# Efficient File Encryption for Cloud Computing Using a Quantum Random Number Generator

A Doctoral Thesis by

**Anish Saini**

Submitted to the University of Hertfordshire in partial fulfilment of the requirement of the degree of

*Doctor of Philosophy*

Supervisors:

*Dr Athanasios Tsokanos*

*Dr Raimund Kirner*

July 2025

School of Physics, Engineering and Computer Science

University of Hertfordshire

## Abstract

This research studies the *high-speed cloud storage network* (HSCSN) components for efficient data transmission. Data is being generated and exchanged in HSCSN at an unprecedented pace in today's digital landscape. Using high-speed components ensures data can be transmitted quickly and reliably, minimizing latency and improving overall network performance. However, the security of personal data is a significant concern as information flows over the internet. A breach of data security can result in unauthorized access to sensitive information. HSCSN uses a cryptosystem to ensure data security between its component nodes. The cryptosystem has three building blocks: Key-schedule algorithm (KSA), Encryption, and Decryption. The randomness and optimization of the KSA components that generate the key for the other two blocks are directly related to data security and lead to security enhancement.

This study reviews literature to find a gap in research, including that on cryptosystems for HSCSN, attacks on cryptosystems, different random number generators (RNGs), RNG-based cryptosystems, cryptographic properties, the KSA, and the dynamic S-box (a component of the KSA) of the cryptosystems.

It focuses on creating CryptoQNRG, a new framework for KSA's cryptographic strength evaluation. Tests are used to explore cryptographic properties such as unpredictability, balance of bits, correlation, confusion, and diffusion in the subkeys generated by the RNG-based KSA. This research firstly evaluates the most common KSAs with different block ciphers and a significant outcome of the proposed framework is the distinction between strong and weak RNG-based KSAs.

Secondly it proposes the QuantumGS-box, a dynamic substitution box (S-box) for high-speed cloud-based storage encryption generated by a genetic algorithm and a quantum RNG. The proposed work generates the S-box values dynamically and by bit-shuffling with a genetic algorithm. An experimental evaluation of the proposed S-box method assesses several cryptographic criteria, including bit-independence criteria, speed, nonlinearity, differential and linear approximation probabilities, strict avalanche criteria, and balanced output. The results demonstrate that the QuantumGS-box adheres to the cryptography properties, enhancing robustness. The proposed S-box is resilient to differential and linear cryptoanalysis and assures nonlinearity. The characteristics of the proposed S-box are compared with recent S-boxes to validate its performance. Furthermore, the new dynamic S-box is balanced and generated at speed. These characteristics indicate that the designed S-box is a promising candidate for cloud-based storage encryption applications.

# Declarations and publications

***Declarations about previous submissions***

I declare that no part of this work has been submitted concurrently for another award of the University or any other awarding body or institution. This thesis contains significant and meaningful work that has not previously been successfully submitted for awards to any university or other institution.

*The research in this thesis leads to the following publications:*

Saini A, Tsokanos A, Kirner R. Quantum Randomness in Cryptography—A Survey of Cryptosystems, RNG-Based Ciphers, and QRNGs. *Information*. 2022; 13(8):358. https://doi.org/10.3390/info13080358.

*Q2, Emerging Sources Citation Index, Impact Factor - 2.9 (2024).*

Saini, A., Tsokanos, A. & Kirner, R. CryptoQNRG: a new framework for evaluation of cryptographic strength in quantum and pseudorandom number generation for key-scheduling algorithms. *J Supercomput* 79, 12219–12237 (2023). https://doi.org/10.1007/s11227-023-05115-4.

*Q2, Science Citation Index Expanded, Impact Factor - 2.7 (2024)*

Saini A, Tsokanos A, Kirner R. QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption. *Sci*. 2024; 6(4):86. https://doi.org/10.3390/sci6040086

*Q1, Emerging Sources Citation Index, Citescore - 5.2 (2024).*

# Acknowledgements

To begin with, I wish to express my gratitude to God for allowing me to use my knowledge and wisdom in this research.

I sincerely thank my supervisors, *Dr Athanasios Tsokanos* and *Dr Raimund Kirner* for their help and guidance during my study. They have supported and encouraged me throughout the research, provided invaluable feedback, and helped me refine my research project.

I am pleased to thank the entire University of Hertfordshire (UH) family, associated members of the Cyber Security Centre, and *Dr Raimund Kirner*, Director of the Centre at the School of Physics, Engineering and Computer Science. I am grateful for the guidance and support, which allowed me to develop my research skills and better understand the topic.

Furthermore, I sincerely thank the UH for sponsoring this research study. I would like to thank the *Doctoral College* and *Research Development Programme team* for their immense support at every point. The team provided me with guidance and advice throughout my PhD, helping me to navigate the various stages of my research project.

My sincere gratitude extends to my family for their support. I cannot express in words my gratitude to my parents. My father, *Dr Sukhpal Singh Saini*, and mother, *Mrs Pushpa Rani Saini*, supported me financially and morally throughout my academic journey and encouraged me to pursue my goals. My father has been a great supporter and adviser to me throughout my life, and has been always available for advice and guidance. Also, I would like to thank my partner, *Mrs Heena Saini*, my son, *Medhansh Kumar Saini*, and other family members for their love and patience.

# Table of Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| HSCSN | High-speed Cloud Storage Network |
| KSA | Key-schedule Algorithm |
| S-box | Substitution Box |
| RNG | Random Number Generator |
| TRNG | True Random Number Generator |
| PRNG | Pseudo Random Number Generator |
| QRNG | Quantum Random Number Generator |
| EC | Evolutionary Computation |
| GA | Genetic Algorithm |
| AES | Advance Encryption Standard |
| SET | S-box Evaluation Tool |

# Chapter 1. Introduction

## 1.1. Research context

The modern digital world is characterized by an interconnected network of high-speed cloud storage for storing vast amounts of sensitive and non-sensitive data. The information in the form of data flows from one device to another with the help of the internet. A *HSCSN* provides the benefits of social connectivity, global access, entertainment, education, etc. However, despite all the benefits of digital technology, data security is a significant concern as information flows through the internet. The data from these devices have to pass different nodes of the high-speed network.

### 1.1.1. Needs of cloud storage compared to local storage

The challenges of local data storage moved the research study towards cloud data storage.

*Data not available everywhere* [1][2][3] – sharing personal documents, such as medical, identity, or confidential files, is not always possible with local storage.

*Physical device is lost, data is lost* [4][5][6] – data stored on local devices (hard disk, pen drive, etc.) is at risk of being lost if the device is physically damaged or lost.

*Cost-trade 0ff* [7][8] – in terms of content sharing or data storage for small businesses, cloud storage offers an efficient and effective way to store data in order to share it or make it accessible.

*Internal Threats* [9][10] – the storage devices at the company pose the risk of internal threats with social engineering attacks.

Cloud computing has advantages due to the following [11]:

• Software-as-a Service: Clients/Users can use applications that are accessible via various client platforms (such as web browsers). The consumer does not control the infrastructure (the network, servers, operating systems, storage) required for this.

• Platform-as-a-Service: Cloud providers provide a platform for developers worldwide to deploy applications on a cloud infrastructure, as well as offer integration and monitoring services for these applications.

• Infrastructure-as-a-Service (IaaS): Cloud providers manage the underlying infrastructure while the consumers have control over the computing resources, including some control of selected networking components (e.g., host- versus network-based firewall).

*Data storage* is an important part of IaaS, and this study favours data security over cloud storage. Bader Alouffi et al. [12] surveyed a systematic review of threats and mitigation strategies related to cloud computing security. Data security in various applications such as

smart homes, relational databases, and data tampering on cloud storage, are some of the major issues. Ishu Gupta et. al. reviewed techniques for data protection (cryptography, access control, differential privacy with machine learning, watermarking, and probability techniques) in cloud storage. Attar N et. al. [13] proposed an architecture for data security in cloud storage space. Ahmed Albugmi et al. [14] discussed the risk of data loss if not properly protected in the cloud. In addition to discussing the risks and security threats to data, they recommended encryption techniques, such as block and stream ciphers. Pan Yang et al. [4] provided a detailed analysis of the challenges and requirements of data security and privacy protection in cloud storage systems. They also summarized the data encryption technologies and protection methods. Cloud Security Alliance mapped the cloud model to a security model in which data security is a major concern. Cryptography provides the techniques in the security model to protect the data on the cloud [15].

In *cryptography* [16][17], cryptosystems are designed for secure communication in HSCSN and focus on services such as confidentiality to secure data. Cryptosystems have three main parts: the *KSA*, Encryption, and the Decryption algorithm. The KSA generates and expands [18] the required key for the other two parts. The sender and receiver perform the encryption (convert the plaintext into ciphertext) and decryption (convert ciphertext into plaintext), respectively, with the help of a key generated by the KSA.

The KSA generates the key with the *substitution box* (S-box) and the permutations for encryption and decryption. Weak keys generated by the KSA invite an attacker to learn the key and decrypt data unintentionally. The potential consequences of data being decrypted unintentionally can be severe. It can expose sensitive information, such as personal or financial data, which can be used for identity theft or other malicious purposes. Furthermore, it can compromise the integrity and confidentiality of the data, undermining trust in the system or organization that holds the data in cloud storage. This research focuses on the KSA of the cryptosystem and the S-box of the KSA. The S-box performs an essential operation of substitution in the KSA. The S-box operation and its different types are discussed in section 2.3.

Also, the *randomness* of the key makes it harder for hackers to identify it. Key identification is crucial for hackers because it allows them to gain unauthorized access to encrypted data. Making the key truly randomly generated by KSA makes it significantly more challenging for hackers to decipher and exploit the encryption, enhancing the system's security.

The randomness can be achieved by different random method generators. Adequate RNGs generate more random and unpredictable data to make keys resilient against a

cryptanalysis attack. Randomness is generated using an entropy source, which can be either pseudo, atmospheric noise-based, or quantum-based [19][20]. A *Pseudo Random Number Generator* (PRNG) is achieved by the software method and determined by an initial (seed) value. Shobhit Sinha et al. [21] compared various PRNGs based on their statistical randomness and cryptographic security, concluding that some PRNGs performed significantly worse than others. Atmospheric noise-based RNGs are classified as *True Random Number Generators* (TRNGs), and a portable USB-based TRNG [22][23][24] device generates a truly random number. However, the *Quantum Random Number Generator* (QRNG) [25] ensures high entropy using quantum states of light from a quantum origin. Quantis [26], a QRNG developed by IDQ, generates random numbers using a quantum process [27][28]. The device allows live verification of its operation and provides the highest level of entropy without requiring a post-processing function to increase its entropy rate. The device has a function to monitor continuously. If a failure is detected, the random bit stream is immediately disabled. The device also provides the full randomness from the first bit.

The evolutionary computation, on the other hand, optimizes problem solutions focusing the process of natural selection with quantum random bits for the cryptographic properties. The genetic algorithm (GA) [29] of evolutionary computation works by mimicking the process of natural selection. It begins with generating a population of potential solutions, followed by crossover and mutation processes. Through an iterative process, the algorithm selects the fittest individuals from each generation, allowing them to pass their genetic information to the next generation. Throughout this process, the GA seeks an optimal solution. The genetic representation can be an array of bits, binary, or parse trees. The genetic operators are the basis of the algorithm's design criteria: selection, crossover, and mutation with fitness function. The algorithm based on a GA first represents the genetic representation of bits and then the fitness function to evaluate the suitability for a cryptographic property.

*Random Number Generator* and evolutionary computations play a significant role in enhancing the security of the cryptosystems used in this scenario, especially QRNGs, along with the GA in this research area.

The following sections aim to provide motivation and a more in-depth understanding of these areas by developing research question.

## 1.2. Research motivations

There are four motivations the led to this research study

*High-speed cloud storage data security*

Data can be transmitted between devices and stored in the cloud in real time through high-speed networks. Cloud storage provides various features, such as scalability for efficient cost, data accessibility in storing personal data on the cloud, making it available anytime, anywhere, and operating on sustainable energy through renewable resources. However, data security is always challenging in terms of providing at the different layers of the network. Despite all the benefits of digital technology, data security is a significant concern as information flows through the internet. The data from these devices has to pass through different nodes of the high-speed network. Data security from the physical layer to the application layer fulfils the demand for security, which is the primary exploration of this research. The research focuses on providing security to the data as the user converts it into bits and is ready to send it to another layer of a high-speed network, finally storing the data in cloud storage. This data security leads to a focus on cryptosystems, the second motivation for this research.

*Cryptosystems* provide data security by changing plain or raw text into an unreadable format through the process known as encryption. Decryption reverses this process. The encryption process has the main component, a KSA, that generates the keys for the user to encrypt and decrypt the data. These keys are required to be so random so that they are not vulnerable to attacks. The components, such as permutation, S-box, mathematical functions, etc. of the algorithm are required to be as random as possible. This research focuses on the KSA and S-box with true randomness generated by QRNGs. This leads to the third motivation, the true randomness of the QRNG.

*Quantum Random Number Generator*, as compared to other RNGs such as PRNGs, circuit design-based random number generators (CDRNGs), etc., generate the true random number based on photons of the light particles. A PRNG generates pseudo-randomness because it is implemented by software based on mathematical algorithms and determined by an initial (seed) value. The non-deterministic randomness generated by light photons in QRNG devices leads these devices to generate true quantum random numbers. Using these quantum random numbers, with a focus on the KSA and S-box, leads this research to enhance the security of the data. Quantum randomness combined with the evolution algorithm to follow an optimized solution motivated this research towards the GA.

*Evolution computation – genetic algorithm*

The GA depends on a heuristic search that mimics natural selection and finds the optimal solution to a problem. The GA is a practical algorithm for the system's performance. This

research focuses on the binary level of the data and the various phases of GA that help the binary data find the required optimal solution. The phases of GA include initial population, evaluation, fitness function, selection, crossover, mutation, and termination. The dynamic S-box created with the GA led to an optimal solution, which was the final motivation for this research.

## 1.3. Research aim and objectives

*This study aims to provide data security and especially considers cloud storage for personal and sensitive information, where the sender and receiver are the same. Confidentiality should be in place from the start of the communication of data until the data is stored on the cloud via different layers of the high-speed network.*

In order to achieve this, the study has the following objectives:

1. To examine and synthesize the existing literature on HSCSN cryptosystems and their relevance to cybersecurity, establishing a theoretical and experimental foundation for the study.
2. To explore the essential components of the cryptosystem with RNGs and cryptographic parameters to identify areas for improvement.
3. To develop a framework for evaluating cryptographic properties of the KSA based on RNGs in order to develop a novel method for evaluating KSAs.
4. To create the existing static S-box of a symmetric cryptosystem as dynamic based on quantum randomness and evolution computation, aiming for a novel approach to the dynamic S-box.
5. To analyse the meaningful cryptographic parameters, to identify the improvement in the S-box, creating it dynamically with QRNG and GA.

## 1.4. Research question

Although the S-box is generally well suited to a symmetric cryptosystem for high-speed cloud storage, its static bits make it vulnerable to attack of the secret key using with enhanced cryptanalysis techniques.

Keeping this in mind, this thesis is motivated by the following research question:

***Is it possible to improve the cryptographic properties of the S-box by creating it dynamically based on a user-defined key and exploiting knowledge of its bits matrix using QRNG and evolutionary computation?***

This question is split into the following sub-questions:

1. What is an efficient way to make the S-box of a symmetric cryptosystem dynamic?
2. What are the meaningful cryptographic parameters that will identify the improvement in the S-box, so creating it dynamically in the KSA?
3. Is it possible to design a dynamic S-box based on quantum randomness and a GA that is still suitable for the cryptographic properties of the S-box?

This research will investigate an efficient way to make the existing static S-box of a symmetric cryptosystem dynamic. The cryptographic parameters identify the improvement in the S-box, creating it dynamically and using it for the KSA. The existing S-box is static, but different research studies make the S-box dynamic. However, there needs to be more focus on a dynamic S-box using QRNG and the GA. Furthermore, creating a dynamic S-box justifies the cryptographic properties that focus this research. Hence, this research focuses on creating the dynamic S-box and validating its cryptographic properties. Comparing various cryptographic properties of other research studies ensures validation of the research findings of the proposed work.

## 1.5. Researcher's contribution

Figure 1.1 illustrates the research roadmap and novelty in research. The research roadmap typically includes key components such as problem identification, literature review, research objectives, methodology, analysis, findings, and conclusion. Each of these components plays a crucial role in guiding the research process and ensuring a systematic approach to achieving the desired research outcomes.


*Figure 1.1 Research roadmap & novelty*

## 1.6. Structure of the thesis

The thesis has nine chapters. A brief outline of the chapters is given below.

***Chapter 1. Introduction*** presents an overview of the research area, followed by an outline of the leading research question, research information, and a research roadmap. The research question aims to address the gap in existing knowledge and contribute to the field by providing new insights and solutions. It also aims to delve deeper into the research area and provide a comprehensive understanding of the topic by identifying the main research question. The roadmap of research serves as a guide to ensure a logical progression through the study.

***Chapter 2. Background*** talks about the critical areas of the research and their concepts, focusing on their details. This section focuses on the HSNCS model and its cryptosystems. It discusses the details of the KSA, S-box, QRNG and different stages of the GAs. These areas are significant because they provide the context and foundation for understanding research.

***Chapter 3. Related Work*** reviews the research work by different researchers in cryptosystems for the HSCNS, attacks related to cryptosystems, RNGs, the KSA, and the dynamic S-box domain. The related work by different researchers in the KSA and the dynamic S-box domain reveals various findings. While some researchers emphasize the importance of adaptability and flexibility in S-box design, others focus on the trade-off between security and efficiency. As a whole, these studies contribute to a better understanding of the field's challenges and opportunities. *(Published Paper I and part of Published Paper II & III)*

***Chapter 4. Research Methodology*** discusses the research process and methodology, giving an outline of the steps that were followed in this study. It includes information about data collection methods, analysis methods, and tools and frameworks to ensure validity and reliability.

