive attacks, consist of trying all possible keys to a ciphertext and checking whether the plaintext is recognizable. It is easy to prevent such attacks by using large keys. In AES, the key lengths are 128, 192, and 256 bits. This makes the total key combination of each key length 2^{128}, 2^{192}, and 2^{256}, respectively, which is infeasible even for the fastest supercomputers today. On the other hand, with a computer with quantum technology, due to Grover's algorithm [62], it is possible to perform an exhaustive search in the square root of the classical time, and the key lengths should be 2^{256}.
- *Linear attack.* In 1993, Matsui [63] invented one of the most practical attacks on DES, known as linear cryptanalysis. It can be applicable to AES by approximating the nonlinear parts in the rounds by linear expressions. This makes the round a linear function where the input or the output is easy to compute.
  In the situation where the S-box of the system is constructed following a vectorial boolean function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, the linear cryptanalysis is constructed on the value of its nonlinearity, which is defined by:

  $$\texttt{NL}_F = 2^{n-1} - \frac{1}{2} \max_{a \neq 0, b \in \mathbb{F}_{2^n}} \left| \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot F(x) \oplus a \cdot x} \right|,$$

  where $a \cdot x$ is the inner product in $\mathbb{F}_2$, defined as $a \cdot x = \oplus_{i=0}^{n} a_i x_i$. The nonlinearity of the function F represents the minimum Hamming distance between F and all possible affine functions. It is well-known that $\texttt{NL}_F$ is upper bounded by $2^n - 2^{\frac{n}{2}-1}$. Vectorial Boolean functions that achieve $\texttt{NL}_F = 2^n - 2^{\frac{n}{2}-1}$ are called *bent*. Bent functions exist only when n is even and are important for building balanced S-boxes.
  In practice, the nonlinearity of the vectorial Boolean function F is studied via the linear probability table (LPT) defined for the entry $(a, b) \in \mathbb{F}_{2^n}^2$ by:

  $$\texttt{LPT}_F(a, b) = \left(\frac{\#\left\{x \in \mathbb{F}_2^n : a \cdot x + b \cdot F(x) = 0\right\}}{2^{n-1}} - 1 \right)^2.$$

  For AES, except for the first row and first column, all rows and columns of the LPT have the same distribution of values as given in Table 8.

Table 8. Distribution of the linear probability values of AES.

Value	**0**	$\left(\frac{1}{64}\right)^2$	$\left(\frac{2}{64}\right)^2$	$\left(\frac{3}{64}\right)^2$	$\left(\frac{4}{64}\right)^2$	$\left(\frac{5}{64}\right)^2$	$\left(\frac{6}{64}\right)^2$	$\left(\frac{7}{64}\right)^2$	$\left(\frac{8}{64}\right)^2$
Frequency	17	48	36	40	34	24	36	16	5

- *Differential attack.* In 1991, Biham and Shamir [64] proposed differential cryptanalysis and applied it to DES. Differential cryptanalysis is a chosen-plaintext attack and works with two pairs of plaintext (P_1, P_2) with a fixed difference $a = P_1 + P_2$ and their corresponding ciphertext (C_1, C_2). The goal of the differential cryptanalysis is to study the behavior of the difference $b = P_1 + P_2$.

  For a vectorial Boolean function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, the differential cryptanalysis is studied via the difference distribution table (DDT), which is defined for $(a, b) \in \mathbb{F}_{2^n}^2$ by:

$$\mathtt{DDT}_F(a,b) = \#\left\{x \in \mathbb{F}_{2^n} : F(x) + F(x+a) = b\right\}.$$

  The differential uniformity of F is defined by:

$$\delta_F = \max_{a \in \mathbb{F}_{2^n}^*} \mathtt{DDT}_F(a,b).$$

  The differential cryptanalysis exploits the differential probability DP_F, specifically:

$$\mathtt{DP}_F(a,b) = \frac{\#\left\{x \in \mathbb{F}_{2^n} : F(x) + F(a+x) = b\right\}}{2^n}.$$

  For a randomly chosen permutation and for any $a \in \mathbb{F}_{2^n} \backslash \{0\}$, the value $F(x) + F(a+x)$ is expected to be uniformly distributed with equiprobability. This makes $\mathtt{DDT}_F(a,b)$ a reliable and practical distinguisher if $\mathtt{DP}_F(a,b)$ is sufficiently small.

  For the AES S-box, Table 9 shows the distribution of the $\mathtt{DP}_F$ values and their frequencies.

Table 9. Distribution of the differential probability values of AES.

Value	**0**	$\frac{2}{256}$	$\frac{4}{256}$	**1**
Frequency	33,150	32,130	255	1

  If x_0 is a solution to the equation $F(x) + F(a+x) = b$, then $x_0 + a$ is also a solution. This implies that $\mathtt{DDT}_F(a,b) \geq 2$ for all $a \neq 0$ and, consequently, $\delta_F \geq 2$. Vectorial Boolean functions satisfying $\delta_F = 2$ are called almost perfect nonlinear (APN) functions. As shown in Table 9, the differential uniformity of the AES S-box is 4. Hence, AES does not belong to the APN family; nevertheless, its differential uniformity is too small. This makes AES resistant to differential cryptanalysis.

5.4 Applications of AI to Block Ciphers

There are plenty of attacks on AES that can be performed by AI. The goal of using AI with AES is to test the resistance of its secret keys and its S-boxes to such attacks. AI can be used for the following tasks.

1. *Resistance to side-channel attacks* [65]: Side-channel attacks exploit the operations performed by a cryptographic system during encryption or decryption to gain information about the private key. The most used channel attacks are timing attacks, simple power attacks, differential power attacks, electromagnetic radiation attacks, correlation power attacks, etc. These attacks rely on collecting and interpreting observations in order to infer information about key size and bits. These inferences lend themselves naturally to ML and ANNs in general and to advanced ANNs/models in particular. As described earlier, some work has already been initiated in this direction [53].
2. *Resistance to fault attacks* [66]: Fault attacks are deployed to disturb the normal functioning of a cryptosystem. They are injected by various techniques such as laser, light pulses, electromagnetic perturbations, tampering with the clock, etc. This enables the attacker to collect the erroneous result and to gain information about the private key. As with side-channel attacks, fault attacks can be overcome by testing implementations against an advanced ANN that tries to leverage the erroneous results to infer information about the key. AES cipher implementations need to be tested against an advanced ANN model that tries to leverage collected output to infer the key before deployment.
3. *Resistance to linear attacks* [63]: This task can be processed by computing the linear probability table of the S-box. ANNs as excellent function approximators can be used to model nonlinearity of S-boxes, similarly to the work of [54] on DES.
4. *Resistance to differential attacks* [64]: This task can be performed by computing the difference distribution table of the S-box. As with linear attacks, ANNs as excellent function approximators can be used to model the differential properties of S-boxes, similarly to what has been done by [56] on the round-reduced Speck cipher and by [48] on the round function of GIFT.
5. *Resistance to truncated differentials* [67]: This variant of the differential attack was presented by Knudson in 1994. This task can be processed by adapting the difference distribution table of the S-box under the truncated differentials criteria. As with differential attacks, ANNs as excellent function approximators can be used to model the truncated differential properties of S-boxes.
6. *Resistance to boomerang attacks* [68]: The task of testing the boomerang cryptanalysis can be accomplished by studying the boomerang connectivity table (BCT) as defined by Cid et al. in 2018 [69]. The BCT of an invertible vectorial function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined at the entry $(a, b) \in \mathbb{F}_{2^n}$ by:

$$\mathrm{BCT}_F(a,b) = \#\left\{x \in \mathbb{F}_{2^n} \ : \ F^{-1}(F(x)+b) + F^{-1}(F(x+a)+b) = a\right\}.$$