***Chapter 5. CryptoQNRG: A new Framework for KSA's Cryptographic Strength Evaluation*** proposes a new framework consisting of cryptographic strength evaluation criteria for RNG-based KSAs. *(Published Paper II)*

***Chapter 6. QuantumGS-box – A key-dependent GA and QRNG-based S-box for high-speed cloud-based storage encryption*** proposes a dynamic S-box for high-speed cloud-based storage encryption generated by a GA and a QRNG. *(Published Paper III)*

***Chapter 7. Experimental Evaluation*** *focuses on result analysis of the proposed work.*

***CryptoQNRG*** evaluates the most common KSAs with different block ciphers. A significant outcome of the proposed framework is the distinction between strong and weak RNG-based KSAs. *(Published Paper II)*

***QuantumGS-box*** analyses the proposed dynamic S-box method using several cryptographic criteria, including bit-independence criteria, speed, nonlinearity, differential and linear approximation probabilities, strict avalanche criteria and balanced output. *(Published Paper III)*

***Chapter 8. Conclusion*** concludes the research question and recommends future research. In addition to summarizing the research question and recommending future research, the conclusion highlights the main findings of the research. It discusses the key insights and implications discovered throughout the study and provides a comprehensive overview of the research outcomes. *(Part of Published Paper II & III)*

***Chapter 9.*** **Future work** focuses on future work in designing and developing a dynamic QuantumGS-box for encryption in Raspberry Pi communication for IoT-based smart home system.

# Chapter 2. Background

## 2.1. High-speed cloud storage network

A HSCSN [11][30] transmits data over the network and channel with increased speed, reliability, security, and strength. Figure 2.1 shows the scenario of cloud storage through a high-speed network [31]. The data travels from different nodes and channels from the user to the cloud. The cloud storage service provider stores the data in the public or private cloud. The public cloud refers to data accessible outside the organization, unlike private cloud storage. Data transmission over a network requires rapid processing and poses security concerns to the systems. Information can be transmitted via one node, multiple devices, or a series of devices linked by a high-speed network. High-speed transmission protocols need to protect the data before transmission.

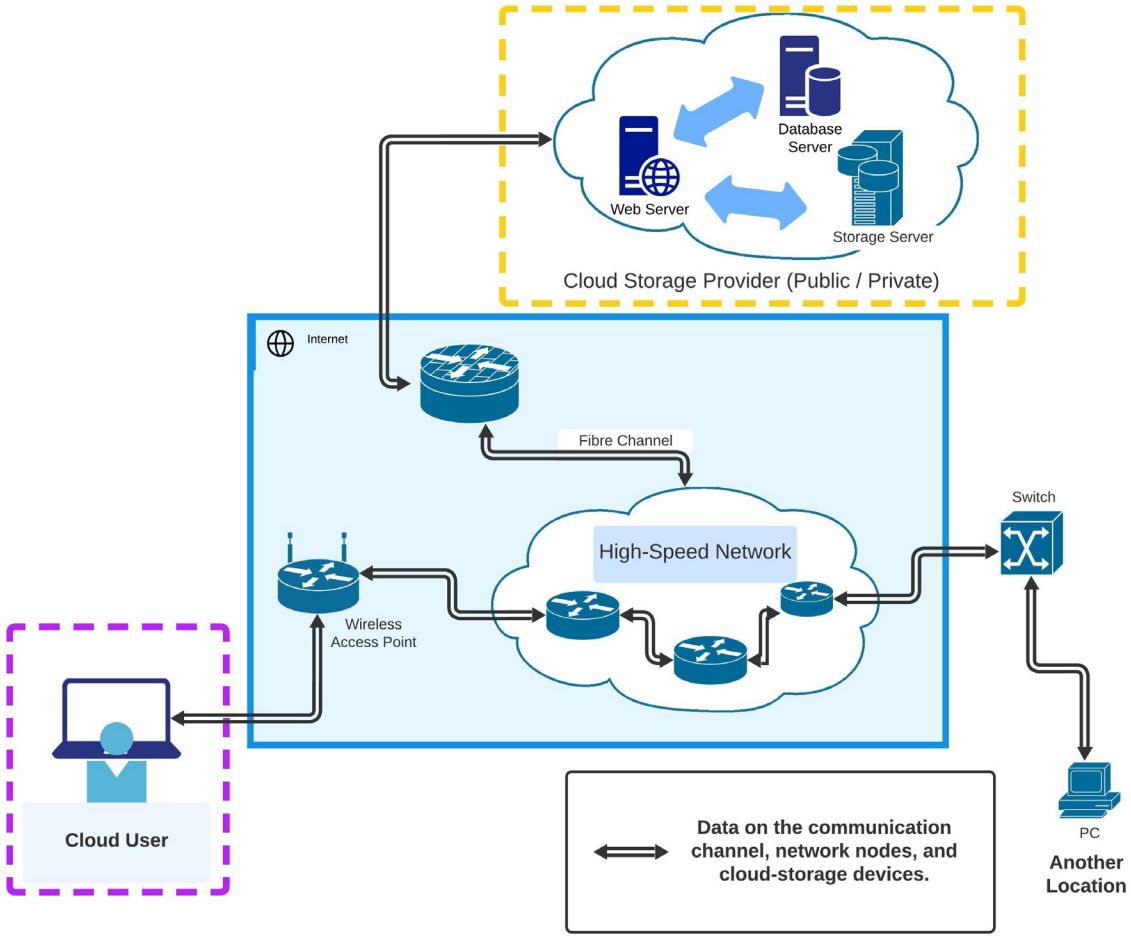



*Figure 2.1 Cloud-based storage through a high-speed network*

Data transferred from one branch to another or over the internet must be encrypted. Data will be securely transmitted and provided with a high-quality service by incorporating

cryptography that fulfils the requirement of secure transmission. In order to ensure secure data transmission over a network protocol, cryptography uses KSAs [32] and encryption[33].

The KSA includes methods of generating a random key with the substitution and permutation concept. The substitution replaces the specific bits with other defined bits, and a S-box is used to implement substitution. The permutation includes the shuffling of bits with particular operations. The S-box does the substitution in both the KSA and encryption.

The following sections focus on the areas used in this research: Advanced Encryption Standard (AES) –a cryptosystem for HSCSN, QRNG and evolutionary computation for cryptosystems – the GA.

## 2.2. Advanced Encryption Standard – a cryptosystem for HSCSN

The AES provides the cryptographic properties to secure data over the HSCSN. It is based on symmetric cryptosystems that use a private key for encryption and decryption. The key generation process generates a 128-bit or 256-bit key. Encryption is based on a block cipher and a substitution and permutation network.

Figure 2.2 [16] shows the AES structure. It displays the encryption and decryption process, which corresponds to key expansion. The 16-byte key expansion has 44-word from $w_0$ to $w_{43}$. The 16-byte plaintext is converted into 16-byte ciphertext by encryption and the opposite by decryption. The proposed research focus on $w_0$ to $w_{43}$ with QRNG bits and the function generator.

### 2.2.1. Key-schedule algorithm

Figure 2.3 [16] shows the existing state of the art of key generation and expansion. The AES key-schedule includes AES key generation and expansion. In AES 128, a random key is generated by 16 bytes, constituting $W_0$, $W_1$, $W_2$, and $W_3$. The key expansion will expand these 4 words to 44 words with g function and XOR operation. The total number of words required depends on the number of round keys. AES 128, 192, 256 requires 11, 13, 15 round keys, respectively. The total number of words needed is [$W_0$, $W_1$, $W_2$…….$W_{4R-1}$], where R is the number of round keys required.

*The g function*: Input to this g function is a W word that splits into $B_0$, $B_1$, $B_2$, and $B_3$. There are three steps involved in the g function.

One-byte Left Circular Shift – the function does the one-byte left circular shift and produces the output $B_1$, $B_2$, $B_3$ and $B_0$.

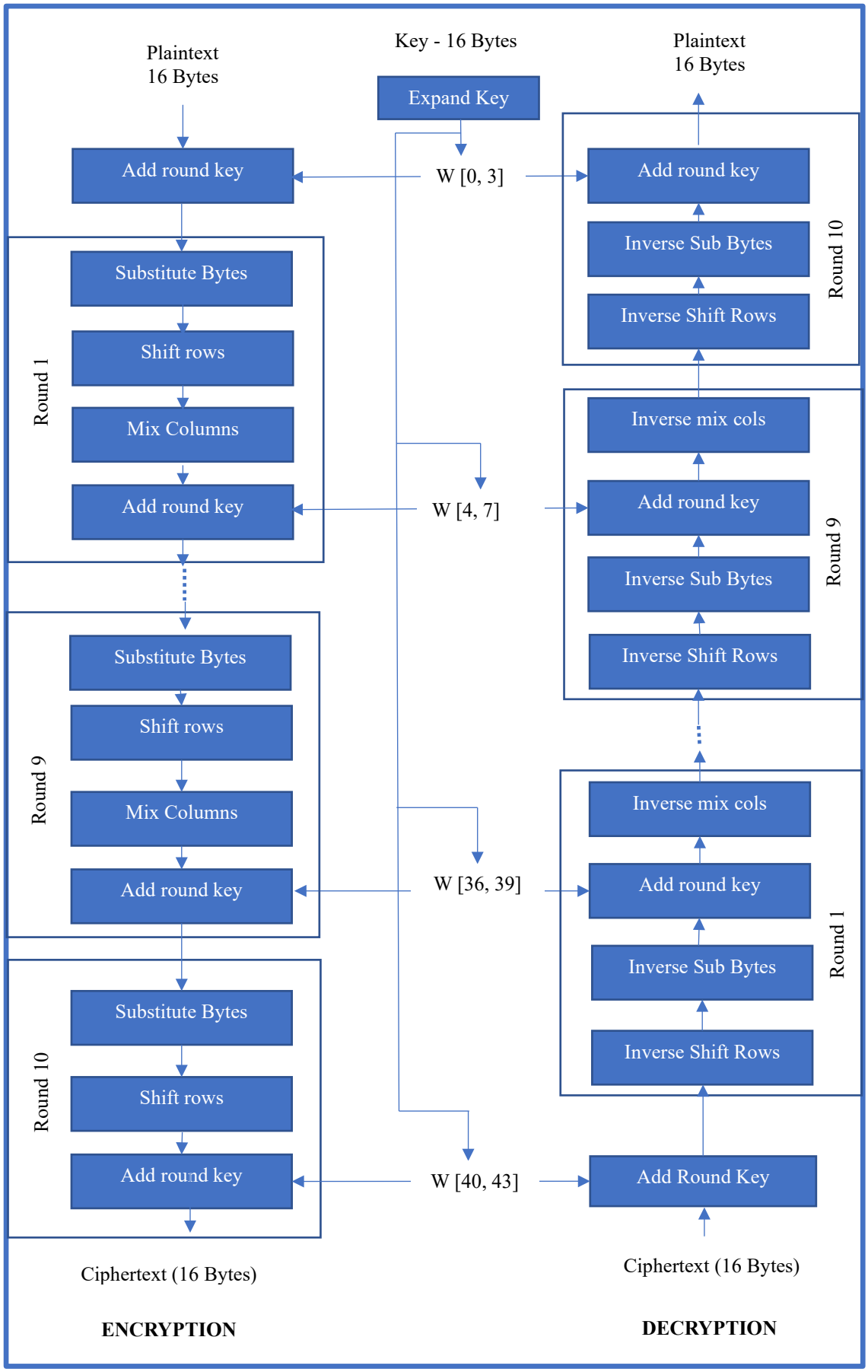



*Figure 2.2 AES structure*

Sub-bytes – the above $B_1$, $B_2$, $B_3$ and B0 use the S-box for creating the new sequence of $B_1$, $B_2$, $B_3$ and $B_0$.

XOR the bytes with the R constant – at the last stage, the new $B_1$, $B_2$, $B_3$, and $B_0$ are XORed with the R constant to generate the following sequence of W.

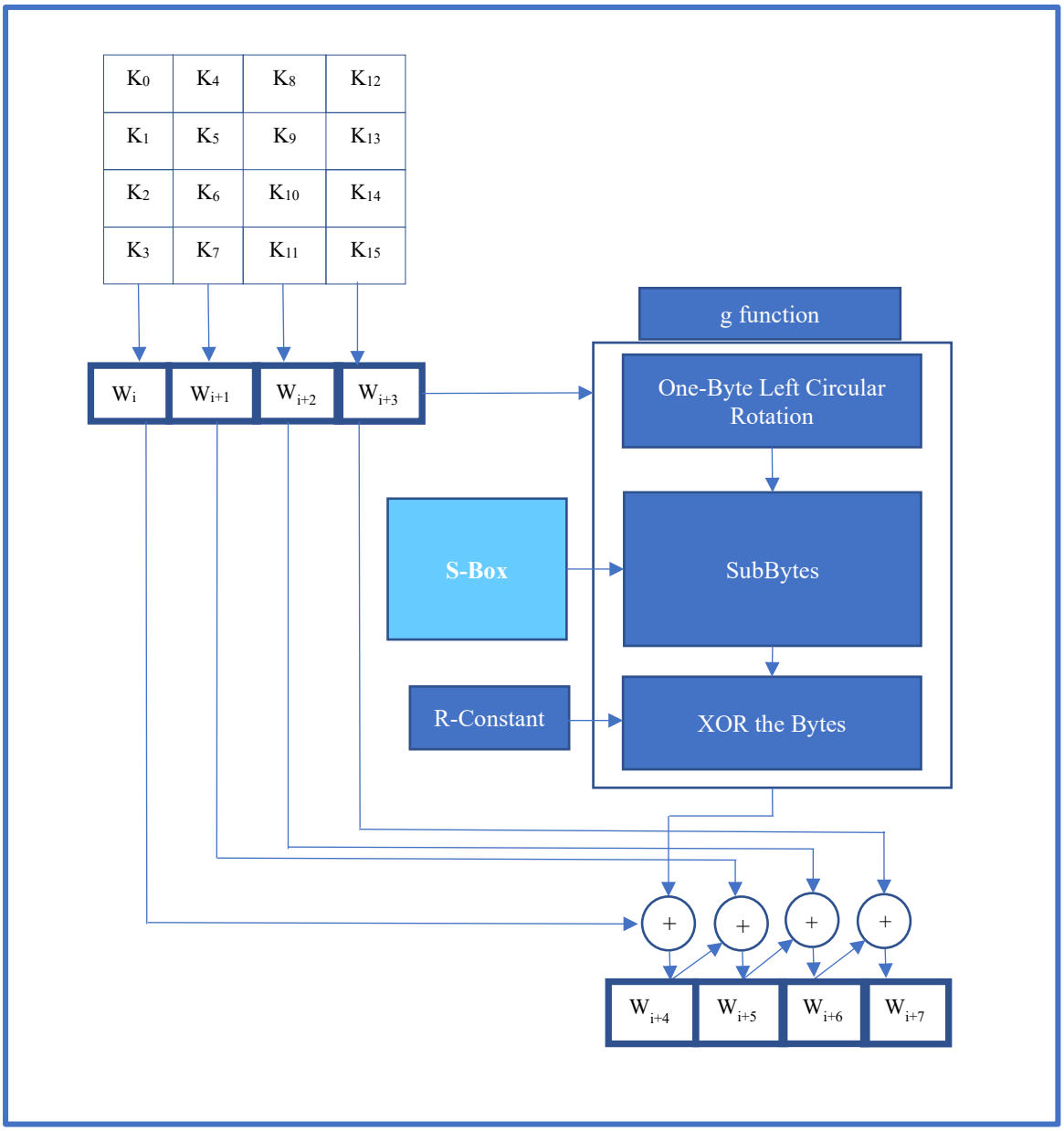



*Figure 2.3 Existing state of art – AES KSA – key generation & expansion*

### 2.2.2. Role of S-box

Cryptographic properties can be obtained from S-boxes. They introduce complexity and make it harder for attackers to analyse and predict the relationship between input and output. This

helps to prevent unauthorized access and ensures the confidentiality and integrity of the encrypted data.

***Confusion*** of AES is a relationship between ciphertext and key. Every bit of the ciphertext must depend on several parts of the AES key, obscuring the connections between the two [34]. S-box design cryptographic properties are based on the performance analysis of confusion.

***Nonlinearity***. An S-box is strong if it has the Boolean function with high nonlinearity [34] value. The nonlinearity Nf of a Boolean function f(x) is

$$N_f = 2^{n-1}\left(1 - 2^{-n}\max|S_f(\omega)|\right) \quad 2\text{-}1$$

Where $f_x\ if\ defined\ as$

$$S_f(\omega) = \sum_{\omega \in GF(2^n)}^{n} (-1)^{f(x)+x.\omega} \quad 2\text{-}2$$

Where $\omega \in GF(2^n)\ and\ x.\omega$ represents the dot product of x and $\omega$

***Balanced output***. An S-box with n input bits and m output bits, m ≤ n, is balanced if each output occurs 2n−m times. For the S-box to be balanced [35], it should have the same number of 0s and 1s.

***Bijectivity Property***. An S-box of n x n size is said to be bijective [36] if it has all possible output values from interval [0,2n-1]. If $f_i(1 \leq\ i\ \leq\ n)$ is a Boolean function of an S-box, it satisfies

$$\text{wt}(\textstyle\sum_{i=1}^{n} a_i f_i) = 2^{n-1} \quad 2\text{-}3$$

where $a_i \in \{0,1\}, (a_1 a_2 \ldots\ldots a_n) \neq (0,0 \ldots..0)$and wt(.) is the hamming weight, which indicates the number of 1s in a given vector. $f_i$ is to be balanced between 0 and 1.

***Strict Avalanche Criteria***. If a Boolean function satisfies the Strict Avalanche Criteria (SAC) [35], half of the output bits should change when there is a change in one bit. Any change in the input vector will significantly change the output vector with a probability of ½.

**Randomness**. Randomness in the key is crucial for ensuring the security of encryption algorithms. By incorporating randomness, the key becomes unpredictable and resistant to attempts at reverse engineering or brute force attacks. When the key is generated using a high level of randomness, it becomes challenging for adversaries to predict or guess the key, making it more resistant to attacks.

**Linear Approximation Probability (LAP)**. S-boxes are evaluated against linear cryptanalysis using the LAP. Linear cryptanalysis is a powerful technique to break cryptographic algorithms by exploiting linear relationships between the input and output. S-boxes can be assessed by examining their linearity through the LAP to determine their capacity to resist linear cryptanalysis and whether they suit cryptographic applications.

**Differential Approximation Probability**. Differential approximation probability (DP) is an effective way of assessing S-box resistance to differential attacks. Differential attacks are a type of cryptanalysis technique whereby the attacker observes the differences in the input and output of a cryptographic function. By analysing these differences, the attacker can gain information about the internal structure of the function and potentially break its security. Therefore, assessing the resistance of the S-box, a critical component in a cryptographic algorithm, to differential attacks is crucial in ensuring the system's overall security. DP quantitatively measures this resistance, allowing for a more precise evaluation of the S-box's strength against such attacks.

## 2.3. Types of substitution boxes

The S-boxes are divided into two categories: static S-boxes and dynamic S-boxes. S-boxes are crucial in modern cryptography, particularly symmetric-key algorithms such as the AES. They introduce stringent cryptographic properties into the KSA and encryption process, making it more resistant to attack. Cryptographic systems can achieve higher security levels by carefully designing and manipulating S-boxes and protecting sensitive data from unauthorized access.

### 2.3.1. Static S-box

An AES S-box[37][38] having order (p x q) is a mapping function L= S(c), where $S = \{0,1\}^p \rightarrow \{0,1\}^q$, which is used to map p-bits input string c to q-bits output string L. The input is mapped with its multiplicative inverse in $GF(2^8)$.

I. The first step is nonlinear function defined $f(c)$

$$f(c) = \begin{cases} 0, if\ c = 0 \\ c^{-1}, if\ c \neq 0 \end{cases} \quad 2\text{-}4$$

The function $f(c)$ maps zero to zero, and, for a non-zero field element c, it maps the element to its multiplicative inverse $c^{-1}$ in $GF(2)^8$

II. The multiplicative inverse is then transformed using following affine transformation

$$L = Affine_8(c) \quad 2\text{-}5$$

$$\begin{bmatrix} L_7 \\ L_6 \\ L_5 \\ L_4 \\ L_3 \\ L_2 \\ L_1 \\ L_0 \end{bmatrix} = \begin{bmatrix} 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \\ 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \end{bmatrix} \times \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Figure 2.4 [39] shows the AES static S-box. It is 16 x 16 matrix of hexadecimal values that are used in KSA and encryption.

| | **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **0a** | **0b** | **0c** | **0d** | **0e** | **0f** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **00** | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| **10** | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| **20** | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| **30** | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| **40** | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| **50** | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| **60** | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| **70** | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| **80** | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| **90** | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| **a0** | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| **b0** | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| **c0** | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| **d0** | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| **e0** | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| **f0** | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 1 |

*Figure 2.4 AES static S-box*

There are several cryptographic properties that make the AES S-box unique, such as high nonlinearity, correlation immunity, algebraic immunity, and no fixed points or opposite fixed points. However, because of its static nature, it is susceptible to enhanced cryptoanalysis techniques [40][41].

### 2.3.2. Dynamic S-box

The dynamic substitution box is also a *nxn* matrix, and the values are not static. Different methods generate them during the data processing. Using a dynamic substitution box provides an added layer of security in the cryptosystem. The constantly changing values make it more difficult for attackers to decipher the encryption algorithms and patterns. This enhances the overall resilience and effectiveness of the system in protecting sensitive information. Unlike a static substitution box with fixed values, the dynamic substitution box can adapt and change its

values based on the specific needs and requirements of the data processing. This flexibility allows for a more effective and customized approach to data encryption and decryption, enhancing the overall security and efficiency of the system. The constantly changing values of the dynamic substitution box make it highly resistant to attacks that rely on analysing patterns and algorithms. By generating new values during data processing, the system ensures that even if an attacker manages to decipher one set of values, those values would no longer be valid for future encryption and decryption processes. This adds an extra layer of complexity and unpredictability, making it significantly more challenging for attackers to compromise the security of the cryptosystem.

Chapter 3, the literature review, will assess various researchers' methods of generating dynamic S-boxes. The effectiveness of S-box generation methods is typically evaluated based on criteria such as nonlinearity, differential uniformity, algebraic complexity, and resistance against linear and differential cryptanalysis. These criteria help researchers determine the security and strength of the generated S-boxes for cryptographic applications.

## 2.4. Quantum random number generator

The researchers of randomness technology took a significant step forward with the advent of QRNGs, in which the properties of light particles determine randomness. Quantum randomness refers to the unpredictable and inherently random nature of specific physical processes at the quantum level.

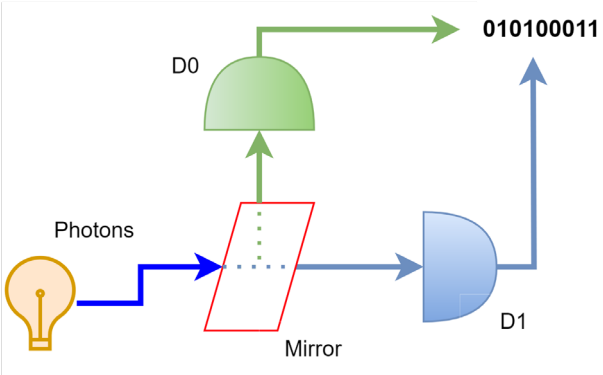



*Figure 2.5 Random number generation using an optical process as a QRNG*

In the context of a QRNG, the randomness of the generated numbers is not based on any pre-existing pattern or algorithm but rather on the unpredictable behaviour of light particles known as photons. This makes QRNGs highly secure and suitable for applications such as encryption and cryptography. Figure 2.5 [26] shows the random number generation process

using an optical device and generating quantum random numbers. Light comprises photons, the smallest particles that cannot be further divided. With the use of a single-photon detector as a mirror, if the light is transmitted, it will count as "1", and if it is reflected, it will count as "0". The randomness of the sequence generated by this phenomenon (at the moment of detection) is truly random because the state of the information path's source (light, photon, glass, or the detectors) is impossible to identify.


*Figure 2.6 Quantis QRNG PCIe & USB*

Figure 2.6 illustrates the IDQ Quantis and outlines the following properties when generating quantum random numbers.

*Unbiasing of random numbers*

*Status monitoring*

*Separating Quantum noise from classical noise*

*Auto calibration*

There are various studies related to QRNGs focusing on different features. Lunghi et al. [42] proposed a protocol for self-testing QRNG, in which the user can monitor the entropy in real time. Hesong Xu et al. [43] proposed a QRNG using single-photon avalanche diodes (SPADs) that produces a quantum random number without any post-processing. Biasing is one of the critical features that researchers consider when generating a quantum number. Pooser et al. [44] used a tapered amplifier consisting of optical semiconductor devices and an array of random number registration techniques to create quantum-based random numbers. A photon arrival time selectively based high-quality bias-free QRNG was introduced by Jian-min Wang et al.[45].

Another aspect of quantum processes is the speed at which random numbers can be generated. Yu-Huai Li et al. [46] proposed QRNG with an uncharacterized laser and sunlight

that generates random numbers at 1 Mbps. Abellan et al. [47] proposed an ultra-fast QRNG accelerated phase diffusion in a pulsed laser diode observing 43 Gbps.

Miguel Herrero-Collantes et al. explained various types of quantum number generators based on different sources such as optical and non-optical QRNGs [25]. Xiongfeng Ma et al. categorized QRNGs into three groups: practical QRNGs (generate randomness at high speed and built on fully trusted and calibrated devices), self-testing QRNGs (randomness without trusting the actual implementation), and semi-self-testing QRNGs (a trade-off between the trustworthiness of the device and the RNG speed) [48]. Jinlu Liu et al. proposed a 117-Gbit random bit generation rate QRNG using min-entropy estimation and Toeplitz-hashing randomness extraction, based on the quantum phase fluctuation of a distributed feedback laser [49]. Meilana Siswanto et al. designed a photonic-based RNG to enhance the security of an Internet of Things (IoT) system comprised of optical components, analogue–digital electronic systems, and an asynchronous transmitter [50]. It utilized Verilog firmware to integrate the IoT system. Zhu Cao et al. proposed a source-independent QRNG in which output randomness could be certified even when the source was uncharacterized and untrusted.

Chapter 3, the literature review, will critically analyse RNGs and their various classifications (PRNGs, TRNGs, CDRNGs and QRNGs) including generation methods, properties, disadvantages and advantages. It will also focus on the properties (research objectives) of the QRNGs that various researchers focus on.

## 2.5. Evolutionary computation for cryptosystems – genetic algorithm

A GA [51] is a search heuristic inspired by the natural evolution of Charles Darwin's theory. The algorithm corresponds to the natural selection of selecting the fittest individuals based on fitness function for reproduction to produce offspring of the next generation. A GA requires the following two factors of a domain solution:

a genetic representation

a fitness function to evaluate suitability.

Initial population, fitness function, selection, crossover, mutation, and termination are the GA phases that support the optimum Boolean function required with cryptographic properties for S-box design.

Figure 2.7 shows the basic flow of the GA [52]. The initial population phase sets the foundation for the GA by creating a diverse set of candidate solutions. The fitness function phase evaluates the quality of each solution, allowing the algorithm to prioritize the fittest individuals. The

selection phase determines which individuals will be chosen as parents for reproduction, promoting the propagation of desirable traits.

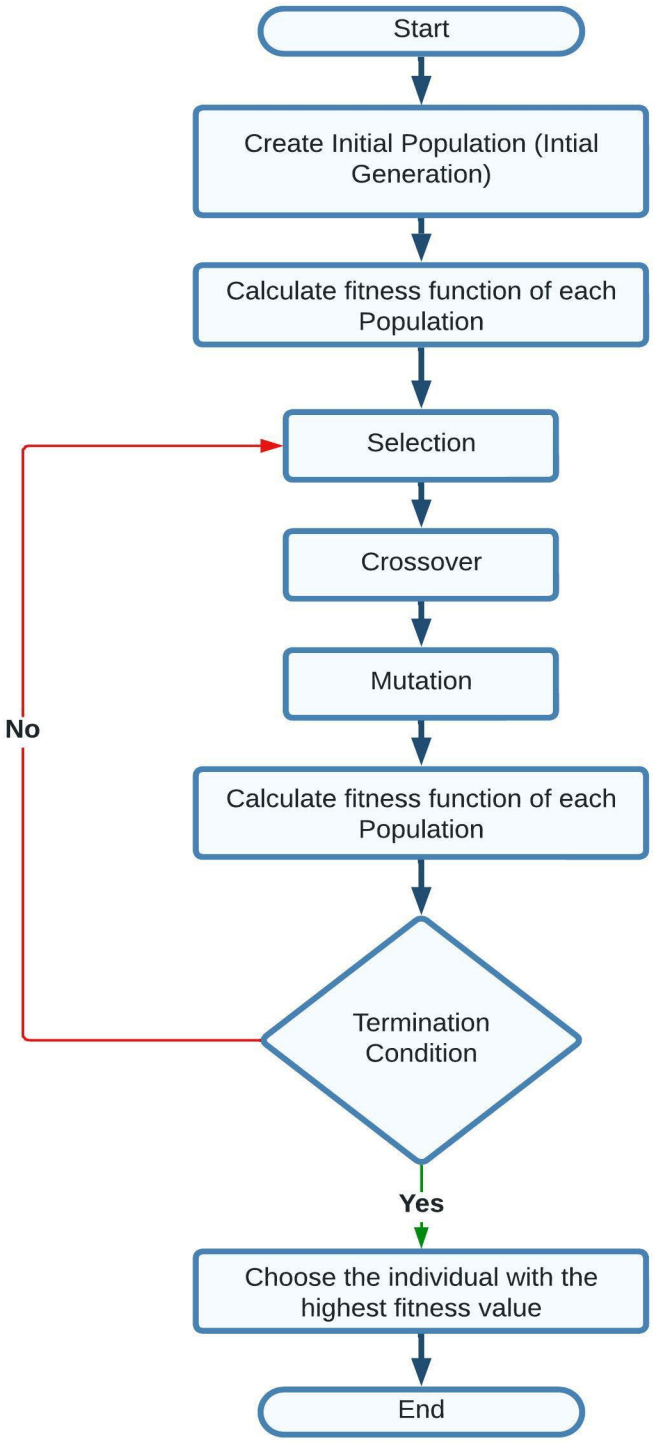



*Figure 2.7 Basic flow of a genetic algorithm*

The crossover phase combines the genetic information of selected parents to create offspring with potentially improved characteristics. The mutation phase introduces small, random changes to the offspring, ensuring the exploration of new solutions. Finally, the

termination phase stops the algorithm once a satisfactory solution is found, or a predefined number of generations is reached.

Sourabh Katoch et al. reviewed the advantages and disadvantages of different GAs such as Classical GAs, Binary-coded GAs, Real-coded GAs, Multi-objective GAs, Parallel GAs, Chaotic GAs, and Hybrid GAs [53]. They also discussed various operators used for different GAs. Yong Wang et al. proposed a novel method for constructing the S-box – transforming it into a Travelling Salesman Problem – and designed an S-box based on chaos and a GA [54]. Another study by Yong Wang et al. proposed a novel GA to construct bijective S-boxes with high nonlinearity, taking the nonlinearity of the S-box as the optimization objective [55].

## 2.6. Summary and conclusion

This chapter outlined the background of the research focus area of HSCSN, where Alice is both sender and receiver. The AES cryptosystem focused on key-schedule expansion and generation, and highlighted the importance of the S-box and its dynamic nature. The chapter also discussed QRNGs and GAs. The next chapter is a literature review of the research area, which it explores in depth.

# Chapter 3. Related work

## 3.1. Literature review – Cryptosystems for HSCSN

In cryptology, a cryptography algorithm is the art of creating codes or algorithms that turn plaintext into ciphertext and ciphertext into the original text. It establishes secure communication between two entities in the public domain, of HSCSN, where unauthorized users and information hackers are present.

The main difference between the cryptography algorithm is the relationship between the encryption and the decryption key. Logically, in any cryptographic system known as a cryptosystem, keys generated and expanded by the KSA play an essential role in the cryptographic process.

A private key is used for both encryption and decryption, and algorithms are classified as *Symmetric-Key cryptography*, whereas *Asymmetric-Key cryptography* uses a pair of public and private keys for encryption and decryption or signing and verification. Figures 3.1 and 3.2 visualize Symmetric-Key cryptography and Asymmetric-Key cryptography, respectively.

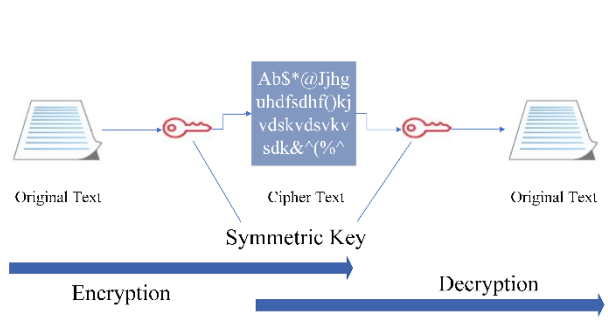



*Figure 3.1 Symmetric-Key cryptography*

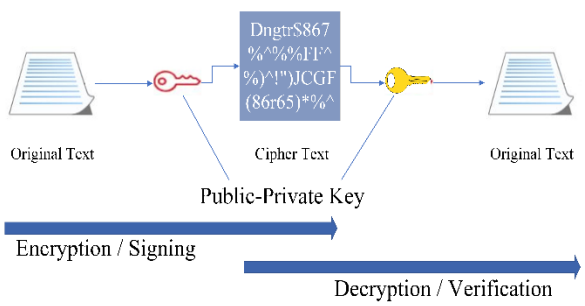



*Figure 3.2 Asymmetric-Key cryptography*

There are two types of *Symmetric-Key cryptography*, block cipher and stream cipher.

Table 3.1 summarizes the Symmetric-Key cryptography categorized into block cipher and stream ciphers.

*Block Cipher:* A block cipher encrypts or decrypts a block of data. Encryption transforms plaintext into an equal length of ciphertext block, whereas decryption does the reverse process on the same block of ciphertext . It primarily uses either of the following two network structures:

*Feistel (F)-Network:* In F Network, the input data splits into two parts, the left and the right; the right part remain unchanged and the left part goes through the operation with the key.

After processing several rounds of operation designed by the algorithm, the combination of the left and right parts constitutes the ciphertext.

*Substitution-Permutation (SP)-Network*: In SP Network, the substitution layer consists of an S-box and operates on short data segments. Due to the layer's highly nonlinear property, it creates confusion in the internal ciphertext. The permutation layer diffuses the effect of substitution across the entire block.

*Table 3.1 List of symmetric-key cryptography systems.*

*Symmetric-Key cryptography*

**(A secret key generated by KSA for Encryption & Decryption)**

| Block Ciphers | | Stream Ciphers | |
|---|---|---|---|
| F Network | SP Network | AX | LFSR |
| 1900's | 1900's | 1900's | 1900's |
| — DES [56]<br>— 3 DES [56]<br>— IDEA [63]<br>— Blowfish [64]<br>— Tea [70]<br>— CAST-128 [71]<br>— XTEA [70]<br>— RC2 [73]<br>— RC6 [75] | — Safer-K [57]<br>— 3-Way [58]<br>— Serpent [59]<br>— AES [39] | — RC4 [60] | — A5/1 [61]<br>— A5/2 [62] |
| | | 2000's | |
| | | — Rabbit [65]<br>— SOBER-128 [66]<br>— QUAD [67]<br>— Trivium [68]<br>— Salsa20 [69] | 2000's<br>— SNOW 2 [72]<br>— Crypto-1 [74] |
| — CAST-256 [71]<br>— MARS [76]<br>— Twofish [77] | | LFSR+NFSR | PF |
| 2000's | | 2000's | 1900's |
| — Camellia, 2000 [81]<br>— Threefish [83]<br>— GOST [84] | | — HC-256 [78]<br>— Grain [79]<br>— MICKEY [80] | — ISAAC [60]<br>— SEAL [82]<br>2000's<br>— Phelix [85] |

*Stream cipher:* A stream cipher performs encryption or decryption on a digital data stream of 1 byte or 1 bit. When data is to be transmitted over a communication channel or through a web browser, this can be helpful. It depends on different structures broadly categorized as follows: Arithmetic and XOR(AX), Linear-Feedback Shift Register (LFSR), combination of Linear-

Feedback Shift Register and Nonlinear-Feedback Shift Register (LFSR+NFSR), and Pseudorandom Function (PF).

*Arithmetic and XOR(AX):* Arithmetic operations include all bitwise operations (XOR, NAND, OR, Modulo operator) for cryptographic functions.

*Linear-Feedback Shift Register:* LFSR is a shift register where the input bit is a previous state's linear function (exclusive-or (XOR)). The input is affected by previous stage values, so it operates as a feedback mechanism.