7. *Algebraic immunity* [70,71]: The algebraic immunity of a vectorial Boolean function F defined on F_{2^n} is the lowest degree of all functions $G \neq 0$ satisfying $F(x) \cdot G(x) = 0$ or $(1+F(x)) \cdot G(x) = 0$, where $a \cdot b$ is the inner product of the vectors a and b. The underlying vectorial Boolean function of AES can be modeled and tested using advanced ANNs.
8. *Balancedness* [72]: A vectorial Boolean function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$ is balanced if every value of $\mathbb{F}_{2^m}$ is the image of exactly 2^{n-m} values from $\mathbb{F}_{2^n}$. The task of verifying balancedness can be processed by studying the vectorial Boolean function that defines the S-box of AES.
9. *Resistance to other attacks:* There are plenty of attacks and criteria that can be implemented with AI to test the security of block ciphers. This includes correlation immunity [73], strict avalanche criterion (SAC) [74], fixed points and opposite fixed points [60], algebraic degree [73], impossible differential [75], etc. A complete list of such attacks can be found in [7,76].

In sum, AI can be used to test AES `SubBytes()` and `MixColumns()` functions, and the AES cipher with its modes of operations and their implementations can be used to test against all former attacks and to propose useful and efficient solutions, such as the choice of the key space, `MixColumns()` matrix polynomials, etc., that nullify/undermine the attacks.

6 The RSA Cryptosystem

In 1978, Rivest, Shamir, and Adleman [8] introduced RSA, a public key and digital signature scheme. RSA is used in various industrial applications, such as privacy, VPNs, communication channels, email services, cybersecurity, and web browsers.

6.1 The RSA Encryption Scheme

The RSA encryption scheme is composed of three algorithms.

1. *Key Generation:* Given a parameter n,
   - Select a random prime number p of bit size n.
   - Select a random prime number q of bit size n with $p \neq q$.
   - Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
   - Select a number e such that $\gcd(e, \phi(N)) = 1$.
   - Compute a number d such that $ed \equiv 1 \pmod{\phi(N)}$.
   - Publish the public key (N, e).
2. *Encryption:* Given a public key (N, e) and a message $M \in \mathbb{Z}/N\mathbb{Z}$,
   - Compute the ciphertext $C \equiv M^e \pmod{N}$.
3. *Decryption:* Given the private key (N, d) and a ciphertext C,
   - Compute $M \equiv C^d \pmod{N}$.

The correctness of the decryption works following Euler's Theorem,

$$C^d \equiv M^{ed} \equiv M^{k\phi(N)} M \equiv M \pmod{N},$$

where k is the integer such that $ed = 1 + k\phi(N)$.

6.2 Attacks on RSA

In RSA, there are originally three parameters: a modulus $N = pq$ with two large prime numbers p and q, a public exponent e satisfying $\gcd(e, (p-1)(q-1)) = 1$, and a private exponent d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$. This modular equation can be rewritten as $ed - k\phi(N) = 1$ and is called *the key equation.* Since its invention in 1978, RSA has been intensively cryptanalyzed by various methods [9,10,12]. We describe below some of these attacks. The prominent attacks on RSA can be categorized into three groups:

- *Factorization attacks.* The most obvious attack on RSA is to factor its modulus N. Nevertheless, since N is the product of two balanced large prime numbers, no known method is efficient to factor RSA moduli of size 1024 bits or more. There are several algorithms devoted to factoring integers, such as the Number Field Sieve method [77], Pollard's Rho method [78], the Elliptic Curve Method [79], and others, with different running times as presented in Table 10.

Table 10. Algorithms to factor an integer n with running times.

Algorithm	**Running Time Complexity**	**Nature of the Factor p**
Pollard's Rho [78]	$O\left(\sqrt{p}\right)$	largest prime factor
Elliptic Curve Method [79]	$O\left(e^{(1+o(1))\sqrt{2\log(p)\log(\log(p))}}\right)$	smallest prime factor
Number Field Sieve [77]	$O\left(e^{1.923\log(n)^{\frac{1}{3}}\log\log(n)^{\frac{2}{3}}}\right)$	any factor
Quadratic Sieve [80]	$O\left(e^{(1+o(1))\sqrt{\log(n)\log(\log(n))}}\right)$	any factor

Despite the existence of such factorization algorithms, there is no known non-quantum-based method that can efficiently factor an RSA modulus of more than 1024 bits. The latest record for integer factorization was obtained in 2020 by Boudot et al. [81], who factored RSA-250, an RSA modulus with 829 bits.

- *Algebraic attacks.* Such attacks are based on the mathematical structure of the cryptosystem. Typically, for RSA, the algebraic attacks are related to the key equation $ed - k\phi(N) = 1$. In 1996, Coppersmith [82] proposed a method to solve certain polynomial equations and applied it to factor an RSA modulus if half of the bits of one of the prime factors were known. Since then, various generalizations of Coppersmith's method have been proposed [9,10,83,11,12].

  In 1990, Wiener [84] showed that using RSA with a small private exponent is insecure. Using the key equation $ed - k\phi(N) = 1$ with $\phi(N) = (p-1)(q-1) = N + 1 - (p+q) \approx N$, he showed that if p and q have the same bit size, and if $d < \frac{1}{3}N^{\frac{1}{4}}$, then:

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2},$$

  which implies that $\frac{k}{d}$ is one of the convergents of the continued fraction expansion of $\frac{e}{N}$. The convergents of $\frac{e}{N}$ can efficiently be computed by applying the continued fraction algorithm. In 1996, Boneh and Durfee [85] improved the bound up to $d < N^{0.292}$ by applying Coppersmith's method and lattice reduction techniques.
- *Side-channel attacks.* The modular exponentiation is a crucial operation in RSA and must be implemented securely to prevent side-channel attacks. The application of side-channel attacks against RSA started in 1996 with the work of Kocher [26]. Since then, numerous studies have been conducted to make side-channel attacks infeasible against RSA [86,87,88,89].

  For the RSA cryptosystem, the running time during the decryption process can leak information about the private key. This method is known as a *timing attack* and is one of the most popular side-channel attacks. In RSA, the timing attack concerns the modular exponentiation if the square-and-multiply method is used. To compute $m^d \pmod{N}$, the square-and-multiply method consists of expanding $d = (d_{r-1}d_{r-2}\cdots d_0)_2$ in base 2, taking $a = 1$, and then, for i from $r-1$ down to 0, computing $a \equiv a^2 \pmod{N}$; additionally, if $d_i = 1$, $a \equiv am \pmod{N}$. The drawback of this method is that the computation time is not the same when $d_i = 1$ and $d_i = 0$. This can be exploited to guess the binary decomposition of d and then to compute d. To ovoid timing attacks, there are various implementations of the modular exponentiation, such as square-always exponentiation [90].