*Nonlinear-Feedback Shift Register: NFSR is a shift register where the input bit is a previous state's nonlinear function. The stream cipher also uses the LFSR+NFSR structure, which combines the LFSR and the NFSR.*

*Pseudorandom Function:* These functions are based on pseudo random numbers generated by PRNG.

Equally important is *Asymmetric-Key cryptography,* which is based on a pair of keys, and categorized into two approaches: Public-Key cryptography and Digital Signature.

Table 3.2 shows the Asymmetric-Key approaches used by Public-Key cryptography and Digital Signatures

*Public-Key cryptography:* A public-key cryptography system uses a pair of public and private keys. The KSA generates both the keys, a public key for the encryption and a private key for the decryption. Encryption is performed by the receiver's public key, generating the ciphertext, and the ciphertext can only be decrypted with the sender's private key. *Factorization and Diffie–Hellman's (DH) key exchange* are the two main techniques of Public-Key cryptography.

*Factorization:* In factorization, a composite number, part of mathematical computation, is transformed into a product of smaller integers, whereas when the integers are primes, the process is known as prime factorization.

*Diffie–Hellman key exchange*: DH is a cryptographic set of rules for exchanging the private key over an unsecured channel. The KSA of both sender and receiver generates the public and private key pair. Both the parties share their public key. Once they get the public key, they calculate their secret or private key and use it for sending data securely.

*Post-Quantum Public-Key cryptography:* The cryptanalysis of classical ciphers led researchers to develop quantum and classical computer-resistant cryptosystems. Moreover, these systems can communicate with an existing communication protocol, allowing their practical implementation for applications. The National Institute of Standards and Technology (NIST) [86] announced the four post-quantum public-key encryption and key-establishment

algorithms in round 3 compared to 17 algorithms in round 2. NIST has started the process of standardizing these algorithms.

*Table 3.2 List of Asymmetric-Key cryptography systems*

*Asymmetric-Key cryptography*

**(A public and a private key generated by KSA for Encryption/Signing & Decryption/Verification)**

| Public-Key cryptography | | Digital Signature | |
|---|---|---|---|
| Factorization | DH-Key Exchange | DSS | ECC |
| | | | 2000's |
| | | | — ECIES, [90] |
| — RSA, [87] 1978 | — ElGamal, [88] 1985 | — DSA, [89] 1994 | — ECDSA, [90] |
| | | | — EdDSA, [91] |
| Post-Quantum Public-Key cryptography [86] | | Post-Quantum Digital Signature [86] | |
| — BIKE | — Classic McEliece | — CRYSTALS-DILITHIUM | |
| — CRYSTALS-KYBER | — FrodoKEM | — GeMSS | |
| — HQC | — LAC | — MQDSS | |
| — LEDAcrypt | — NewHope | — qTESLA | |
| — NTRU | — NTRU Prime | — SPHINCS+ | |
| — NTS-KEM | — ROLLO | — FALCON | |
| — SIKE | — Three Bears | — LUOV | |
| — RQC | — SABER | — Picnic | |
| — Round5 | | — Rainbow | |

*Digital Signature:* The digital signature is an electronic signature in which the sender signs or encrypts the document with a private key. The receiver will decrypt or verify that document with the sender's public key. It can be incorporated into cryptography using the Digital Signature Standard or Elliptic Curve Cryptography.

Digital Signature Standard (DSS): DSS is a standard established by NIST to generate digital signatures.

Elliptic Curve Cryptography (ECC): A cryptographic algorithm based on elliptic curves generates smaller key sizes while providing the same level of security as those without them.

*Post-Quantum Digital Signature:* NIST announced the three post-quantum digital signature algorithms in round 3 compared to nine algorithms in round 2. These are another part

of post-quantum cryptography. NIST has started the process of standardizing these algorithms also.

### 3.1.1. Literature review – cryptographic primitive function properties

The primitives of cryptography are affected by the objectives and operations required to perform their functions. Randomness is the fundamental function of the cryptographic primitive. Programming languages have methods for generating random numbers based on PRNGs. The software function determines the randomness of the PRNG, based on an initial seed value. Using Python, the *random()* [92] function generates a random number, whereas SecureRandom [93] uses SHA1PRNG as a cryptographically secure PRNG.

However, some researchers offered alternative approaches for creating randomness in the cryptographic key. The random key suggested by [94] encrypted the plaintext with the use of a laser stream. [95] and introduced a new version of AES using a new key generation process with random keys [6].

Other factors, such as resilience [96] and correlation-immune [97] characteristics, are also crucial to the development of cryptography primitives. A low-autocorrelation S-box of the block cipher with the Nash equilibrium-based multi-objective is proposed in [98]. An autocorrelation [99] function can also be represented when the symmetric function has a fixed hamming weight. Hamming weight was also shown to be more effective in genetic programming [100]. [101] showed a highly nonlinear resilient Boolean function. Multi-output resilient functions have been achieved by using the Walsh spectrum property [102]. Vector Boolean function design are also proposed for block and stream cipher to associate resiliency. A nonlinear resilient through vector BF has been proposed in [103].

The algebraic immunity is another crucial function performed by the Boolean function with balanced bits [104]. Furthermore, the rotation of some variables in the Boolean function has achieved algebraic immunity [105]. The Boolean function [106] can also be formulated to calculate nonlinearity along with algebraic degree [107]. The nonlinearity of different Boolean functions has been compared in [108]. Apart from BF, Chaos-Based Rotational Matrices [109] were proposed to achieve nonlinearity and dynamicity in the substitution box of an AES cipher.

Twofish [110] proposed using multiple-level key space to increase the complexity with no drastic change to other factors. Enhanced key-scheduling [111] led to efficient optimization and security in 3DES.

Of equal importance, the strict avalanche effect measures the probability and secures the cipher from attack. The balanced Boolean function also plays a vital role in the avalanche effect, as shown in [112]. A cost analysis of the GA to generate a balanced Boolean function is presented in [113].

### 3.1.2. Literature review – attacks on secured cryptosystems

The security of a Symmetric-Key or Asymmetric-Key depends on the vulnerability of the key to cryptographic attacks. Cryptanalysis analyses cryptosystems to look for weak points or opportunities for information leaks to access or find the key. Cryptanalysis is another part of cryptology that differentiates different cryptographic attacks. This research categorizes the cryptographic attacks into six categories: Differential Cryptanalysis, Linear Cryptoanalysis, Meet-in-the-middle (Mt-in-M), Side-Channel Attacks, Related-Key Attacks and Other attacks.

Table 3.3 shows the survey of cryptographic attacks on different ciphers with the key size of KSA.

*Differential Cryptanalysis (DC):* DC refers to a chosen plaintext attack, where the attacker can choose a plaintext and find the corresponding ciphertext in order to obtain the key. By computing the differences between the ciphertexts, the attacker can detect statistical patterns in their distribution. These differences constitute a differential attack.

*Linear Cryptoanalysis (LC):* LC finds affine approximations to the cipher's action based on linear equations. Depending on linear equations, it comes close to a plaintext-ciphertext pair and attempts to find the key.

*Meet-in-the-middle:* Mt-in-M attack is a known plaintext attack that tries to break the long chain of encryption blocks by halving what they are analysed to find the key more accurately. This accuracy cost depends on breaking the encryption chain into smaller parts requiring more storage. However, it is still more efficient than a brute force attack regarding time and computational complexity.

*Table 3.3 Cryptanalysis of cryptography ciphers*

| | | ***Cryptoanalysis*** | | | | | |
|---|---|---|---|---|---|---|---|
| ***Ciphers*** | ***KSA Key Size (Bits)*** | ***DC*** | ***LC*** | ***Mt-in-M*** | ***SCA*** | ***RKA*** | ***Others*** |
| DES | 56 | ✓ | ✓ | ✓ | | | Brute Force |
| 3 DES | 112 or 168 | | | | | | Sweet32 |

| *Ciphers* | *KSA Key Size (Bits)* | *Cryptoanalysis* | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***DC*** | ***LC*** | ***Mt-in-M*** | ***SCA*** | ***RKA*** | ***Others*** |
| IDEA | 128 | | | Bicliques Attack | | | |
| Blowfish | 32–448 | | | | | | Birthday Attack |
| Tea | 128 | ✓ | | | | ✓ | |
| GOST | 256 | ✓ | | Reflection | | ✓ | |
| CAST 128 | 40 to 128 | ✓ | | | | | |
| XTEA | 128 | ✓ | | ✓ | | ✓ | |
| RC2 | 1–128 Bytes | | | | | ✓ | |
| RC6 | 128, 192, 256 | | | | | | Statistics Attack |
| CAST 256 | 128, 160, 192, 224, 256 | ✓ | | | | | |
| Mars | 128, 192, 256 | | | ✓ | | | |
| Twofish | 128, 192, 256 | Truncated, Impossible | | | Power Analysis | ✓ | |
| Camellia | 128, 192, 256 | | | | Cache Timing | | Square Attack |
| Threefish | 256, 512, 1024 | Boomerang Attack | | | | | |
| Safer-K | 64,128 | Boomerang Attack, Impossible | | | | ✓ | |
| 3-Way | 96 | | | | | ✓ | |
| Serpent | 128, 192, 256 | Differential-Linear | | | Power Analysis | | |
| AES | 128, 192, 256 | | | Bicliques Attack | | ✓ | Brute Force |
| RSA | 2,048 to 4,096 | | | | | | Shor’s Algorithm |

*Side-Channel Attacks (SCA):* In SCAs, information leaks from a physical cryptosystem. Timing, power consumption, and electromagnetic emissions can also be exploited.

*Related-Key Attacks (RKA):* An attack model called related-key is a subset of cryptanalytic attacks in which the attacker is able to select or identify the relationship between multiple keys. These keys are then used for both encryption and decryption operations.

*Other attacks (Others):* Other attacks, apart from DC, LC, M-in-M, SCA, and RKA, are included in this category.

The survey shows that different attacks on a cryptosystem are possible, even on an AES, a highly secure block cipher. These attacks illustrate a requirement to modify various cryptosystems to make them more secure against cryptographic attacks. Figure 3.3 represents the graphical view of the cipher's cryptanalysis by the specific cryptographic attacks.

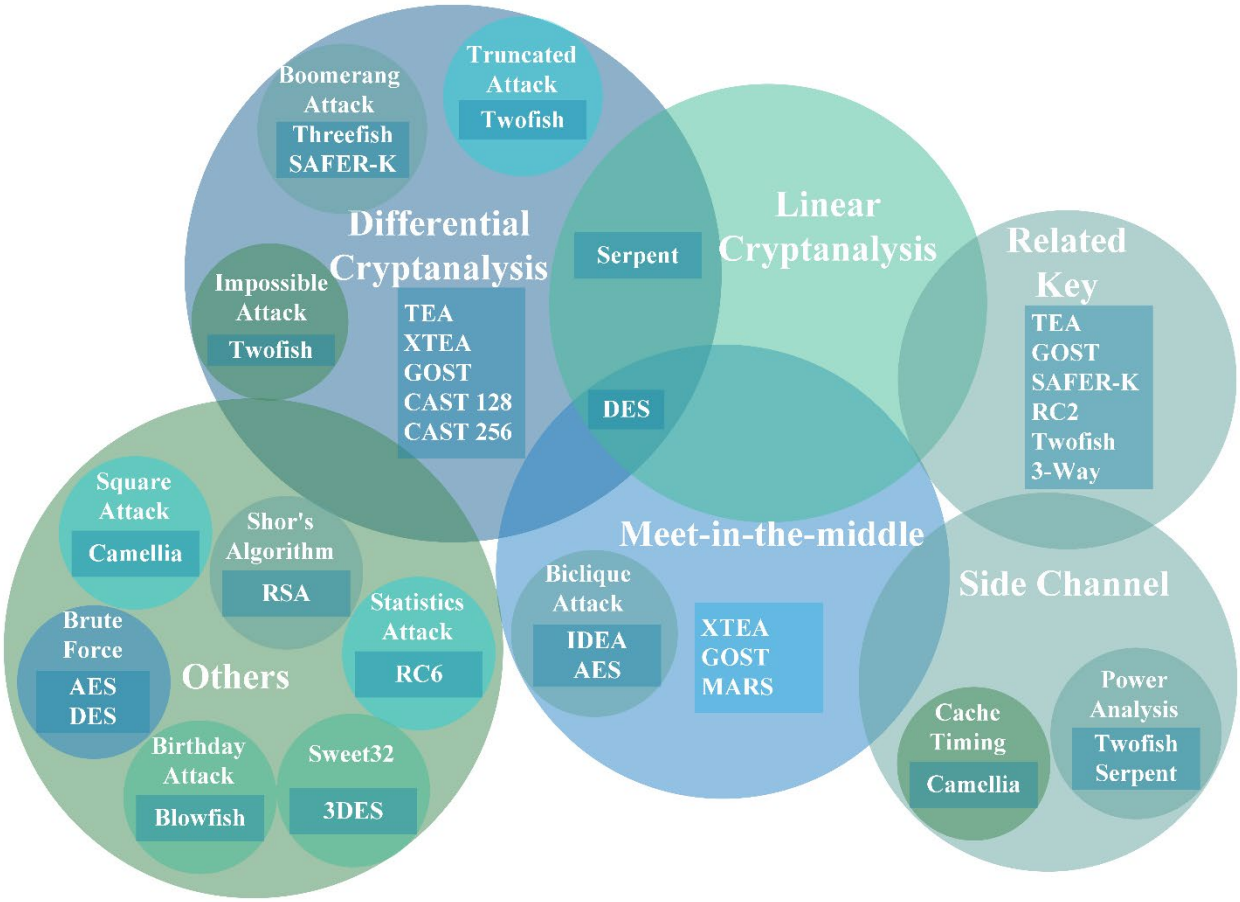



*Figure 3.3 Graphical review of cryptographic cryptoanalysis*

The next category discusses different RNGs and focuses on cryptosystems based on RNGs.

## 3.2. Literature review – random number generator-based cryptosystems

The cryptosystems are affected by the objectives and operations needed to perform their functions. RNGs can play a significant role in providing cryptography's fundamental function, randomness, and enhancing the security of the cryptosystem. The randomness can be achieved by different random method generators.

- *Pseudo Random Number Generator*
- *True Random Number Generator*
- *Circuit Design-Based Random Number Generator*

- *Quantum Random Number Generator*

Figure 3.4 illustrates the categories of RNGs and their implementation methodology or practical approach for integrating them into a cryptosystem.

***Pseudo Random Number Generator***

*PRNGs* are computer-based algorithms that use mathematical computation to generate random numbers. A PRNG sequence of the random number is periodic, which means the sequence repeats itself periodically, and deterministic, in that the sequence reproduction is possible if the initial value is known. Mathematical computations such as linear congruential generators, LFSRs, and chaos provide pseudo-randomness.

A PRNG is a Cryptographic Secure PRNG, or CSPRNG if it passes the next-bit test (if the attacker knows the first k bits, the attacker cannot predict the k+1 bits) and withstands the state compromise extensions (if any stream of bits is guessed correctly, the last bits cannot be predicted).

Programming languages have techniques for generating CSPRNG. The secrets [114] module from Python and SecureRandom [93] class from Java support CSPRNG.

***True random Number generator***

*True Random Number Generator:* Random numbers generated by TRNGs are unbiased, independent, and unpredictable because they are produced by different physical phenomena such as infinite noise, voltage fluctuation, clock jitter, atmospheric radio noise, and continuous- and discrete-time chaotic systems.

There are various USB interface-based TRNGs that help to quickly connect with a laptop to work with any application. Alea II [22] by Araneus Information Systems, TrueRNG v3 [23] by ubldit, and infinite noise TRNG [24] by Crowd Supply are some of the USB-based TRNGs. On the other hand, RANDOM.ORG is a service-oriented online platform that provides TRNGs for various applications such as games, lotteries, and web pages.

***Circuit design-based random Number generator***

*Circuit Design-Based Random Number Generator:* Circuit Design, including various electronic components, such as gates, integrated circuits, is also helpful in generating RNGs. Pietro Nannipieri et al. [115] proposed a TRNG using the Fibonacci-Galois Ring Oscillator for cryptographically secure applications on a Field Programmable Gate Array (FPGA), resulting in the best entropy. Luca Crocetti et al. [116] proposed an all-Digital RNG with high portability

and entropy, resulting in it being ready to integrate with the European Processor Initiative (EPI) [117] chip.

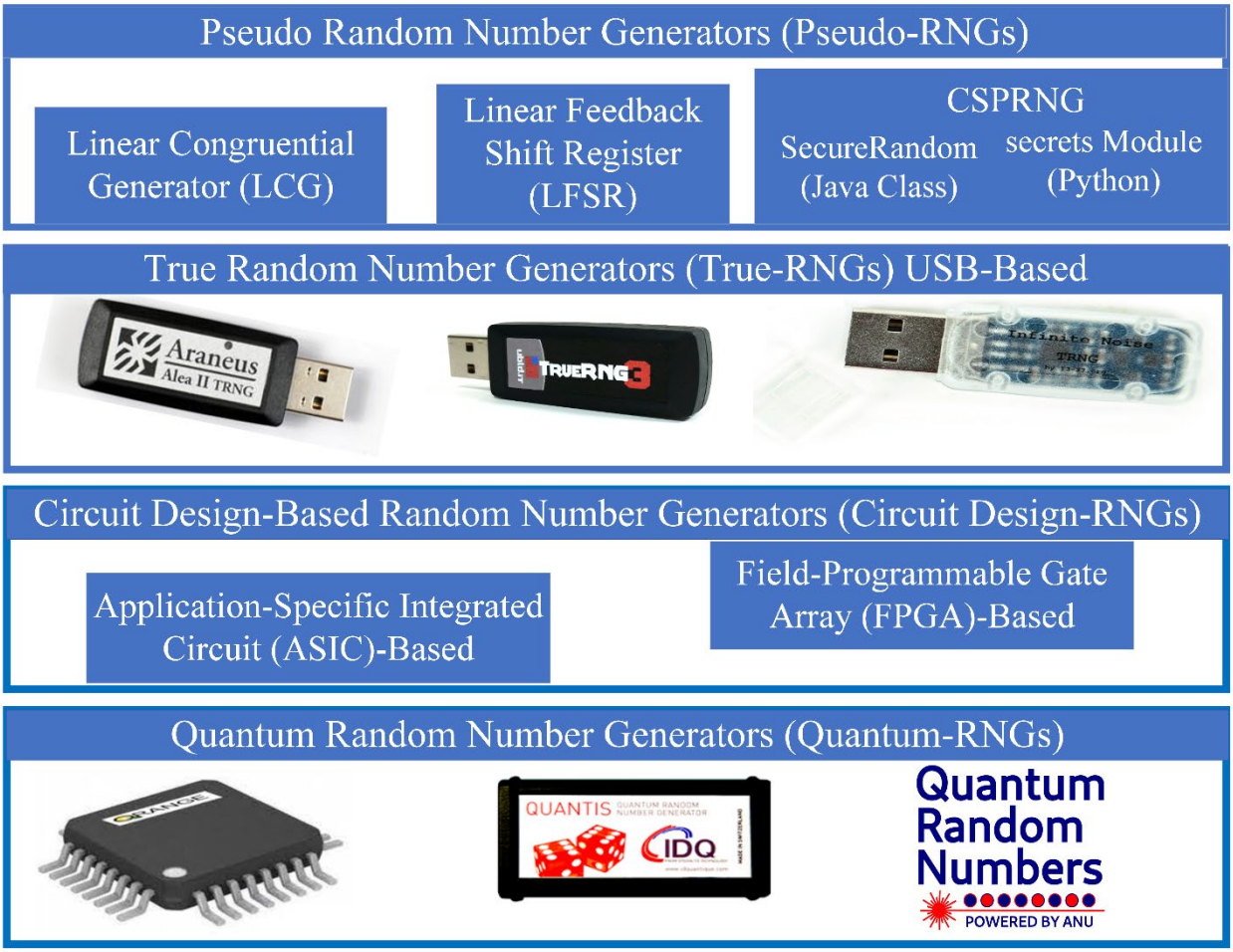



*Figure 3.4 Categorization of random number generators (RNGs): implementation with applications*

***Quantum Random Number Generator***

*Quantum Random Number Generator:* The fourth classification of RNGs is quantum-based, and is derived from quantum mechanics. Different QRNGs generate established and robust outcomes by applying different entropy sources, the source of randomness, based on quantum physics.