6.3 Applications of AI to RSA

1. *Resistance to side-channel attacks* [26]: RSA is vulnerable to side-channel attacks depending on its arithmetic operations, especially during the decryption process. Numerous studies have been proposed to protect

it from side-channel attacks [88,89,91]. As with side-channel attacks on AES, advanced ANNs can be used to test the RSA cryptosystem and its implementations against the side-channel attacks before deployment. Some work has already been done in this direction. Ref. [58] used deep learning in side-channel attacks against a secure implementation of the RSA algorithm.

2. *Resistance to fault attacks* [92]: In addition to side-channel attacks, RSA is vulnerable to fault attacks [93,94]. There are many techniques to force faults, such as variations in the clock, laser, X-rays, voltage, etc. These attacks also lend themselves to the use of advanced ANNs to infer key bits or plaintext from the collected output resulting from the faults.
3. *Resistance to factorization attacks:* The security of RSA is partly based on the difficulty of factoring its modulus N. Obviously, the bit size of N is crucial against factoring algorithms, such as the Number Field Sieve and the Elliptic Curve method. The current recommendation for the size of the RSA modulus is at least 3000 bits [95]. Some initial work has been conducted in this direction by [96,97], but more is needed in order to strengthen the choice of primes p and q.
4. *Resistance to Fermat's factoring method* [98]: This method is based on solving the equation $N = x^2 - y^2 = (x-y)(x+y)$, which leads to $p = \frac{x+y}{2}$, $q = \frac{x-y}{2}$. If the difference $|p-q|$ is too small relative to N, then y is too small, and $\sqrt{N}$ is an approximation of x. This can be exploited to retrieve x, y, and the prime factors from p and q. The method works efficiently when $|p-q| < N^{\frac{1}{4}}$. AI can be used to learn x and y for different Ns and to eliminate the RSA prime factors p and q that are vulnerable to Fermat's factoring method during the generation phase. Furthermore, biases in the distribution of consecutive primes [99] can be learned using an advanced ANN to help reduce the search space in factorization and Fermat's factoring attacks.
5. *RSA with existing modulus:* If $N_1 = p_1 q_1$ is the RSA modulus of two independent entities, then both entities know the prime factors and can decrypt the encrypted messages of each other. Unfortunately, AI cannot help guard against this scenario. Luckily, the likelihood that two organizations generate the same primes p and q is extremely slim, knowing that p and q are on the order of 2^{1024}.
6. *RSA moduli with common factors:* If $N_1 = pq_1$ and $N_2 = pq_2$ are two RSA moduli, then an attacker can compute $p = \gcd(N_1, N_2)$, $q_1 = \frac{N_1}{p}$, and $q_2 = \frac{N_2}{p}$. This factors the two moduli. To generate a safe RSA modulus N, testing whether N is coprime to every modulus in the list of collected moduli can be efficiently performed by using the method of Bernstein [100,101] without the need for AI.
7. *RSA moduli with primes sharing most, middle, or least significant bits:* If $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ are two RSA moduli, where $p_1 \approx p_2$ share an amount of their least, middle, or most significant bits, then one can apply the method of May and Ritzenhofen [102] or the method of Faugère et al. [103] to factor N_1 and N_2. Here too, the factorization problem can be posed as an approximation function implemented using ANNs, leading to the elimination of the prime factors that share a significant number of their least significant bits.
8. *Resistance to small private exponents:* The private exponent in RSA with a modulus $N = pq$ and a public exponent e is the integer d satisfying $ed - k(p-1)(q-1) = 1$. Because of the attack of Wiener [84], and the attack of Boneh–Durfee [85], it is required that d be larger than $\sqrt{N}$. Nevertheless, in many instances, one can find the value d even if d is arbitrarily large [104,105]. AI can be used to build an approximation function using advanced ANNs for solving the equation above and using it to test the resistance of a generated RSA modulus to such attacks.
9. *Resistance to partial key exposure attacks:* When a fraction of the most significant or the least significant bits of the private exponent d is guessed by an attacker, then Coppersmith's method can be used to retrieve d entirely [106,107,108]. An ANN approximator for learning d from its fractions and known ciphertext plaintext pairs can be used to test any generated private key d against such attacks before using it for practical applications.

7 Learning with Errors

In 2005, Regev [13] introduced the Learning With Errors problem (LWE). It has become an important computational problem in lattice-based cryptography.

7.1 Description of Learning with Errors

An instance of LWE is parameterized by a positive integer m, a prime number q, and a probability distribution χ over $\mathbb{Z}_q$, the ring of integers modulo q. A typical example of a probability distribution is the continuous Gaussian distribution centered in $c \in \mathbb{R}^n$ with a parameter $\sigma > 0$. It is defined for a vector $x \in \mathbb{R}^n$ by:

$$\chi_{c,\sigma}(x) = \frac{1}{\sigma^n} e^{-\pi \| \frac{x-c}{\sigma} \|^2}.$$

There are two main equivalent sub-problems in LWE, Search LWE and Decision LWE, which are known to be equivalent.

- Search LWE can be summarized as follows. Let χ be a probability distribution over $\mathbb{Z}_q$. Given a matrix $A \in \mathbb{Z}_q^{m\times n}$ and a vector $b \in \mathbb{Z}_q^m$ whose entries are chosen uniformly, find a vector $s \in \mathbb{Z}_q^n$ such that $As + e = b$, where $e \in \mathbb{Z}_q^m$ is a vector generated by χ.
- Decision LWE can be summarized as follows. Given a matrix $A \in \mathbb{Z}_q^{m\times n}$ and a vector $b \in \mathbb{Z}_q^m$, determine whether $(A, b) \in L_1$ or $(A, b) \in L_2$, where L_1 is the set of all tuples $(A, b) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ generated by uniformly random distribution and L_2 is the set of all tuples $(A, b) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, such that $b = As + e$ for a vector $s \in \mathbb{Z}_q^n$, uniformly distributed, and $e \in \mathbb{Z}_q^m$, generated by χ.

The first cryptosystem based on LWE was presented by Regev in 2005. It is parameterized by the four parameters m, n, q, and χ, with q prime, $m \geq 4(n + 1)\log(q)$, and χ as a probability distribution, such that a vector $e \in \mathbb{Z}_q^m$ generated by χ satisfies $\|e\| < B < \frac{1}{4mq}$ with overwhelming probability. The system is composed of four algorithms, which can be summarized as follows.

1. *Key Generation:* Given the parameters m, n, q, and χ,
   - Select a matrix $A \in \mathbb{Z}_q^{m\times n}$ at random.
   - Select a secret vector $s \in \mathbb{Z}_q^n$.
   - Select a private vector $e \in \mathbb{Z}_q^m$ according to a probability distribution χ over $\mathbb{Z}_q$.
   - Compute $b \in \mathbb{Z}_q^m$, such that $b = As + e$, and publish the public key (A, b).
2. *Encryption:* Given the parameters m, n, q, χ, a public key (A, b), and a message $M \in \{0, 1\}$,
   - Select a vector $r \in \mathbb{Z}_q^m$ at random.
   - Compute the ciphertext (C_1, C_2), where $C_1 = r^tA \in \mathbb{Z}_q^n$ and $C_2 = r^tb + \left\lfloor \frac{q}{2} \right\rfloor M \in \mathbb{Z}_q$. Here, x^t represents the transpose of x.
3. *Decryption:* Given the parameters m, n, q, χ, a ciphertext (C_1, C_2), and a secret key s,
   - Compute $u = C_2 - C_1 s \in \mathbb{Z}_q$.
   - If $|u| \leq \frac{q}{4}$, then the decryption is 0, else the decryption is 1.

The correctness of the decryption depends on the size of u. Indeed,

$$u = C_2 - C_1 s = r^te + \left\lfloor \frac{q}{2} \right\rfloor M,$$

and if e satisfies $\|e\| < B$ with high probability, then $\|r^te\| < mB < \frac{q}{4}$, and the decryption occurs.

7.2 Hardness of LWE

The security of LWE is based on the hardness of various open problems in lattice reduction theory. A lattice $\mathcal{L} \subset \mathbb{R}^n$ is a discrete subgroup of $\mathbb{R}^n$ that is generated by m vectors $u_1, \ldots, u_m \in \mathbb{R}^n$ using integer coefficients; that is,

$$\mathcal{L} = \left\{ \sum_{i=1}^{m} a_i u_i \;\middle|\; a_i \in \mathbb{Z} \right\}.$$

The set $B = \{u_1, \ldots, u_m\}$ is called a *basis* of $\mathcal{L}$, m is its *rank*, and n is its *dimension.* When $n = m$, the lattice is called a *full-rank lattice.*

Lattices have plenty of properties and hard unsolved problems that are used to build cryptosystems that are still resistant, even to quantum computers. The most known and used hard problems in lattice theory are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Both problems use the minimum distance $\lambda_1(\mathcal{L})$ of $\mathcal{L}$, which is defined by:

$$\lambda_1(\mathcal{L}) = \min_{v \in \mathcal{L} \setminus \{0\}} \|v\|,$$

where $\|v\|$ is the Euclidean norm defined for $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ by $\|v\| = \left(\sum_{i=1}^{n} v_i^2\right)^{\frac{1}{2}}$.

- *Shortest Vector Problem (SVP)*: Let $\mathcal{L}$ be a lattice with a basis B. Find the shortest nonzero lattice vector $u \in \mathcal{L}$ with $\|u\| = \lambda_1(\mathcal{L})$.
- *Closest Vector Problem (CVP)*: Let $\mathcal{L}$ be a lattice with a basis B and $v \notin \mathcal{L}$ be a vector. Find a lattice vector $u \in \mathcal{L}$ such that $\|u - v\| \leq \lambda_1(\mathcal{L})$.

The security of LWE is based on two sub-problems in lattices: the Decisional Approximate SVP ($\texttt{GapSVP}_\gamma$) and the Approximate Shortest Independent Vectors Problem ($\texttt{SIVP}_\gamma$), where $\gamma \geq 1$ is a positive real parameter.

- *Decisional Approximate SVP* ($\texttt{GapSVP}_\gamma$): Let $\mathcal{L}$ be a lattice with a basis B and $r > 0$ be a real number. Decide whether $\lambda_1(\mathcal{L}) \leq r$ or $\lambda_1(\mathcal{L}) > \gamma r$.
- *Approximate Shortest Independent Vectors Problem* ($\texttt{SIVP}_\gamma$): Let $\mathcal{L}$ be a full-rank lattice with dimension n and a basis B. Find n linearly independent vectors $v_i \in \mathcal{L}$ such that $\|v_i\| \leq \gamma \lambda_n(\mathcal{L})$, where $\lambda_n(\mathcal{L})$ is the n-th successive minimum of the lattice.

In 2005, Regev [13] showed that, when the LWE error e is generated by a Gaussian distribution with a parameter $\sigma = \alpha q$, where $\frac{\sqrt{n}}{q} < \alpha < 1$, then solving LWE implies a quantum solution of $\texttt{GapSVP}_\gamma$ and $\texttt{SIVP}_\gamma$ over n-dimensional lattices in the worst case for $\gamma = \widetilde{O}(n/\alpha)$, where $\widetilde{O}(\cdot)$ is a function with various polylogarithmic factors. In 2009, Peikert [109] showed that classical reductions are possible from the worst-case hardness of the GapSVP problem to the search version of LWE when the modulus q is exponential in the dimension n, especially when $q \geq 2^{\frac{n}{2}}$. In 2013, Brakerski et al. [110] showed that LWE is classically at least as hard as standard worst-case lattice problems with any subexponential modulus.

7.3 Applications of AI to LWE

The security of LWE comes from its reduction to worst-case lattice problems. Such problems are believed to be hard for both classical and quantum computers. As a consequence, there is a very limited number of attacks that can be launched against LWE. While the theoretical security of LWE depends on hard problem in lattices, its practical security depends on the parameters used in a specific instantiation. Intuitively, both $\texttt{GapSVP}_\gamma$ and $\texttt{SIVP}_\gamma$ problems can benefit from the power of advanced ANNs as approximators. The set of parameters that are vulnerable to solutions of $\texttt{GapSVP}_\gamma$ and $\texttt{SIVP}_\gamma$ by an advanced ANN should be discarded. To date, no such attempts can be found in the literature.

8 The Ascon Family of Ciphers

In this section, we describe Ascon [24], the family of authenticated encryption and hashing algorithms selected by NIST for future standardization of lightweight cryptography.

8.1 Description of Ascon

Ascon is a family of several algorithms devoted to different tasks. The family includes Ascon-128 and Ascon-128a authenticated ciphers, the Ascon-Hash hash function, and the Ascon-Xof extendable output function. They ensure 128-bit security and use a common 320-bit permutation. All of the algorithms operate at 320-bit states. Each state S is divided into an inner part S_r of size r bits and an outer part S_c of size $c = 320 - r$ bits, where r depends on the Ascon variant. Moreover, each state is divided into five 64-bit registers $x_0, \cdots, x_4$, such that:

$$S = S_r \| S_c = x_0 \| x_1 \| x_2 \| x_3 \| x_4.$$

The authenticated encryption design of Ascon is parameterized by a key bit length of $k \leq 160$, a rate r, and two integers a and b. The integers a and b serve to count the compositions of a permutation p of the set $\{0,1\}^{320}$. The permutation p is the composition of three permutations, specifically:

$$p = p_L \circ p_S \circ p_C,$$

where p_C is a constant addition, p_S is a substitution layer, and p_L is a linear diffusion layer (see [24] for more details).

8.2 Ascon Encryption

The encryption process of Ascon starts with four initial parameters:

- A key $K \in \{0,1\}^k$ with $k \leq 160$;
- A nonce $N \in \{0,1\}^{128}$;
- Associated data $A \in \{0,1\}^*$;
- A plaintext $P \in \{0,1\}^*$.