USB-based QRNGs are well tested and certified for generating high-quality and unique random numbers such as Quantis [118] by IDQ. They also invented a chip-based QRNG that can fit into small devices such as smartphones. Another project, named QRANGE [119], is under the Quantum Flagship focused on CMOS technology generating quantum random numbers. ANU QRNG [120] is an online platform on which anybody can generate quantum random numbers with just one click and use them.

These RNGs can be used in the RNG-based cryptosystem to enhance the security in the KSA or encryption. Table 3.4 shows the survey of RNG-based cryptosystems, improving different parameters due to the random numbers generated by incorporating RNGs. Also, the survey indicates that RNGs can be used with any cryptosystem and that they make it more secure than the system without RNGs.

*Table 3.4 Survey of RNG-based cryptosystems*

| Cipher Design | RNG | Type of Cryptosystem | Result | Parameters Improved | Ref |
|---|---|---|---|---|---|
| Blostream | PRNG | Symmetric | Immune to Brute Force, Statistical, Deferential, Distinguishing and Correlations Attacks | ➢ Speed,<br>➢ Memory Requirements | [121] |
| Hybrid Cryptosystem | PRNG + TRNG | Symmetric | Strong Key | ➢ Non-deterministic<br>➢ Unpredictable<br>➢ Reproducible<br>➢ Periodic | [122] |
| Text Encryption Stream | PRNG | Symmetric | Immune To All Known Plaintext Cryptanalysis Attacks | ➢ Large Plaintext<br>➢ Key Sensitivity | [123] |
| Present | PRNG + TRNG | Symmetric | Better Performance with High Security | ➢ Slices<br>➢ LookUp Table<br>➢ Frequency<br>➢ Power | A [124] |
| Text Encryption Algorithms | PRNG | Symmetric | Strength against Linear, Differential and Statistical Attacks | ➢ Plaintext sensitivity<br>➢ Key sensitivity<br>➢ Robustness against known plaintext attack | [125] |
| Diffie–Hellman Key Exchange – Using QRNG | QRNG | Asymmetric | Non-Vulnerable Cryptographic System | ➢ Key Entropy<br>➢ Plaintext Entropy<br>➢ Ciphertext Entropy | B [126] |
| CCAES - Chaos-based | PRNG | Symmetric | More Secure and Effective<br><br>Resistant to Differential Attacks | ➢ Histogram<br>➢ Correlation<br>➢ Number of Changing Pixel Rate (NPCR) | [127] |

| Cipher Design | RNG | Type of Cryptosystem | Result | Parameters Improved | Ref |
|---|---|---|---|---|---|
| | | | | ➢ Unified Averaged Changed Intensity (UACI)<br>➢ Information Entropy | |
| Image Encryption Algorithm - Chaos-based | PRNG | Symmetric | Security Enhancement | ➢ Entropy<br>➢ Cross Correlation<br>➢ Mean Square Error<br>➢ Peak Signal to Noise Ratio (PSNR) | [128] |
| Authenticated Encryption with Associated Data (AEAD) – Chaos-based | PRNG | Symmetric | Highly Secure for Ciphertext and Authentication Tag<br><br>Resistant to Differential, Linear, Algebraic, and Timing Attacks. | ➢ Privacy and Integrity<br>➢ Measured by Statistical Test suite NIST, DIEHARD and ENT. | C [129] |
| Hybrid RSA | PRNG | Asymmetric | Strong Encryption | ➢ Histogram<br>➢ Correlation<br>➢ NPCR and UACI,<br>➢ Key Sensitivity<br>➢ Key Space<br>➢ Information Entropy | [130] |

Note: Publication project funded by

A- Supported by the Xiamen University Malaysia Research Fund (XMUMRF)

B- Prometeo Project of the Ministry of Education Superior, Science, Technology and Innovation of the Republic of Ecuador

C- Fundamental Research Grant Scheme funded by the Ministry of Higher Education of Malaysia (MOHE)

## 3.3. Research objectives of quantum-RNGs for cryptosystems

QRNGs as an external device are most useful when the sender and receiver, i.e., Alice and Bob, are the same. Alice wants to store data such as personal files (driving licence, passport, other identity proof) and highly secure work files such as military data on the cloud. So, the data should be encrypted before sending it to the network. Here, the KSA comprises the QRNG bits to make the encrypted key stronger and more complex.

Already discussed are classifications of RNGs in category II, and the survey shows the improvement in various cryptosystem ciphers even using pseudo-randomness, which is not truly random.

It is impossible to consider true randomness in PRNG-generated [21] sequences because they are implemented by software, based on mathematical algorithms, and determined by an initial (seed) value. In contrast, the TRNG and QRNG, based on unpredictable physical means, generate the true randomness in the sequence of random numbers. Furthermore, there are two significant differences between TRNGs and PRNGs. TRNGs [33] are non-deterministic and use a physical mechanism, whereas PRNGs are deterministic and utilize mathematical algorithms. Although TRNGs are unpredictable, they are still vulnerable to attack because a failure in the TRNG hardware system is hard to detect.

Dr Mads Haahr [131] introduced the concept of providing different TRNGs online. Some are freely available such as Numbers-based, Lists, Strings, and Maps-based; some are paid for and used for Random Drawings. Web tools and widgets-based TRNGs are also freely available to be integrated into web pages and to provide the TRNG-based random number. As atmospheric noise creates these random numbers, they are better than pseudo random numbers.

In some TRNGs, the entropy generated by the entropy source may be confused with thermal noise or shot noise [20]. However, a highly secure and unique random number is based on the quantum principle and generated by the QRNG [4]. A QRNG-based cryptosystem incorporates quantum randomness to generate a more robust key of the KSA or to enhance encryption security.

Furthermore, Shor's algorithm [132] concluded that quantum computers could easily break public-key cryptography. It proposed to solve the prime factorization of RSA with high probability. The quantum random numbers generated by QRNGs are also quantum-based and might secure the new cryptosystem without a prime number. Also, incorporating quantum randomness challenges this type of quantum algorithm for cryptoanalysis of the cipher.

In this category, the study discusses the QRNG that generates unbiased, high speed, and unpredictable random numbers generated by various methods such as laser pulses, phases diffusion, photon-based, and other quantum mechanics techniques. Figure 3.5 shows the difference between RNGs in terms of their generation methods, properties, disadvantages and advantages.

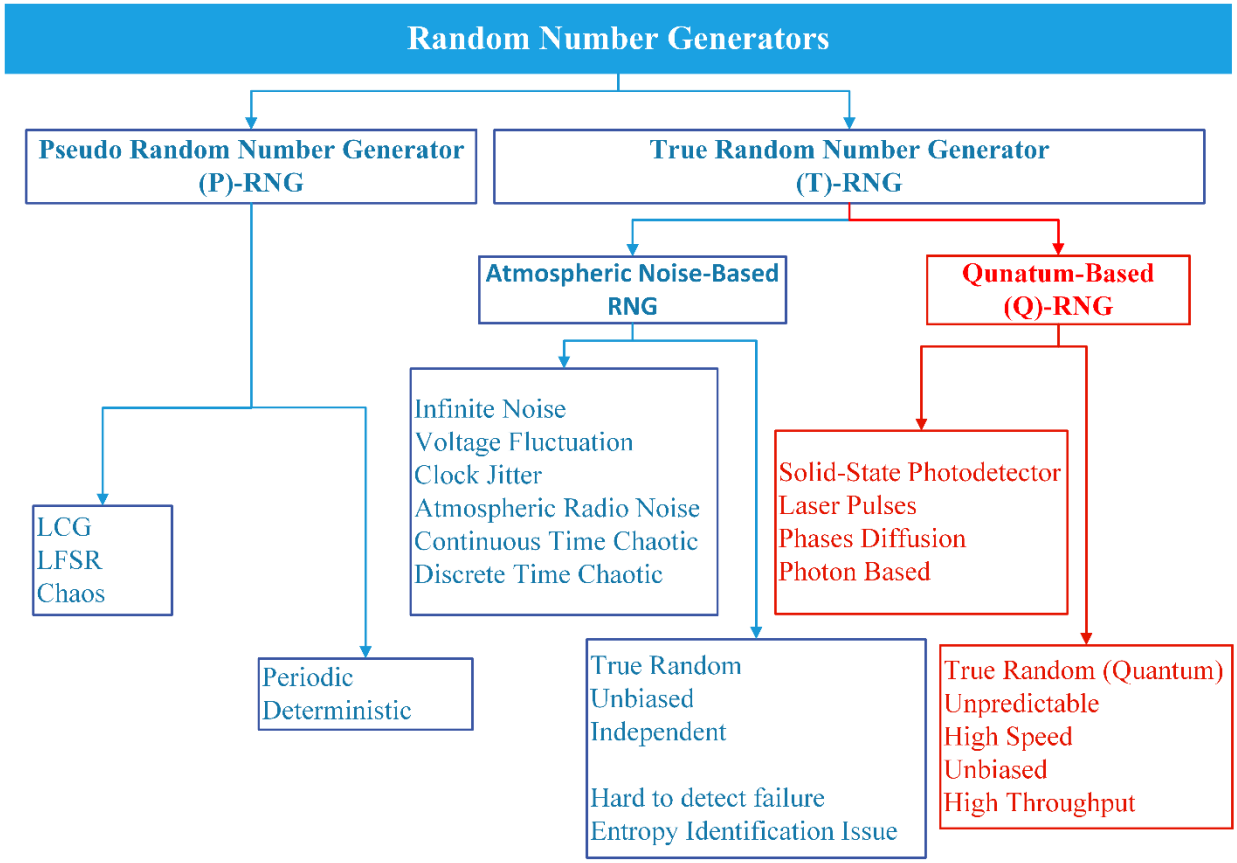



*Figure 3.5 Categorization of random number generators (RNGs): generation methods, properties, disadvantages and advantages*

This section has reviewed 20 QRNGs and divided them into three important research objectives: High Speed (HS), Bias Improvement (BI) and High Efficiency (HE).

*High Speed:* The speed of the RNG device is indicated by the rate at which random bits are generated per second by the device. Speed is critical in determining the optimal trade-off for the RNG device. Different research methodologies attempt to identify the shortest time it takes for the device to generate the most random bits.

*BI*: Random bits should be bias-free. Bias occurs when a significantly larger number of 0s than 1s, or vice versa, are generated. If the difference between 0s and 1s is small enough, it should not introduce bias into the quantum random bits. Numerous research proposals have been designed to improve the bias in order to achieve balanced quantum random bits.

*High Efficiency:* The performance of a QRNG can be improved by researching the different technologies on which the generator can be built. The efficiency of a device is defined in terms of extracting random bits or maximizing the percentage of bits with specific physical photon-based conditions. Table 3.5 shows the survey of different QRNG techniques and their research objectives.

*Table 3.5 Literature review of QRNGs*

| QRNG Proposal Techniques | Ref | HS | BI | HE | Other Properties |
|---|---|---|---|---|---|
| Weak Laser Pulses | [133] | ✓ | ✓ | | |
| Random Quantum States in Mathematica | [134] | ✓ | | | |
| Tapered Amplifiers | [44] | | ✓ | | Self-detecting |
| One Single-Photon Detector | [135] | | ✓ | | |
| Photon Arrival Time Selectively | [45] | | ✓ | | |
| Three-Types QRNG | [136] | ✓ | | | Self-testing |
| Ultra-fast QRNG – Pulsed Laser Diode | [50] | ✓ | | | |
| Heterodyne-Based | [137] | ✓ | | | Security |
| 16 × 16 Pixel QRNG Based on SPADs | [43] | ✓ | | ✓ | |
| Laser Phase Fluctuations | [138] | ✓ | | | |
| One And Two Entropy Sources | [27] | | | ✓ | |
| Phase Diffusion Process-Based | [139] | ✓ | | | |
| SPAD-based QRNG using FPGA | [140] | | | ✓ | |
| SPAD-based QRNG Pixel-based | [141] | | | ✓ | |
| Multi-Bit | [142] | | | ✓ | |
| Uncharacterized Laser & Sunlight | [46] | ✓ | | | |
| Coupled Quantum Dots | [143] | | ✓ | | |
| Phase Diffusion in Lasers | [144] | | ✓ | | Interference Quality, Input/Output Monitoring |
| Quantis – USB | [26] | | ✓ | | Autocalibration Status Monitoring |
| Qrange | [119] | ✓ | | | Security |

**Note:** HS - High Speed; BI – Bias Improvement; HE – High Efficiency

The next section addresses some open research challenges specific to QRNG-based cryptosystems.

### 3.3.1. Open research problems

Section II reviewed the literature, and the analysis identified some open research problems to address to enhance the security of cryptosystems. RNGs play a significant role in securing systems that use random numbers. QRNs [182][33] are based on true randomness and can be used to strengthen the security of existing cryptography systems.

Future research will be needed to address the following challenges:

a) *Incorporate the quantum randomness in stream cipher operations compared to pseudo-based ciphers.*

Boolean arithmetic-based cryptography mainly uses CSPRNG. A Pseudo-Hadamard transform based on the modular operation of Boolean arithmetic provides diffusion in a cryptographic cipher. The diffusion depends on the change of an input bit. An RNG can generate the input bit to affect the cryptanalysis of the cipher.

Guillermo Sosa-Gómez et al. [183] showed the cryptanalysis of PRNGs for cryptographic ciphers. Also, a correlation analysis of the entropy of PRNGs and Hadamard values indicated a strong correlation. On the other hand, a QRNG based on the photon can mitigate this cryptanalysis.

For most CSPRNGs, the entropy is derived from the operating system, along with a PRNG generator [184]. The underlying PRNG often 'reseeds', which means that when an entropy is supplied by the operating system (e.g., from user input, system interruptions, disk I/O or hardware random generators) it changes its internal state. However, quantum randomness is derived from quantum particles without the intervention of reseeding.

b) *Design a KSA using different entropy sources for quantum random bits in order to randomize the keyspace for differential attacks.*

The differential attacks try to find the key with the chosen plaintext and corresponding ciphertext. The subkeys of the KSA are vulnerable if the key space between them is not random or large. Incorporating QRNGs can generate random subkeys, increasing or enhancing the randomness of their key space. The stronger the key space the more challenging it is to detect even a single subkey. As a result, a QRNG-based KSA may produce a strong cipher that is not vulnerable to differential attack.

c) *Study in depth the effects of key-related attacks on QRN-based ciphers*

In key-related attacks, the attacker knows (or chooses) a relation between several keys (up to 256 in some recent attacks) and is given access to encryption functions with these related keys. Keys and plaintexts can be selected with specific differences when using differential RKAs.

The QRNG will support the true randomness in the cryptography key against key-related attacks. The effect of the true randomness on the cryptanalysis of the cipher should be analysed to secure the key and to make it truly random.

d) *Designing a new high-speed encryption cipher based on different research designs (high speed and bias-free), using QRNs to enhance the security of the cryptosystem.*

The encryption speed of the block cipher is one of the major primary concerns when designing a new cipher. However, the security of the cipher must not be compromised on the basis of time, but the time complexity of an algorithm can make it more competitive against alternatives. The QRNG will add true randomness into the key by generating the quantum random bits from an external photon device. These random bits are highly important to secure the cipher, but might come with a time cost. New cryptographic primitives are sought that reduce this time and introduce quantum randomness, leading to a highly secure cryptosystem.

e) *Research storing and exchanging the QRNG-based keys generated for asymmetric cryptosystem over the cloud.*

QRNGs are based on devices that can generate photons. These devices can be with sender or receiver and so are important when both have the same key, i.e., symmetric cryptosystem. They help generate two keys – a pair of public and private keys – but transfer them with the knowledge of quantum random bits and store them on the cloud as a staging post. This needs further research.

The next section focuses on the literature review on Key-schedule Algorithms.

## 3.4. Literature review – Key-schedule Algorithm

In cryptography, encryption methods [16] and ciphers are used to ensure data security and prevent unauthorized access. An encryption algorithm combines plaintext with key information to generate a ciphertext, making the plaintext difficult for hackers to decipher. This process of encryption with a secret key can make the cipher secure by combining nonlinearity, propagation criteria, correlation, algebraic immunity, and randomness [107][99].

As part of an encryption process, the strength of the KSA directly impacts the security of an encryption algorithm. The KSA expands the secret key and generates subkeys according to the number of rounds of encryption required for a particular cryptographic algorithm. This generation and expansion of the secret key into multiple subkeys increases the robustness and complexity of the KSA. One of the potential flaws in this approach is vulnerability in the key-schedule expansion, which is characterized by the algorithm's resistance to cryptoanalysis attacks [41][145][146].