It is split into four phases:

1. *Initialization:* The 320-bit initial state in Ascon is built by the concatenation of an initial vector IV, a secret k-bit key K, and a 128-bit nonce N; that is:

$$S = IV \| K \| N.$$

The initial vector IV has the form:

$$IV = k \| r \| a \| b \| 0^{160-k},$$

where a is the initialization and finalization round number, and b is the intermediate round number. After initializing the state, a permutation p is applied to it a times, followed by XORing the key K so that the output has the form:

$$S \leftarrow p^a(S) \oplus \left(0^{320-k} \| K\right).$$

2. *Processing Associated Data:* If associated data $A \in \{0,1\}^*$ is provided and not null, they are appended with a single 1 and $r-1-(|A| \pmod r)$ 0s, then split into s blocks of size r so that:

$$A \leftarrow A\|1\|0^{r-1-(|A| \pmod r)} = A_1\|A_2\|\cdots\|A_s.$$

If A is empty, then $s=0$. For each $i=1,\cdots,s$, the following calculation is performed:

$$S \leftarrow p^b((S_r \oplus A_i)\|S_c).$$

After all calculations, the state is transformed as:

$$S \leftarrow S \oplus \left(0^{319}\|1\right).$$

3. *Plaintext Processing:* The plaintext $P \in \{0,1\}^*$ is also appended with a single 1 and $r-1-(|P| \pmod r)$ 0s, and then is split into t blocks of size r, so that:

$$P \leftarrow P\|1\|0^{r-1-(|P| \pmod r)} = P_1\|P_2\|\cdots\|P_t.$$

Then, for $i=1,\cdots,t$, the following calculations are performed:

$$C_i \leftarrow S_r \oplus P_i,$$
$$S \leftarrow \begin{cases} p^b(C_i\|S_c) & \text{if } 1 \leq i < t, \\ C_i\|S_c & \text{if } i = t. \end{cases}$$

After processing C_t, the value $\tilde{C_t} = \lfloor S_r \oplus P_t \rfloor_{|P| \pmod r}$ is calculated, where $\lfloor x \rfloor_k$ is the bitstring x truncated to the most significant k bits.

4. *Finalization:* In this phase, the following values are calculated:

$$S \leftarrow p^a\left(S \oplus \left(0^r\|K\|0^{320-r-k}\right)\right),$$
$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128},$$

where $\lceil x \rceil^k$ is the bitstring x truncated to the least significant k bits. The ciphertext is finally composed as:

$$C = C_1\|\cdots\|C_{t-1}\|\tilde{C_t},$$

which is transmitted together with the tag T.

8.3 Ascon Decryption

In Ascon, the decryption algorithm starts with the following parameters, already fixed in the encryption algorithm:

- The key $K \in \{0,1\}^k$ with $k \leq 160$;
- The nonce $N \in \{0,1\}^{128}$;
- The associated data $A \in \{0,1\}^*$;
- The ciphertext $C \in \{0,1\}^*$;
- The tag $T \in \{0,1\}^{128}$.

The decryption is nearly identical to encryption. More precisely, the initialization and the processing associated data are identical, while plaintext processing is replaced by ciphertext processing as follows:
First, the ciphertext $C \in \{0,1\}^*$ is split into t blocks of size r so that:

$$C = C_1 \| C_2 \| \cdots \| C_{t-1} \| \tilde{C}_t,$$

with $0 \leq |\tilde{C}_t| < r$.

Second, for $i = 1, \cdots, t-1$, the following calculations are performed:

$$\begin{aligned} P_i &\leftarrow S_r \oplus C_i, \\ S &\leftarrow p^b \left(C_i \| S_c \right). \end{aligned}$$

Then, two values $\tilde{P}_t$ and S_r are computed as:

$$\begin{aligned} \tilde{P}_t &\leftarrow \lfloor S_r \rfloor_{|\tilde{C}_t|} \oplus \tilde{C}_t, \\ S_r &\leftarrow S_r \oplus \left(\tilde{P}_t \| 1 \| 0^{r-|\tilde{C}_t|-1} \right). \end{aligned}$$

Next, the following values are computed:

$$\begin{aligned} S &\leftarrow p^a \left(S \oplus \left(0^r \| K \| 0^{320-r-k} \right) \right), \\ T^* &\leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}. \end{aligned}$$

Finally, if $T = T^*$, then return $P_1 \| \cdots \| P_{t-1} \| \tilde{P}_t$, else return $\perp$.

8.4 Security of Ascon

The hardness of Ascon is tightly linked to the choice of the parameters a, b, and r and to the key size k used to perform the encryption and the decryption. To guarantee 128-bit security, the recommended parameters are listed in Table 11.

Table 11. Parameters for Ascon-128 and Ascon-128a.

Algorithm	**Key**	**Nonce**	**Tag**	**Data**	$\boldsymbol{a}$	$\boldsymbol{b}$	**Rate** $\boldsymbol{r}$
Ascon-128	128	128	128	64	12	6	64
Ascon-128a	128	128	128	128	12	8	128

Since its selection as the winner of the CAESAR competition, the Ascon family has been intensively analyzed for vulnerabilities. Section 6 of [24] presents an overview of the security analysis and resiatance to attacks. All published results so far support its security and efficiency.

8.5 Applications of AI to Ascon

The Ascon family uses a 5-bit S-box

[24] that needs to be immune against known attacks for substitution layers p_S, such as the linear and differential attacks. Subsection 5.4 presents an exhaustive list of these attacks that are also applicable to the Ascon lightweight cipher.

To this end, AI can be used to test the Ascon S-box, as well as the full or partial transformation process reflected by small a and b values against all these attacks, in order to propose the set of parameters that nullify/undermine these attacks.

9 Conclusions

Artificial intelligence (AI) and, in particular, deep learning using sophisticated artificial neural network (ANN) architecture, is exponentially developing and gaining practical use in all sectors of daily life. In this paper, we presented areas where the use of AI can help enhance the security of cryptographic systems. We particularly focused on four prominent systems in modern cryptography, namely, the Advanced Encryption Standard (AES), the Rivest–Shamir–Adleman (RSA) scheme, the Learning With Errors (LWE) scheme, and the lightweight Ascon cipher family. We reviewed their security and pinpointed layers, functions, and areas that could potentially benefit from cryptanalysis that uses advanced ANN architectures. This said, depending on the function to approximate S-box and vectorial Boolean functions for AES, S-box and permutations for Ascon, Diophantine equations and factorization for RSA, lattice problems for LWE), ANNs may not necessarily outperform other machine learning (ML) techniques. For instance, LWE introduces vectors of errors similar to noise in the encryption process, which may hinder the performance of ANNs, since it is a well-known fact that ANNs suffer from noisy training data. Experimentation is needed to confirm this hypothesis. Furthermore, sophisticated ANN architectures can have the tendency to overfit the presented training data, which may lead to errors for unseen encrypted data or plaintext.

Finally, beyond prediction, ANNs do not provide any insights into the structure of the function being approximated, which may not help in fine-tuning the function/layer being approximated (see [2] for Explainable AI). For further research, we envisage experimenting with different ANN architectures and building an ANN generator that automatically generates an adversary ANN from the specification of the S-box or the vectorial Boolean function to help cryptosystem designers quickly test the strength of the cryptographic functions and substitution layers.

References

1. Goldman, S. Foundation models: 2022’s AI paradigm shift. VentureBeat. `https://venturebeat.com/ai/foundation-models-2022s-ai-paradigm-shift/` (Accessed on 05 August 2023).
2. Phillips, P.; Hahn, C.; Fontana, P.; Yates, A.; Greene, K.; Broniatowski, D.; Przybocki, M. (2021-09-29). Four Principles of Explainable Artificial Intelligence. `https://doi:10.6028/nist.ir.8312`.
3. Rivest, R.L. Cryptography and machine learning. In *Advances in Cryptology—ASIACRYPT ’91,* Proceedings of the ASIACRYPT 1991; Fujiyoshida, Japan, November 11-14, 1991 ; Imai, H., Rivest, R.L., Matsumoto, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1991; Volume 739.
4. Ertel, W. *Introduction to Artificial Intelligence*, 2nd ed.; Undergraduate Topics in Computer Science; Springer: Cham, Switzerland, 2017.
5. Tencent Research Institute; CAICT; Tencent AI Lab; Tencent Open Platform (Eds.) *Artificial Intelligence, A National Strategy*; Palgrave Macmillan: Singapore, 2021. `https://doi.org/10.1007/978-981-15-6548-9`.
6. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654.
7. Picek, S.; Batina, L.; Jakobović, D.; Ege, B.; Golub, M. S-box, SET, Match: A Toolbox for S-box Analysis. In *Information Security Theory and Practice: Securing the Internet of Things, Proceedings of the WISTP*; Heraklion,

Crete, Greece, June 30 - July 2, 2014; Naccache, D., Sauveron, D., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8501.

8. Rivest, R.; Shamir, A.; Adleman, L. A Method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126.
9. Boneh, D. Twenty years of attacks on the RSA cryptosystem. *Not. Am. Math. Soc.* **1999**, *46*, 203–213.
10. Hinek, M.J. *Cryptanalysis of RSA and Its Variants*; Chapman & Hall/CRC Cryptography and Network Security; CRC Press: Boca Raton, FL, USA, 2009.
11. Nitaj, A. The Mathematical Cryptography of the RSA Cryptosystem. In *Cryptography: Protocols, Design and Applications*; Lek, K., Rajapakse, N., Eds.; Nova Science Publishers: Hauppauge, NY, USA, 2012.
12. Mumtaz, M.; Ping, L. Forty years of attacks on the RSA cryptosystem. *J. Discret. Math. Sci. Cryptogr.* **2019**, *22*, 9–29. https://doi.org/10.1080/09720529.2018.1564201.
13. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th ACM Symposium on Theory of Computing (STOC), Baltimore, MD, USA, May 22-24, 2005; pp. 84–93.
14. Peikert, C. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.* **2016**, *10*, 283–424.
15. National Institute of Standards and Technology. Post-Quantum Cryptography. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography (Accessed on 30 June 2023).
16. 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 Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, United Kingdom, April 24, 2018; pp. 353–367.
17. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, 238–268. https://doi.org/10.13154/tches.v2018.i1.238-268.
18. Prest, T.; Fouque, P.-A.; Hoffstein, J.; Kirchner, P.; Lyubashevsky, V.; Pornin, T.; Ricosset, T.; Seiler, G.; Whyte, W.; Zhang, Z. FALCON. National Institute of Standards and Technology. 2020. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions (Accessed on 30 June 2023).
19. Bernstein, D.J.; Hülsing, A.; Kölbl, S.; Niederhagen, R.; Rijneveld, J.; Schwabe, P. The SPHINCS+ Signature Framework. Cryptology ePrint Archive, Paper 2019/1086. 2019. Available online: https://eprint.iacr.org/2019/1086 (Accessed on 30 June 2023).
20. Stehlé, D.; Steinfeld, R.; Tanaka, K.; Xagawa, K. Efficient public key encryption based on ideal lattices. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009; pp. 617–635.
21. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In Proceedings of the Advances in Cryptology—EUROCRYPT; French Riviera, May 30 - June 3, 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
22. Langlois, A.; Stehlé, D. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* **2015**, *75*, 565–599.
23. Bruna, J.; Regev, O.; Song, M.J.; Tang Y. Continuous LWE. *arXiv* **2020**, arXiv:2005.09595.
24. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. ASCON v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.* **2021**, *34*, 33. https://doi.org/10.1007/s00145-021-09398-9.
25. Daniel J. Bernstein, The CAESAR Committee Secretary. Caesar: Competition for Authenticated Encryption: Security, Applicability, and Robustness. 2014.https://competitions.cr.yp.to/caesar.html (Accessed on 30 June 2023).
26. Kocher, P. Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems. In Proceedings of the CRYPTO'96, Santa Barbara, California, USA, August 18âĂŞ22, 1996; Volume 1109, pp. 104–113.
27. Kocher, P.C.; Jaffe, J.; Jun, B.; Rohatgi, P. Introduction to differential power analysis. *J. Cryptogr. Eng.* **2011**, *1*, 5–27.
28. van Eck, W. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk. *Comput. Secur.* **1985**, *4*, 269–286.
29. Biham, E.; Shamir, A. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97, Proceedings of the CRYPTO*;Santa Barbara, California, USA, August 17-21, 1997; Kaliski, B.S., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1294.

30. Genkin, D.; Shamir, A.; Tromer, E. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. Cryptology ePrint Archive, Paper 2013/857. 2013. Available online: `https://eprint.iacr.org/2013/857` (Accessed on 30 June 2023).
31. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 4th ed.; Prentice Hall: Hoboken, NJ, USA, 2020.
32. Zheng, B.; Verma, S.; Zhou, J.; Tsang, I.W.; Chen, F. Imitation Learning: Progress, Taxonomies and Challenges. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. `https://doi.org/10.1109/TNNLS.2022.3213246`.
33. Mukhamediev, R.I.; Popova, Y. Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges. *Mathematics* **2022**, *10*, 2552. `https://doi.org/10.3390/math10152552`.
34. Li, J.; Cheng, J.; Shi, J.; Huang, F. Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. In *Advances in Computer Science and Information Engineering*; Jin, D., Lin, S., Eds.; Advances in Intelligent and Soft Computing; Springer: Berlin/Heidelberg, Germany, 2012; Volume 169.
35. Cybenko G. Approximation by Superpositions of Sigmoidal Function. *Math. Control. Signals, Syst.* **1989**, *2*, 303–314.
36. CS231n: Deep Learning for Computer Vision, Convolutional Neural Networks for Visual Cognition. Available online: `https://cs231n.github.io/neural-networks-1` (Accessed on 30 June 2023).
37. Wolpert, D.; Macready, W. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67âĂŞ82.
38. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133âĂŞ3181.
39. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.*, **2014**, *27*, 2672–2680.
40. Vaswani, A.; Shazeer, N.M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems 30 (NIPS 2017), 4-9 December 2017, Long Beach, California, USA.*, **2017**, *30*, 5998–6008.
41. The LLM Index. Available online: `https://sapling.ai/llm/index` (Accessed on 30 June 2023).
42. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; Dray, J.; Vo, S. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22 (May 2001). Available online: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf` (Accessed on 30 June 2023).
43. Lee, J.; Kang, H.; Lee, Y.; Choi, W.I.; Eom, J.; Deryabin, M.A.; Lee, E.; Lee, J.; Yoo, D.; Kim, Y.; No, J.Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network. *IEEE Access* **2021**, *10*, 30039–30054.
44. Levina, A.; Bolozovskii, R. Application of Neural Networks to Power Analysis. *Eng. Proc.* **2023**, *33*, 27. `https://doi.org/10.3390/engproc2023033027`.
45. Karras, D.A.; Zorkadis, V. Improving pseudo random bit sequence generation and evaluation for secure internet communication using neural network techniques. In Proceedings of the International Joint Conference on Neural Networks, IJCNN 2003, Portland, OR, USA, 20-24 July 2003; Volume 2, pp. 1367–1372.
46. Hu, X.; Zhao, Y. Research on Plaintext Restoration of AES Based on Neural Network. *Secur. Commun. Netw.* **2018**, *2018*, 6868506. `https://doi.org/10.1155/2018/6868506`.
47. Xiao, Y.; Hao, Q.; Yao, D.D. Neural Cryptanalysis: Metrics, Methodology, and Applications in CPS Ciphers. In Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC), 23 December 2019, Hangzhou, China, 2019; pp. 1–8.
48. Sun, L.; Gérault, D.; Benamira, A.; Peyrin, T. NeuroGIFT: Using a Machine Learning Based Sat Solver for Cryptanalysis. In Proceedings of the International Conference on Cyber Security Cryptography and Machine Learning 2020; Beer Sheva, Israel, July 2âĂŞ3, 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 62–84.
49. Albassal, A.; Wahdan, A. Neural network based cryptanalysis of a feistel type block cipher. In Proceedings of the International Conference on Electrical, Electronic and Computer Engineering (ICEEC '04), Cairo, Egypt, 2004; pp. 231–237.