Researchers have demonstrated that logical gates such as AND, NAND, XOR, and OR, as well as Boolean functions with symmetrical properties, can be used to achieve security in the key expansion [147]. In addition to that, quantum data generation by generating quantum random number bits can increase security.

The key generation is a process that creates a key and expands it based on logical operations to encrypt plaintext. The strength of the KSA [147] can be evaluated on the different properties of the key. In 2019, Hakim and Nusrom [148] proposed a new algorithm for scheduling subkeys in the L-block cipher, as the Niaz correlation test concluded that the LBlock's KSA generates keys with a high correlation. Kareem and Rahma [110] proposed a novel method for modifying the Twofish algorithm by implementing multi-level keys in the KSA to control the dynamic block bit sizes and multi-state tables. Using these keys allows for a greater complexity in the algorithm while incurring a relatively small amount of additional computational time. Sulaiman et al. [149] proposed an enhancement of Rijndael's KSA [35]. The analysis conducted by Sulaiman addresses the algorithm's shortcomings and optimizes the KSA in terms of frequency and Strict Avalanche Criteria. Huang et. al. [150] modified the AES's KSA by transposing its subkey matrix. According to the authors, the new KSA is immune to SQUARE, Mt-in-M, and related-key differential type attacks. Shahzadi et al. [151] proposed that 2D Chaotic maps enhance the strength of the generated keys in the KSA of RC5 algorithm, making it difficult for hackers to decrypt the data. Their KSA work targets resource-constrained environments and analyses the security mechanism for specific applications of critical clinical images.

The security of the key generation process starts with the generation of bits based on a random number. Sahmoud et al. [152] proposed generating distinct subkeys from the AES real key using a PRNG and encrypting the block with each subkey of the KSA. Their research focused on the two techniques: first, preventing predictions of obtaining the subkey from an available one, and second, presenting an initialization method to speed up subkey generation. Maram et al. [153] also used a PRNG and proposed a dynamic key-dependent S-box in the KSA that achieved a better avalanche effect with a cryptography algorithm.

Rahul Saha et al. [95] took a different approach, proposing a Symmetric Random Function Generator (SRFG) as a cryptographic function generating randomness in the KSA of the AES. The results indicate that their proposed work has a threefold improvement in terms of confusion property and avalanche effect over the original AES. In another study [111], the FORTIS algorithm developed by Vuppala et al. [111] for generating subkeys of the KSA was implemented on an FPGA and the authors analysed the algorithm's resistance to a side power

channel attack. Gaetan Leurent et. al. [154] presented a new representation for the AES's KSA that efficiently combined the information from the first subkey and the last subkey to reconstruct the master key. Lauren et. al. [32] discussed the security properties of the AES's KSA and its vulnerability to published attacks. Also, based on the information principle introduced by Claude Shannon, they proposed a faster and more secure KSA for generating subkeys in an AES cipher.

Afzal et al. [155] recently published work that is closely related to this research. They proposed a Key-Schedule Evaluation Criterion (KSEC) to evaluate the cryptographic strength of subkeys of different KSAs and to establish a distinction between weak and strong keys using four statistical [156] tests.

They proposed a test suite consisting of a frequency test to analyse the balance of 0 and 1 bits, Bit-Independence tests for confusion and diffusion property, Bitwise-Uncorrelation tests for correlation among subkeys, and high/low-density key tests for testing randomness of the subkeys. Their test suite compares the strength of KSAs of different block ciphers. This is the first time that a framework, CryptoQNRG, has been built to compare the strength of the KSA based on RNGs and various block ciphers.

The next section reviews different categories of the dynamic S-boxes.

## 3.5. Literature review – dynamic S-box

During the past two decades, cryptosystem designers have become increasingly concerned with designing key-dependent S-box methods with random properties. Various cryptoanalysis techniques, such as differential, linear, Mt-in-M, key recovery, and shortcut attacks, require dynamic methods and randomly generated substitution cryptosystems. As such, a strong emphasis has been placed on developing and implementing key-dependent S-box methods with random properties resistant to various cryptoanalysis techniques. Julia Juremi et al. [157] surveyed many dynamic S-boxes focusing on framing the technology for generating them and creating them dynamically to increase cryptographic strengths and resist different attacks significantly. Various methodologies for constructing dynamic S-boxes have been proposed, including PRNGs and TRNGs with specific considerations, Chaos-based, algebraic, and optimization techniques.

There are various approaches to creating a dynamic S-box:

*Algebraic theory-based*: A mixture of classical and modern mathematics, Galois Theory used to processe the solutions of polynomial equations (mathematical equations made of numbers and variables) to form different values of a dynamic S-box [158][159][160][161].

*Chaos theory-based:* This theory uses different patterns and mathematical formulations to generate randomness in the system and corresponds to different dynamic values of the S-box [162][163][164].

*Random number generator-based:* The random numbers generated by sources such as electrical and quantum noise lead to pseudo, true, and quantum generators, which are used to generate the values of the dynamic S-box [165][166].

*Advanced data analysis techniques-based:* Advanced data analysis techniques such as machine learning, particle swarm optimization, GAs, and heuristic techniques analyse the different sources of data and generate the values of the S-box dynamically [167][168][169].

Mangal Deep Gupta et al. [170] proposed an architecture for reconfigurable PRNGs designed using Verilog HDL, and synthesized on the Xilinx tool using the Virtex-5 (XC5VLX50T) and Zynq (XC7Z045) FPGA. Utilizing these PRNGs they designed two 16 × 16 S-boxes. The proposed S-boxes fulfil the cryptographic criteria such as Bijective, Balanced, Nonlinearity, Dynamic Distance, SAC, and BIC nonlinearity criterion. Mengdi Zhao et al. [171] proposed a non-degenerate 2D enhanced quadratic map (2D-EQM) that exhibits ergodicity and randomness. They used it to generate affine transformation matrices and constants for seeding S-boxes based on affine transformation matrices, and generated a batch of keyed strong S-boxes with high nonlinearity.

Researchers have also used irreducible polynomials, additive constants, and static lookup tables [172] to make a dynamic S-box. Alamsyah et al.[173] proposed a combination of an irreducible polynomial (three best irreducible polynomials) and an affine matrix (a Boolean operation XOR for each affine matrix element of the AES S-box). Several high-quality S-boxes with corresponding cryptographic properties are built from the combination of the results of these modifications. Using dynamic S-boxes and shift rows to simplify the encryption process also increased the encryption strength of the AES cipher [174]. Manjula G. et al. [175] proposed that the modified S-boxes are dynamic, random, and key-dependent, adding to the algorithm's complexity and making cracking them more challenging. Praveen et al. [176] proposed a dynamic key-dependent S-box that relies on affine transformations with irreducible polynomials and affine constants. Grasha Jacob et al. [35] developed an S-box that can be generated dynamically based on the sub-byte transformation used in cryptography.

Chaos-based dynamic S-boxes are also of interest to many researchers. Alaa F. Kadhim et al. [169] used shifting, chaotic theory (1D, 2D logistic maps), and particle swarm algorithms to generate a dynamic S-box based on the input key. The system provides enhanced cryptographic properties. The authors of another study [177] proposed an uncorrelated S-box element for generating an S-box that meets cryptographic criteria, such as bijection, nonlinearity, strict avalanche, output bit-independence, equiprobable input/output XOR distribution, and maximum expected linear probability computed on typical chaos-based schemes without taking into account lag-time chaotic series [177]. Muhammad et al. [109] report a new chaos-based affine transformation generation method that uses rotational matrices to generate key-based S-boxes.

Researchers also focus on GAs for creating dynamic S-boxes that lead to secure symmetric cryptosystems. Aguirre et al. [178] proposed high nonlinearity using the multi-objective evolutionary approach to evolve Boolean functions. Stjepan Picek et al. [179] experimented with a GA and programming by modifying the mutation operator and initialization process to search Boolean functions with cryptographic properties. Anand Kumar et al. [180] proposed a robust S-box design with a hybrid approach of using a GA and particle swarm optimization (PSO). Their performance evaluation is better regarding nonlinearity, strict avalanche effects, bits distribution, and bijectivity. By employing a GA in SP boxes and implementing a nonlinear neural network (NN) in the SP network, Kalaisel et al. proposed a redesigned, enhanced AES cryptosystem that increases security against timing attacks and reduces computation time. In order to reduce the probability of the algorithm being impervious to future dialects, Alaa F. Haitham et al. [181] used the chaotic function to generate an initial random sequence of bits and the quantum crossover to provide a new and improved substitution box with increased nonlinearity.

Researchers focused on creating dynamic S-boxes. However, generating dynamicity using quantum randomness and a user key, creating a dynamic S-box, and retaining the box's cryptographic properties still present a challenge for research.

Our proposed substitution method is a contribution to enhancing the security of symmetric cryptosystems in HSCSN by generating the values of the S-box at the time of execution from the secret key and QRNG.

## 3.6. Summary and conclusion

This chapter reviewed the various literature associated with this research in terms of cryptosystems of HSCSN, associated attacks, and the impact of RNGs used in different cryptosystems. The focus on QRNGs highlighted their important properties in using the KSA and S-box along with the GA. Other researchers focused on creating a dynamic S-box using various techniques and focused on improving its cryptographic property.

# Chapter 4. Methodology

## 4.1. Introduction

The methodology of this study comprises different stages of review, designing the framework, the algorithm, and quantitively analysing the proposed work. This chapter outlines the research design and experimental set-up. The hardware and software specifications necessary for replication are detailed and there is a thorough explanation of data analysis.

## 4.2. Research design

Figure 4.1 shows the research process for this study. The process starts with the preliminary research and identifying the gaps related to data security of the HSCSN. Preliminary research is followed by intensive research focusing deeply on related security prospects and components, so formulating the research aim, objectives, and question. The design and development of the framework and algorithm leads to the novel research of this study. The study implements the research practically, and testing phases test the result on the basis of data collection. Finally, the analysis phase assesses the results and concludes the research objectives.

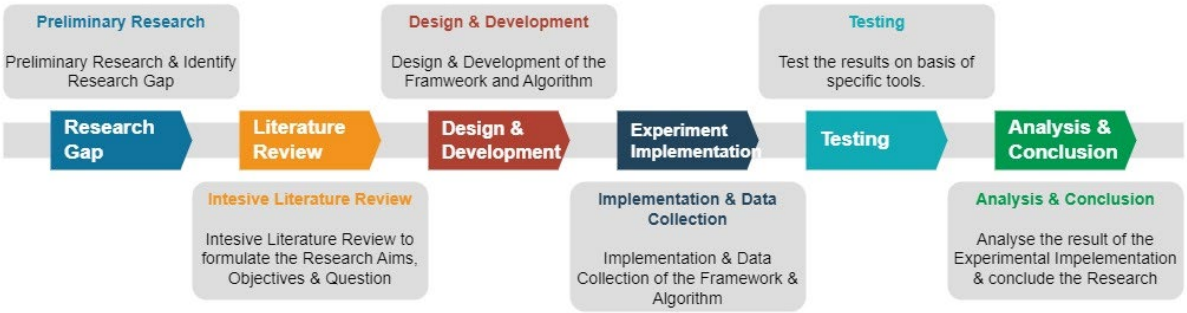



*Figure 4.1 Research design*

*Preliminary Research* focuses on how the user acts as sender and receiver to send the data stored on the cloud rather than as local data. The important components of the communication between the user and the data travelling through the high-speed network move forward this research towards the security of the data.

The literature review focuses on the various contexts of data security by the cryptosystems in HSCSN and various attacks on the encryption and its components to identify the research gaps. The Key-Scheduling Algorithm and its S-box is the main area of focus to work with true randomness. The randomness made the static S-box dynamic through quantum randomness and evolution computation, using the GA to find the optimized solution.

*Design & Development framework* focuses on the KSA and dynamic S-box to develop the novel KSA framework for RNGs and the algorithm needed to design the quantum randomness and GA-based dynamic S-box for the KSA.

*Experimental implementation* implements this framework and algorithm and generates the data for testing and statistical analysis.

*The testing phases* focus on the testing of the data with different tools and mathematical formulae, leading to the *analysis phases*, which are based on the cryptographic property of the KSA and dynamic S-box.

## 4.3. Research methodology

The data security of the HSCSN is ensured by encrypting the data into a ciphertext, with the KSA and S-box serving as its primary components. The dynamic nature of the S-box enhances the security of the data travelling into the network. This dynamic nature is related to randomness, and shifts this research towards quantum randomness, the true randomness of photons and the GA of evolution.

The techniques used in this research are based on quantitative analysis, which is used to analyse the cryptographic properties of the KSA and dynamic S-box. The analysis of their cryptographic behaviours is done through statistical testing of the numerical data to test the hypothesis and produce the analysis results.

### 4.3.1. Design and development

The objectives of this study leads to the development and design of the KSA-RNG evaluation framework and QRNG-GA-based dynamic S-box algorithm.

*KSA-RNG evaluation framework*

The framework evaluates the strong and weak keys of the various KSAs of the block cipher. The framework consists of different statistical formulae to satisfy the cryptographic properties of the keys. Figure 4.2 shows the block diagram of the KSA-RNG evaluation framework. The two different bit streams of the two different RNGs of the block cipher's KSA will be given to a set of criteria proposed as an evaluation framework. The framework consists of different criteria to justify the cryptographic properties of the KSA. The output will differentiate the strong or weak keys based on different the RNGs of the KSA.

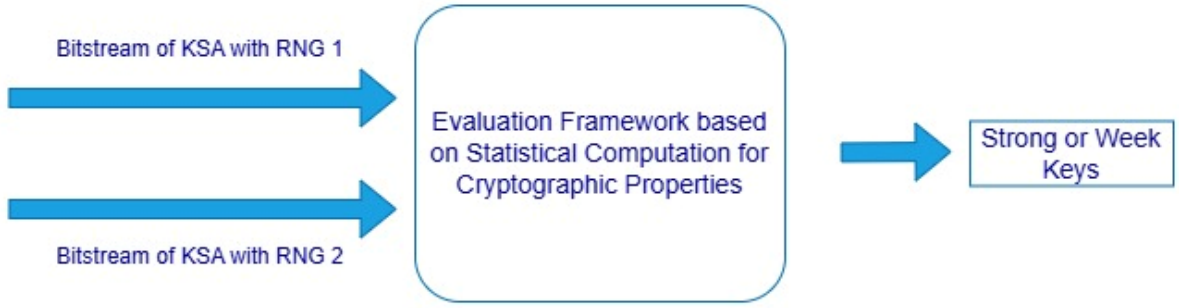



*Figure 4.2 Block diagram of KSA-RNG evaluation framework*

### *QRNG-GA-based dynamic S-box*

An evolutionary computation algorithm with the QRNG will make a QRNG-based dynamic S-box This S-box will be used in the KSA to generate an encryption key. Figure 4.3 shows the block diagram of the proposed work.

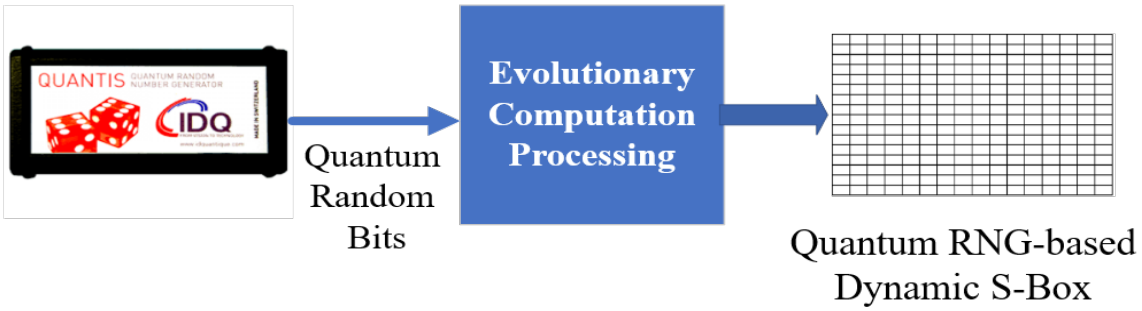



*Figure 4.3 Block diagram of the proposed QRNG-based dynamic S-box*

Input: Quantum random bits from QRNG along with user-defined key is taken as input to the evolutionary computation.

Processing: The evolutionary computation process first converts the user-defined key into bits and processes it with quantum random bits for the cryptographic properties.

The algorithm based on a GA first represents these bits in a genetic representation and then fitness function to evaluate the suitability for a cryptographic property. The genetic representation can be an array of bits, binary, or parse trees. The genetic operators are the basis of the algorithm's design criteria: selection, crossover, and mutation with fitness function, and will be scored for different cryptographic properties: nonlinearity, balance, LAP, DP, and randomness.

Output: The 256 elements of the 16 x 16 matrix of the S-box, and each element, has two nibbles (a half byte) generating a dynamic S-box for a specific KSA based on quantum random bits and evolutionary computation.

### 4.3.2. Implementation set-up & tools:

Hardware: Quantis [68] – A USB-based QRNG developed by IDQ. Its general specifications include random bit rate 1 : 4 Mbit/s ± 10% (Quantis-USB-4M), Thermal noise contribution: < 1% (Fraction of random bits arising from thermal noise), Storage temperature : -25 to + 85°C, USB specification 2.0 and Power Via USB port

Computer Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM: 8.00 GB, System type: 64-bit operating system, x64-based processor

Software: Java – Netbeans with JDK 1.8.0

Quantis, the USB-based QRNG used in this research is compatible with the Java-based library and used to generate the quantum random number from the device. Also, Java has a supporting library to implement GAs. Java is a more suitable language compared to other languages such as python and go for implementing the proposed framework KSA-RNG evaluation framework and dynamic S-box QRNG-GA-based dynamic S-box.

The Cryptol [185] language code generates keys for the different KSAs of the block ciphers for the KSA-RNG evaluation framework.

### 4.3.3. Data collection

*KSA-RNG evaluation framework*

The data collection of the framework is based on different keys generated by the KSA of the various block ciphers. Two different RNGs (pseudo and quantum random numbers) will be used in Cryptol language to create RNG-based keys for different block ciphers.

*QRNG-GA-based dynamic S-box*

The data collection for QRNG-GA-based dynamic S-box will be the dynamic S-box generated based on the quantum random numbers and the GA-based generation evolution computation.