50. Alani, M.M. Neuro-Cryptanalysis of DES and Triple-DES. In Proceedings of the International Conference on Neural Information Processing, 2012; Doha, Qatar, November 12-15, 2012; Springer: Cham, Switzerland, 2012; pp. 637–646.
51. Yee, L.P.; de Silva, L. Application of MultiLayer Perceptron Network as a one-way hash function. In Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290); Honolulu, Hawaii, 12-17 May 2002; Volume 2, pp. 1459–1462.
52. Arvandi, M.; Wu, S.; Sadeghian, A. On the use of recurrent neural networks to design symmetric ciphers. *IEEE Comput. Intell. Mag.*, **2008**, *3*, 42–53.
53. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking Cryptographic Implementations Using Deep Learning Techniques. IACR Cryptology ePrint Archive, Paper 2016/921. Available online : `https://eprint.iacr.org/2016/921` (Accessed on 4 August 2023).
54. Hou, B.; Li, Y.; Zhao, H.; Wu, B. Linear Attack on Round-Reduced DES Using Deep Learning. In Proceedings of the European Symposium on Research in Computer Security, Guildford, UK, September 14âĂŞ18, 2020.
55. Gomez, A.N.; Huang, S.; Zhang, I.; Li, B.M.; Osama, M.; Kaiser, L. Unsupervised Cipher Cracking Using Discrete GANs. *arXiv* **2018**, arXiv:1801.04883.
56. Gohr, A. Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18âĂŞ22, 2019.
57. Perusheska, M.G.; Trpceska, H.M.; Dimitrova, V. Deep Learning-Based Cryptanalysis of Different AES Modes of Operation. In *Advances in Information and Communication, Proceedings of the FICC 2022*; San Francisco, USA 3-4 March 2022; Arai, K., Ed.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2022; Volume 439.
58. Carbone, M.; Conin, V.; Cornélie, M.-A.; Dassance, F.; Dufresne, G.; Dumas, C.; Prouff, E.; Venelli, A. Deep Learning to Evaluate Secure RSA Implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, 132–161. `https://doi.org/10.13154/tches.v2019.i2.132-161`.
59. National Institute of Standards and Technology. Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES). Available online: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf` (Accessed on 30 June 2023).
60. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2002.
61. *NBS FIPS PUB 46*; Data Encryption Standard. National Bureau of Standards: Gaithersburg, MD, USA; U.S. Department of Commerce: Washington, DC, USA, 1977.
62. Grover, L.K. A fast quantum mechanical algorithm for database search. *arXiv* **1996**, arXiv:quant-ph/9605043v3.
63. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology—EUROCRYPT'93, Proceedings of the EUROCRYPT*; Lofthus, Norway, May 23âĂŞ27, 1993; Helleseth, T., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1993; Volume 765, pp. 386–397.
64. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, pp. 3–72.
65. Koeune, F., Standaert, FX. (2005). A Tutorial on Physical Security and Side-Channel Attacks. In: Aldini, A., Gorrieri, R., Martinelli, F. (eds) Foundations of Security Analysis and Design III. FOSAD FOSAD 2005 2004. Lecture Notes in Computer Science, vol 3655. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/11554578_3`
66. Piret, G., Quisquater, JJ. (2003). A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD . In: Walter, C.D., KoÃğ, ÃĞ.K., Paar, C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2003. CHES 2003. Lecture Notes in Computer Science, vol 2779. pp. 77–88. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/978-3-540-45238-6_7`
67. Knudsen, L.R. (1995). Truncated and higher order differentials. In: Preneel, B. (eds) Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol 1008. pp. 196–211. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/3-540-60590-8_16`
68. Wagner, D. (1999). The Boomerang Attack. In: Knudsen, L. (eds) Fast Software Encryption. FSE 1999. Lecture Notes in Computer Science, vol 1636. pp. 156–170. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/3-540-48519-8_12`

69. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L. (2018). Boomerang Connectivity Table: A New Cryptanalysis Tool. In: Nielsen, J., Rijmen, V. (eds) Advances in Cryptology âĂŞ EUROCRYPT 2018 . EUROCRYPT 2018. Lecture Notes in Computer Science(), vol 10821. pp. 683–714. Springer, Cham. https://doi.org/10.1007/978-3-319-78375-8_22

70. Courtois, N.T., Meier, W. (2003). Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (eds) Advances in Cryptology âĂŤ EUROCRYPT 2003. EUROCRYPT 2003. Lecture Notes in Computer Science, vol 2656. pp. 345–359. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39200-9_21

71. Meier, W., Pasalic, E., Carlet, C. (2004). Algebraic Attacks and Decomposition of Boolean Functions. In: Cachin, C., Camenisch, J.L. (eds) Advances in Cryptology - EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. pp. 474–491. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24676-3_28

72. Carlet, C. *Boolean Functions for Cryptography and Coding Theory*; Cambridge University Press: Cambridge, UK, 2021.

73. Braeken, A. Cryptographic Properties of Boolean Functions and S-Boxes. Ph.D. Thesis, Katholieke Universiteit Leuven, March 2006, Leuven, Belgium, 2006. ISBN 90-5682-649-2

74. Webster, A.F., Tavares, S.E. (1986). On the Design of S-Boxes. In: Williams, H.C. (eds) Advances in Cryptology âĂŤ CRYPTO âĂŹ85. Lecture Notes in Computer Science, vol 218. pp. 523–534. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39799-X_41

75. Biham, E.; Biryukov, A.; Shamir, A. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Advances in Cryptology—EUROCRYPT '99, Proceedings of the EUROCRYPT*; Prague, Czech Republic, May 2-6, 1999; Stern, J., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592.

76. Lim, J.; Ng, D.; Ng, R. SoK: Security Evaluation of SBox-Based Block Ciphers. Cryptology ePrint Archive, Paper 2022/1098. 2022. Available online: https://eprint.iacr.org/2022/1098 (Accessed on 3 july 2023).

77. Lenstra, A.K.; Lenstra, H.W., Jr. (Eds.) *The Development of the Number Field Sieve*; Lecture Notes in Mathematics; Springer: Berlin/Heidelberg, Germany, 1993; Volume 1554.

78. Pollard, J.M. A Monte Carlo method for factorization. *BIT Numer. Math.* **1975**, *15*, 331–334.

79. Lenstra, H.W., Jr. Factoring integers with elliptic curves. *Ann. Math.* **1987**, *126*, 649–673.

80. Pomerance, C. Analysis and Comparison of Some Integer Factoring Algorithms. In *Computational Methods in Number Theory, Part I*; Lenstra, H.W., Jr., Tijdeman, R., Eds.; Mathematical Centre Tracts; Mathematisch Centrum: Amsterdam, The Netherlands, 1982; Volume 154, pp. 89–139.

81. Boudot, F.; Gaudry, P.; Guillevic, A.; Heninger, N.; Thomé, E.; Zimmermann, P. Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment. Cryptology ePrint Archive, Paper 2020/697. 2020. Available online: https://eprint.iacr.org/2020/697 (Accessed on 6 July 2023).

82. Coppersmith, D. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptol.* **1997**, *10*, 233–260.

83. May, A. Using LLL-Reduction for Solving RSA and Factorization Problems. In *The LLL Algorithm*; Nguyen, P., Vallée, B., Eds.; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2009; pp. 315–348.

84. Wiener, M. Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inf. Theory* **1990**, *36*, 553–558.

85. Boneh, D.; Durfee, G. Cryptanalysis of RSA with private key d less than $N^{0.292}$. In Proceedings of the Advances in Cryptology—Eurocrypt'99; Prague, Czech Republic, May 2-6, 1999; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany 1999; Volume 1592, pp. 1–11.

86. Clavier, C.; Joye, M. Universal Exponentiation Algorithm A First Step towards Provable SPA-Resistance. In *Cryptographic Hardware and Embedded Systems—CHES 2001, Proceedings of the CHES*; Paris, France, May 14-16, 2001; Koç, Ç.K., Naccache, D., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2162.

87. Chevallier-Mames, B.; Ciet, M.; Joye, M. Low-cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Trans. Comput.* **2004**, *53*, 760–768.

88. Giraud, C. An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. *IEEE Trans. Comput.* **2006**, *55*, 1116–1120.
89. Moreno, C.; Hasan, M.A. SPA-Resistant Binary Exponentiation with Optimal Execution Time. *J. Cryptogr. Eng.* **2011**, *1*, 87–99.
90. Clavier, C.; Feix, B.; Gagnerot, G.; Roussellet, M.; Verneuil, V. Square Always Exponentiation. In *Progress in Cryptology—INDOCRYPT 2011, Proceedings of the INDOCRYPT*; Chennai, India, December 11-14, 2011; Bernstein, D.J., Chatterjee, S., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7107.
91. Bauer, A.; Jaulmes, E.; Lomné, V.; Prouff, E.; Roche, T. Side-Channel Attack against RSA Key Generation Algorithms. In *Cryptographic Hardware and Embedded Systems—CHES 2014, Proceedings of the CHES*; Busan, South Korea, September 23-26, 2014; Batina, L., Robshaw, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8731.
92. Boneh, D.; DeMillo, R.A.; Lipton, R.J. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology—EUROCRYPT'97, Proceedings of EUROCRYPT*; Konstanz, Germany, May 11âĂŞ15, 1997; Fumy, W., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1233.
93. Vigilant, D. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES; Washington, D.C., USA, August 10-13, 2008; Oswald, E.; Rohatgi, P., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5154, pp. 130–145.
94. Boscher, A.; Naciri, R.; Prouff, E. CRT RSA Algorithm Protected Against Fault Attacks. In *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems, Proceedings of the WISTP*; Heraklion, Crete, Greece, May 9-11, 2007; Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4462.
95. BSI—Technical Guideline, Cryptographic Mechanisms: Recommendations and Key Lengths, BSI TR-02102-1. 9 January 2023. Available online: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile` (Accessed on 7 July 2023).
96. Jansen, B.; Nakayama, K. Neural networks following a binary approach applied to the integer prime-factorization problem. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, 2005; 31 July-4 August 2005, Montreal, QC, Canada; Volume 4, pp. 2577–2582.
97. Murat, B.; Kadyrov, S.; Tabarek, R. Integer Prime Factorization with Deep Learning. *Adv. Interdiscip. Sci.* **2021**, *2*, 1–5.
98. de Weger, B. Cryptanalysis of RSA with small prime difference. *Appl. Algebra Eng. Commun. Comput.* **2002**, *13*, 17–28.
99. Lemke Oliver, R.J.; Soundararajan, K. Unexpected biases in the distribution of consecutive primes. *Proc. Natl. Acad. Sci. USA* **2016**, *113*, E4446–E4454.
100. Bernstein, D.J. How to Find the Smooth Parts of Integers. Available online: `http://cr.yp.to/factorization/smoothparts-20040510.pdf` (Accessed on 1 July 2023).
101. Nemec, M.; Sýs, M.; Svenda, P.; Klinec, D.; Matyás, V. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security 2017; Dallas Texas USA 30 October 2017- 3 November 2017.
102. May, A.; Ritzenhofen, M. Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint. In *Public Key Cryptography—PKC 2009, Proceedings of the PKC*; Irvine, CA, USA, March 18-20, 2009; Jarecki, S., Tsudik, G., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5443.
103. Faugère, J.-C.; Marinier, R.; Renault, G. Implicit factoring with shared most significant and middle bits. In Proceedings of the PKC; Paris, France, May 26-28, 2010; Nguyen, P.Q., Pointcheval, D., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6056, pp. 70–87.
104. Blömer, J.; May, A. A generalized Wiener attack on RSA. In Proceedings of the Public Key Cryptography—PKC; Singapore, March 1-4, 2004; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2947, pp. 1–13.

105. Nitaj, A. Another Generalization of Wiener's Attack on RSA. In *Progress in Cryptology—AFRICACRYPT 2008, Proceedings of the AFRICACRYPT*; Casablanca, Morocco, June 11-14, 2008; Vaudenay, S., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5023.
106. Ernst, M.; Jochemsz, E.; May, A.; de Weger, B. Partial Key Exposure Attacks on RSA up to Full Size Exponents. In *Advances in Cryptology—EUROCRYPT 2005, Proceedings of the EUROCRYPT*; Aarhus, Denmark, May 22-26, 2005; Cramer, R., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494.
107. Blömer, J.; May, A. New Partial Key Exposure Attacks on RSA. In *Advances in Cryptology—CRYPTO 2003, Proceedings of the CRYPTO*; Santa Barbara, California, USA, August 17-21, 2003; Boneh, D., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2729.
108. Takayasu, A., Kunihiro, N. (2014). Partial Key Exposure Attacks on RSA: Achieving the Boneh-Durfee Bound. In: Joux, A., Youssef, A. (eds) Selected Areas in Cryptography – SAC 2014. SAC 2014. Lecture Notes in Computer Science(), vol 8781. pp. 345–362 Springer, Cham. `https://doi.org/10.1007/978-3-319-13051-4_21`
109. Peikert, C. Public-key cryptosystems from the worst-case shortest vector problem. In Proceedings of the STOC 2009; Washington, DC, Sunday, May 31 - Tuesday, June 2009; pp. 333–342.
110. Brakerski, Z.; Langlois, A.; Peikert, C.; Regev, O.; Stehlé, D. Classical Hardness of Learning with Errors. *arXiv* **2013**, arXiv:1306.0281.