### 4.3.4. Testing

*KSA-RNG evaluation framework:*

The framework will be tested and evaluated by different statistical and hypothesis tests [186][187], taking into consideration the cryptographic properties of the KSA.

*QRNG-GA-based dynamic S-box*

The output of the algorithm with different cryptographic parameters of QRNG-GA-based dynamic S-box will be tested by SET tool [55].

#### 4.3.5. Analysis and conclusion

*KSA-RNG evaluation framework*

The framework focuses on the analysis and conclusion of differentiating the strong or weak keys based on different RNGs of the KSA.

*QRNG-GA-based dynamic S-box*

The output of the algorithm generating a dynamic S-box based on quantum randomness and GA will be evaluated based on the cryptographic properties of the S-box. The characteristics of the proposed S-box will be compared to other state-of-the-art S-boxes to validate it.

### 4.4. Attacker model of HSCSN

The cloud network is vulnerable to a variety of different types of attacks, including network, physical, and software-based attacks, which aim to compromise and infiltrate the network. This study focuses on data security through encryption. The attacker attacks to access the key to crypt the data and gain access to the data travelling through the network. Figure 4.4 illustrates the classification of various attacks on the HSCSN.

*Attacks on network*

An attack on a network [188] occurs when an attacker attempts to access resources on the network by exploiting vulnerabilities. Various types of attacks such as man-in-the-middle attacks, denial of service attacks, and eavesdropping attacks are dealt with by hackers attempting to spoof the entire network.

*Attacks on hardware*

An attack on hardware [189] involves the techniques used to compromise the components of the system that are used to communicate with the user or with the network as a whole. USB attacks are attacks carried out with pre-programmed USB devices to take advantage of the USB port.

*Attacks on data*

The information derived from data helps the attacker gain knowledge. Attackers target the system's key components to gain access to sensitive data. The encryption key attack targets the various components of encryption to gain access to the key used to encrypt the data. The data

security measures are designed to protect data from these attacks [190][191] and make it harder for attackers to execute them. This research focuses on the security of data travelling over high-speed networks to cloud storage. Section 3.1.2 reviewed various attacks on encryption key access.

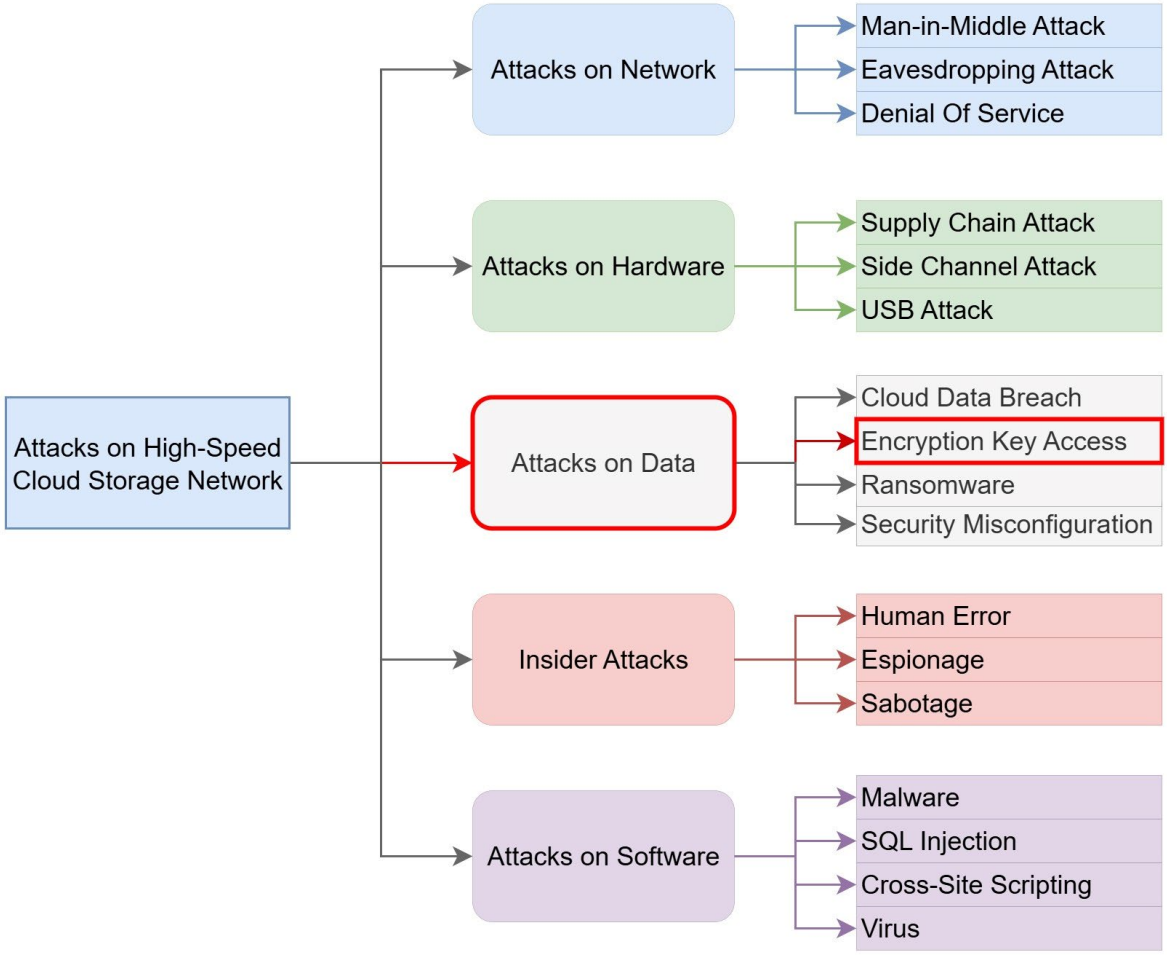



*Figure 4.4 Classification of attacks on high-speed cloud storage network*

*Insider attacks*

Insider attacks [192] are designed to exploit the vulnerabilities of an organization's human side. Often, attackers employ social engineering and espionage tactics to exploit human vulnerabilities and achieve success in their attacks.

*Attacks on software*

Software attacks [193] include malicious code such as viruses, malware, SQL injection, or cross-site scripting that takes advantage of vulnerabilities to gain unauthorized access to the system or disrupt its operation.

## 4.5. Threat modelling for HSCSN

Figure 4.5 shows the threat model for the network created in the Microsoft Threat Modeling Tool 2016. The model represents the data starting from the cloud human user, Alice, and flowing through the high-speed network to be stored on the cloud. The data flows from different nodes in a bi-directional manner.

Table 4.1 presents a portion of the STRIDE framework[194], focusing on information disclosure, to analyse the threats to the HSCSN related to data security.

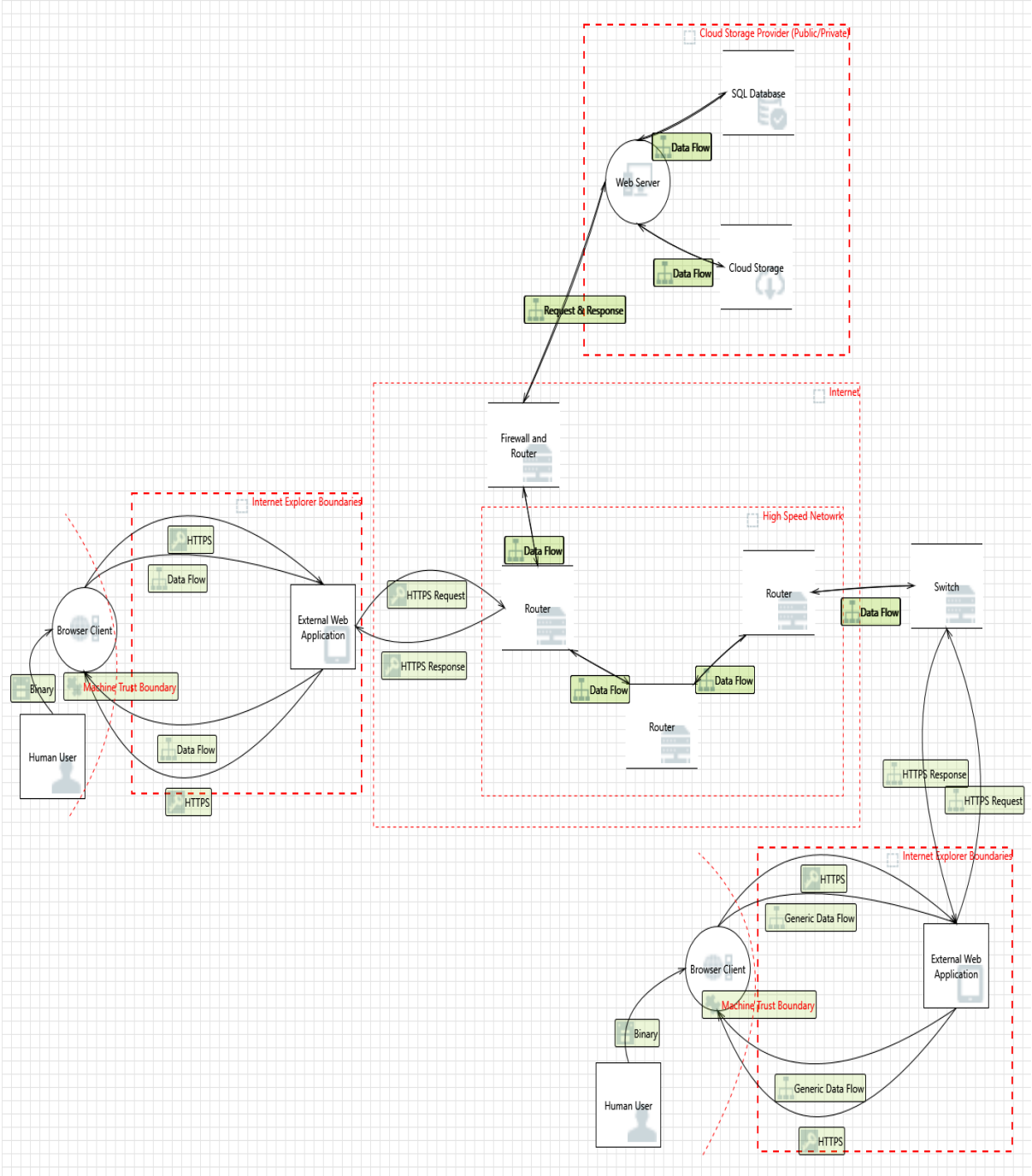



*Figure 4.5 Threat model for HSCSN*

*Table 4.1 Information disclosure threats to HSCSN*

| | | |
|---|---|---|
| **Information Disclosure** | Weak Access Control for a Resource | Improper data protection of router can allow an attacker to read information not intended for disclosure. |
| | Data Flow Sniffing | Data flowing across data flow may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. |
| | Weak Access Control for a Resource | Improper data protection of cloud storage can allow an attacker to read information not intended for disclosure. |
| | Weak Access Control for a Resource | Improper data protection of SQL database can allow an attacker to read information not intended for disclosure. |
| | Weak Access Control for a Resource | Improper data protection of firewall and router can allow an attacker to read information not intended for disclosure. |
| | Data Flow Sniffing | Data flowing across request & response may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. |
| | Data Flow Sniffing | Data flowing across generic data flow may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. |
| | Weak Access Control for a Resource | Improper data protection of Switch can allow an attacker to read information not intended for disclosure. |

## 4.6. Summary and conclusion

This chapter highlighted the research design and methodology, including the tools and experimental set-up required. An attack model described various attacks associated with HSCSN and focused on data security attacks. The threat model exhibits the threats associated

with information disclosure based on the STRIDE model. The next chapter will focus on the design and development of the KSA-RNG evaluation framework based on the statistical formulation and cryptographic properties.

# Chapter 5. CryptoQNRG: A new framework for KSA's cryptographic strength evaluation

## 5.1. Introduction

A KSA generates subkeys based on the input of the user-defined key. An RNG-based KSA adds another layer of security to the subkeys. CryptoQNRG has a series of test criteria used to evaluate the strength of an RNG-based KSA.

An improvement in terms of RNG, from pseudo-randomness to quantum randomness, will benefit a KSA. The focus of this article is to create a framework to evaluate the strength of cryptography KSAs generated by RNGs. In particular the concrete contributions of this work are:

- Formulation of a test suite to evaluate the strength of cryptography KSAs generated by RNGs. This test suite consists of four parts: a bit-frequency test, a bit-correlation test, a bit-interfold test, and a bit-entropy test.
- Applying the proposed criteria in the framework's test suite to a number of different block ciphers' key-scheduling algorithms. The P-value is a measure of the statistical significance of a test result calculated by statistical computation. The test is based on statistics and provides a P-value and the probability of achieving a pass or fail result. A pass does not indicate that the cipher is secure against all attacks. However, a failure suggests that the algorithm is highly susceptible to attacks.

In contrast to existing randomness tests such as CryptRndTest, NIST RNG test, and Diehard battery test, the proposed test suite of this study evaluates the strength of the KSA's subkeys [195] [196] [197]. The tests of the framework are designed to evaluate the main properties of a key-schedule such as unpredictability, balance, confusion, diffusion, and correlation. Moreover, the proposed work compares two different RNG-based key-schedules of the same block cipher simultaneously to distinguish their strength.

Figure 5.1 illustrates an example of a basic scenario considered in this research. In this scenario personal data are stored on an internet server (cloud or a data centre) so that they can be easily accessible from anywhere.

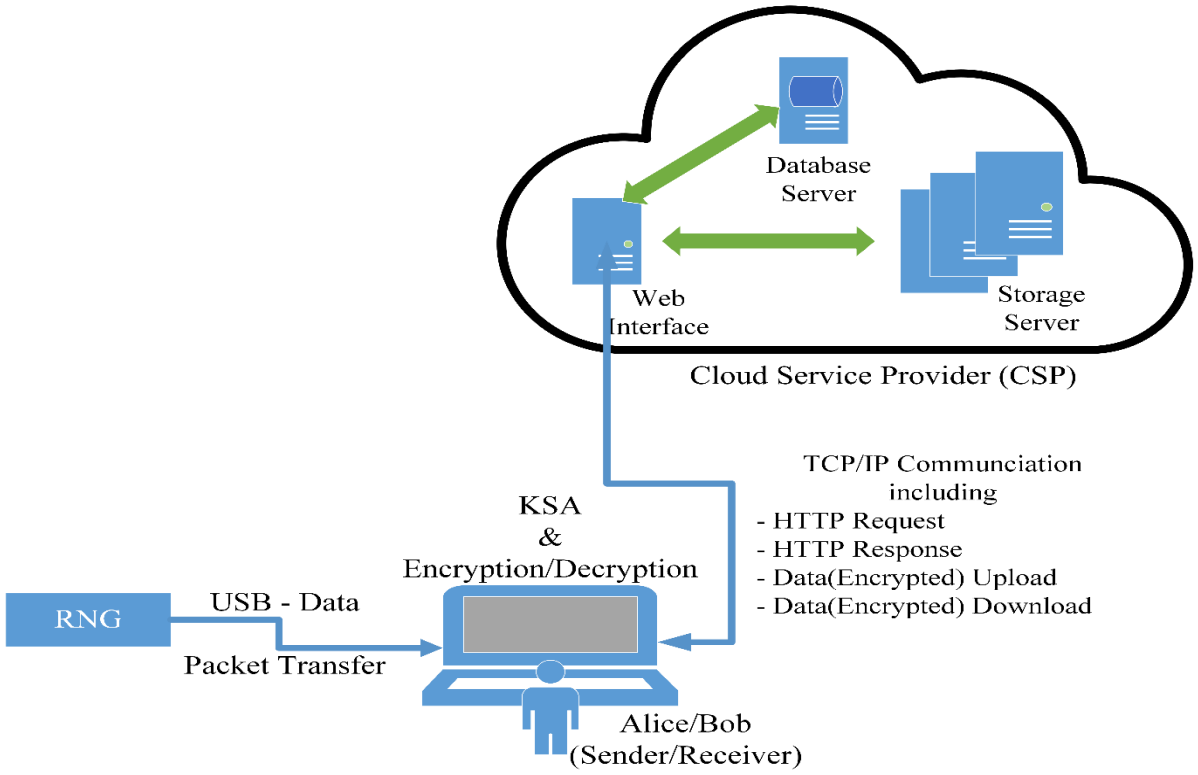



*Figure 5.1 Basic scenario of a secure communication*

The requirement is that the files must be encrypted before being sent over the network to make them secure, and when the user downloads them they can be decrypted locally. An RNG plays a significant role as the KSA key comprises the bits of random numbers generated by an RNG to encrypt the contents.

## 5.2. The CryptoQNRG framework

Figure 5.2 shows the block diagram of CryptoQNRG for evaluating the strength of an RNG-based KSA. The two keys, $K_1$ and $K_2$, are subkeys of individual KSAs generated using two different RNGs. The $K_1$ and $K_2$ are then passed to the proposed KSEC – $KSEC_{RNG}(K_1, K_2)$ – and tested for the cryptographic properties. The criterion includes four statistical tests. The first test, the Frequency test $FT(K_1, K_2)$, calculates the frequency of 0s and 1s and checks how balanced their distribution is. The second test, the Bit-Correlation $BCT(K_1, K_2)$ test, measures correlation, whereas Bit-Interfold $BIT(K_1, K_2)$, the third test, checks for the confusion and diffusion properties. The last test is called the Bit-Entropy $BET(K_1, K_2)$ test and examines the unpredictability of the bit stream generation. Based on the evaluation of each test, the strength of the KSA is considered. The strength of the KSA of $K_1$ and $K_2$ is strong if $K_1$ and $K_2$ pass all

the tests of the $KSEC_{RNG}(K_1, K_2)$; otherwise, the KSA is weak. The proposed criterion CryptoQNRG also compares $K_1$ and $K_2$ with the Bit-Entropy test.

The strength of KSA ($ST_{KSA}$) and difference of KSA ($DF_{KSA}$) are defined as follows:

$$ST_{KSA} = \begin{cases} \text{Strong}, \ if \ (K_1, K_2) \ pass \ each \ test \ in \ EC_{RNG}(K_1, K_2) \\ \text{Weak}, \ otherwise \end{cases}$$

$$DF_{KSA} = \begin{cases} K_1 \ is \ better \ then \ K_2, \ if \ K_1 > K_2 \ in \ BET \ (K_1, K_2) \ tests \\ K_2, \ otherwis \end{cases}$$

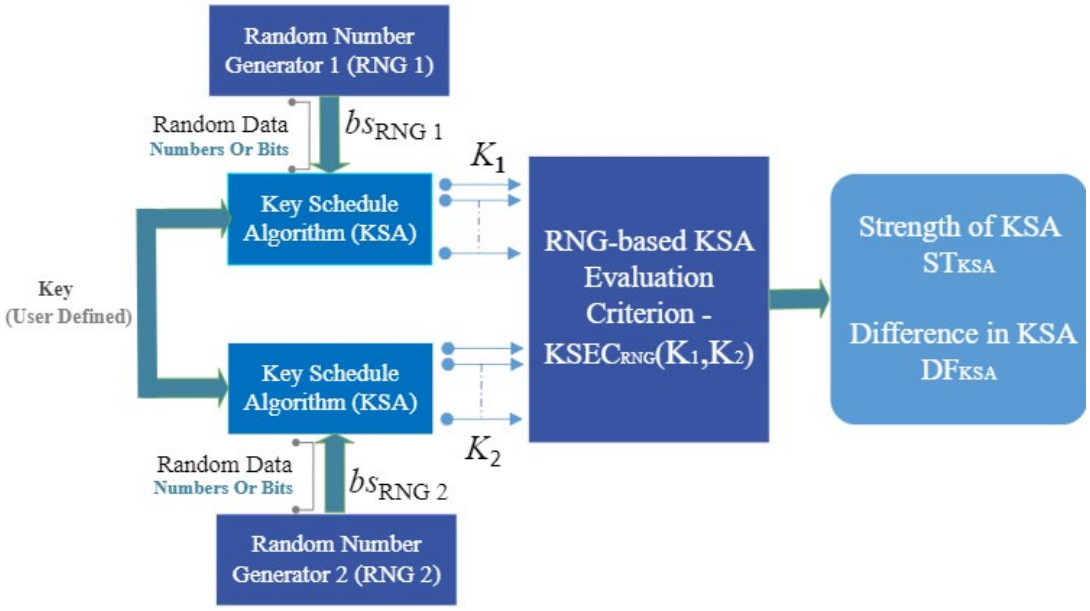



*Figure 5.2 Block diagram of CryptoQNRG*

The following notation is used in the equations below:

$Key$ ... user-defined key

$bs_{RNGi}$ ... the bitstring generated by RNG i, with $i \in \{1,2\}$

$K_i$ ... the subkey of length $L$, obtained with the KSA based on RNG i using multiple iterations ($rounds$), with $i \in \{1,2\}$:

$$K_i = KSA(Key, bs_{RNGi}, rounds, L)$$

$K_{i,j}$ ... the j-th bit of subkey $K_i$: $1 \leq j \leq len(K_i)$ for $i \in \{1,2\}$

$len(K_i)$ ... number of bits in $K_i$ , with $i \in \{1,2\}$

(note that for all keys $K_i$ the length $len(K_i)$ $is\ even$)

### 5.2.1. Frequency test $FT(K_1, K_2)$

The first test to be performed is the frequency test, which checks that the number of 1s and 0s in a sequence is the same, in order to avoid biasing in the $K_i$. The test measures the distribution

of bits in the RNG-based key-schedule algorithms. The sequence is a set of secret subkeys of different block ciphers. A subkey that fails the frequency test is considered weak because it fails the fundamental requirement of randomness, and there is no need to investigate other weaknesses based on the remaining tests. The $K_1$ and $K_2$ are balanced if it satisfies the key bits as;

$$FT(K_1, K_2) = \forall K_i \in \{K_1, K_2\}. \left| \bigcup_{K_{i,j} \in K_i \wedge K_{i,j} = 0} K_{i,j} \right| == \left| \bigcup_{K_{i,j} \in K_i \wedge K_{i,j} = 1} K_{i,j} \right| == \frac{len(K_i)}{2} \quad 5\text{-}1$$

where $K_i$ is the key obtained with the KSA based on RNG ($KSA_{RNG\ i}$) with $i \in \{1,2\}$

### 5.2.2. Bit-Correlation $BCT(K_1, K_2)$

The second test is the Bit-Correlation test, which measures the correlation of bits in the $K_B$. This evaluation is divided into two parts, the *Rogers-Tanimoto* distance measure $R(K_1, K_2)$ and the *Pearson's Correlation* $r_{K_1,K_2}$.

The first part computes the dissimilarity index between the two different RNG key-schedules with the *Rogers-Tanimoto* distance measure $R(K_1, K_2)$ [198]:

$$R(K_1, K_2) = \frac{T}{C_{11} + C_{00} + T} \quad 5\text{-}2$$

where $C_{pq}$ is the number of corresponding pairs of elements in $K_1$ and $K_2$ respectively equal to p and q and $T = 2(C_{10} + C_{01})$.

The second part calculates the *Pearson's Correlation* $r_{K_1,K_2}$ [199]. Based on that the research tests whether the key-schedules $K_1$, $K_2$ are independent (null hypothesis $H_0$) or dependent (alternative hypothesis $H_A$).

The *Pearson's Correlation* $r_{K_1,K_2}$ will be calculated as follows:

$$r_{K_1,K_2} = \frac{\sum_{j=1}^{len(K_i)} (K_{1,j} - \overline{K_1})(K_{2,j} - \overline{K_2})}{\sqrt{\sum_{j=1}^{len(K_i)} (K_{1,j} - \overline{K_1})} \sqrt{\sum_{j=1}^{len(K_i)} (K_{2,j} - \overline{K_2})}} \quad 5\text{-}3$$

where $\overline{K_i}$ is the mean value of $K_{i,j}$ with j = 1 ... $len(K_i)$.

Performing the Hypothesis Test based on the P-value generated by $r_{K_1,K_2}$:

- Null hypothesis $H_0$: $K_1$ and $K_2$ are dependent on each other

$$H_0 = \left(r_{K_1,K_2} \leq 0.1\right) \quad 5\text{-}4$$

- Alternative hypothesis $H_A$: $K_1$ and $K_2$ are independent of each other

$$H_A = \left(r_{K_1,K_2} > 0.1\right) \quad 5\text{-}5$$

The Bit-Correlation Test $BCT(K_1, K_2)$ combines the two parts and is calculated as follows:

$$BCT(K_1, K_2) = \begin{cases} Pass, H_0 \ \wedge \ (\ R(K_1, K_2) \leq 0.5\ ) \\ Fail, \ otherwise \end{cases} \quad 5\text{-}6$$

The advantage of using $R(K_1, K_2)$ to compute dissimilarity is that it calculates the value of symmetric binary attributes. Symmetric binary attributes mean both attributes are of the same significance, and in terms of the input to this test, both the bits 0 and 1 are equally significant. If $K_1$ and $K_2$ pass the frequency test, only $R(K_1, K_2)$ is computed. The next part $r_{K_1,K_2}$ determines the exact extent of the linear corelation between $K_1$ and $K_2$. Linear relationships occur when one variable changes proportionally to another.

Therefore, $BCT(K_1, K_2)$ combines the linearity and dissimilarity test for $K_1$ and $K_2$. The $K_1$ and $K_2$ are considered acceptable if the dissimilarity index of the $R(K_1, K_2)$ is greater than or equal to the threshold value of 0.5 and is not linearly dependent on any other subkey.

### 5.2.3. Bit-Interfold $BIT(K_1, K_2)$

The third test is the Bit-Interfold test $BIT(K_1, K_2)$, which measures the confusion and diffusion in the $K_1$ and $K_2$, which in turn is an important cryptographic property. This test is divided into two parts.

The first part is the calculation of the Hamming Distance $H(K_1, K_2)$ between the two subkeys $K_1$and $K_2$. The Hamming Distance is a dissimilarity distance [200]. $H(K_1, K_2)$ is calculated as the number of bit positions of $K_{1,j}$ in $K_1$ that are different to those of $K_{2,j}$ in $K_2$, divided by the subkey length:

$$H(K_1, K_2) = \frac{\left|\left\{K_{1,j} \ | K_{1,j} \neq K_{2,j} \wedge \left(0 \leq j < len(K_1)\right)\right\}\right|}{len(K_1)} \quad 5\text{-}7$$

The Inverse Hamming Distance $\overline{H(K_1, K_2)}$, which is the inverse of $H(K_1, K_2)$, that is, counting the number of equal bit positions, is calculated as follows:

$$\overline{H(K_1, K_2)} = len(K_1) - H(K_1, K_2) \quad 5\text{-}8$$

$\overline{H(K_1, K_2)}$ refers to whether the bits' position will produce the same proportion of confusion and diffusion in $K_1$ and $K_2$. Confusion refers to the process of combining subkey bits with plaintext to make a cipher. Diffusion refers to the change in a plaintext resulting in changes to the bit order in the subkeys $K_1$ and $K_2$. To analyse this proportion of similar bits leading to

complex subkeys in $K_1$ and $K_2$ that are the basis of confusion and diffusion, the second part Z-Proportion [201] statistics hypothesis, is introduced.

In the second part, the calculated *Inverse Hamming Distance* $\overline{H(K_1, K_2)}$ is used to evaluate the Z-Proportion [201] statistics hypothesis. The analysis checks whether the confusion and diffusion will be similar by $K_1$ and $K_2$.

The Z-proportion test is calculated as follows:

$$Z(K_1, K_2) = \frac{\overline{H(K_1, K_2)} - k_0}{\sqrt{\frac{k_0(1-k_0)}{len(K_1)}}} \quad 5\text{-}9$$

where $k_0$ is the hypothesized value of population proportion in the null hypothesis, i.e., it is the acceptance threshold of the Hamming Distance. $len(K_1)$ is the sample size.

The value of $k_0$ is 0.7, because at least 70 per cent of bits differ in $K_1$ and $K_2$ to justify the bits responsible for confusion and diffusion; KSA is considered strong.

The *Null Hypothesis* ($H_0$) and *Alternative Hypothesis* ($H_A$) based on a P-value computed by $Z(K_1, K_2)$ are as follows:

$H_0$ = Confusion and Diffusion is similar in $K_1$ and $K_2$.

$$H_0 = Z(K_1, K_2) \leq 0.7 \quad 5\text{-}10$$

$H_A$ = Confusion and Diffusion is not similar in $K_1$ and $K_2$.

$$H_A = Z(K_1, K_2) > 0.7 \quad 5\text{-}11$$

The Bit-Interfold Test $BIT(K_1, K_2)$ is calculated as:

$$BIT(K_1, K_2) = \begin{cases} Pass, H_A \\ Fail, H_0 \end{cases} \quad 5\text{-}12$$

The advantage of using $H(K_1, K_2)$ to compute dissimilarity is that it calculates the value of the exact number of different bits in $K_1$ and $K_2$. The inversion of $H(K_1, K_2)$, i.e., $\overline{H(K_1, K_2)}$, is given to the $Z(K_1, K_2)$ to analyse the proportion of bits responsible for generating similar confusion and diffusion by $K_1$ and $K_2$. $K_1$ and $K_2$ pass the Bit-Interfold Test if the confusion and diffusion with threshold value is not similar in both the KSAs, i.e., the $Z(K_1, K_2)$ results in the *Alternative Hypothesis (HA).*

### 5.2.4. Bit-Entropy $BET(K_1, K_2)$

The entropy is a measure of a random variable's uncertainty, and plays a critical role in information theory. The higher the entropy, the greater is the uncertainty in predicting the value of an observation. There are various definitions available for entropy. This work uses the Shannon entropy [202] (or *entr* for short) and the BiEntropy [203] (or *BiEn* for short).

The *entr* calculates the entropy as the amount of information conveyed when identifying a random outcome. The *BiEn* is a weighted average of the Shannon entropies of the string and the first n − 2 binary derivatives of the string.

The *entr* is advantageous for the larger value of binary strings, whereas the *BiEn* calculation is helpful for the smaller length of binary strings.

The *entr* $E(K_i)$ of $K_1$ and $K_2$, which takes values from the set $A=\{K_{i,j}, K_{i,j+1}, \ldots, K_{i,n}\}$ with probability Pr (X=$K_i$) = $K_{i,j}$ for $i \in \{1,2\}$ (keys $K_1, K_2$) and $j \in \{1,2, \ldots, len(K_1)\}$, is defined as:

$$E(K_i) = -\sum_{j=1}^{len(K_i)} p(K_{i,j})\, log_2(K_{i,j}) \qquad 5\text{-}13$$

The *BiEn* $BiEn(K_i)$ is calculated as follows for $K_1$ and $K_2$ distinctly. The *BiEn* value ranges from 0 to 1. When the disorder is more significant in a binary string, the *BiEn* value will be higher.

$$BiEn(K_i) = (1/(2^{n-1}-1))\left[\sum_{b=0}^{n-2}\left(-p(b).log_2 p(b) - \big(1-p(b)\big).log_2\big(1-p(b)\big)\right).2^b\right] \qquad 5\text{-}14$$

where, $p(b)$ is the proportion of 1s in $K_1$ and $K_2$.

The Bit-Entropy Test $BET(K_1, K_2)$ is calculated as follows:

$$BET(K_1, K_2) = \begin{cases} Pass, E_{K_i} \geq 1.0 \wedge (BiEn(K_i) \geq 0.1) \\ Fail, otherwise \end{cases} \qquad 5\text{-}15$$

The $BET(K_1, K_2)$ combines $E_{K_i}$ and $BiEn(K_i)$ to test entropy along with relative and disorder bits of any length in $K_1$ and $K_2$. The $K_1$ and $K_2$ are considered to be a pass if the $E(K_i)$ is greater than the threshold value of 0.1 and $BiEn(K_i)$ is greater than 1.0.

### 5.2.5. Key-schedule evaluation criterion – $KSEC_{RNG}(K_1, K_2)$

The four tests: Frequency-$FT(K_1, K_2)$, Bit-Correlation-$BCT(K_1, K_2)$, Bit-Interfold-$BIT(K_1, K_2)$ and Bit-Entropy-$BET(K_1, K_2)$ together form a test suite – $KSEC_{RNG}(K_1, K_2)$ – to

evaluate the strength of the KSA based on RNG. An RNG-based KSEC, $KSEC_{RNG}(K_1, K_2)$, is illustrated in Table 5.1.

*Table 5.1 Performed Tests with the RNG-based KSEC – $KSEC_{RNG}(K_1, K_2)$ with Key Size $L = 2^N$ with $N \in \{6,7,8\}$*

| Test Type | Number of Keys | Cryptographic Property | Threshold level |
|---|---|---|---|
| **Frequency** | | | |
| Frequency | 500 | Balance of 0 and 1 | $\frac{len(K_i)}{2}$ |
| **Bit-Correlation** | | | |
| Rogers-Tanimoto | 500 | Correlation | 0.5 |
| Pearson Correlation | | | 0.1 |
| **Bit-Interfold** | | | |
| Hamming Distance | 400 | Confusion & Diffusion | 0.7 |
| Z-Proportion | | | |
| **Bit-Entropy** | | | |
| BiEntropy: $BiEn(K_i)$ | 50 | Unpredictability | 0.1 |
| entr: $E(K_i)$ | 500 | | 1.0 |

Table 5.1 shows the required data generation for each test and the corresponding value for the threshold. It also summarizes the cryptographic properties and random keys associated with each test. During the test, a key size column specifies the length of the key in bits. The next chapter describes the RNG-based KSA, data generation and evaluates their strengths.

## 5.1. Summary and conclusion

This chapter discussed CryptoQRNG, a new framework for evaluating a KSA's cryptographic strength. The proposed framework develops a test suite of statistical and hypothesis tests designed to assess the strength of the RNG-based KSA, based on its cryptographic properties. The next chapter will focus on the QRNG and GA-based dynamic S-box, and the algorithms associated with generating it.

# Chapter 6. QuantumGS-box – a key-dependent GA and QRNG-based S-box for high-speed cloud-based storage encryption

## 6.1. Introduction

Cloud computing has become a significant part of the IT landscape in the last few years due to the migration from local drive to cloud computing storage. This has led to data being stored remotely rather than locally and, through an interconnected network, being accessed over the internet. Devices are connected, and data is transmitted using different network protocols. The data travels at high speed [17] in a high-speed network, maximizing network efficiency but depending on the various transmission protocols and resources needed to operate and maintain the network.

Transmission of data over a network requires the consideration of a number of factors, including processing speed and security concerns. Cloud storage networks transmit data through a network of nodes, several devices or a series of devices connected by high-speed links. However, data security is a significant concern as information flows through the internet. Incorporating cryptography [16] that meets secure transmission requirements ensures that the data is transmitted securely and with a high-quality service. Secure data transmission over a network can be achieved by cryptography that uses KSAs [32] and encryption [33]. The S-box does the substitution in both KSA and encryption.

Figure 6.1 shows data stored on a cloud storage system by a user, where Alice is both the sender and receiver. Alice encrypts the data before sending it to the network and receiving the same encrypted data. Encrypting the data before sending it to the network is the most effective way of protecting cloud-based (remote) storage from unauthorized access or attacks.

Figure 6.2 illustrates a cloud-based storage network that uses a static S-box to implement the KSA and encryption. The figure shows encryption with an AES S-box [39]. The AES S-box is a 16 x 16 matrix of static bits. Using the KSA and S-box, Alice generates the key and encrypts the data. Alice encrypts the data before sending it to cloud-based storage (remote) through HSN, and then decrypts it after receiving it. The inverse of an S-box is used to decrypt data.

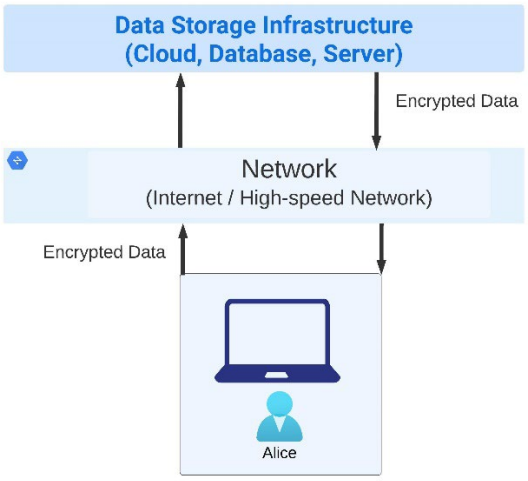



*Figure 6.1 Alice is sender and receiver*

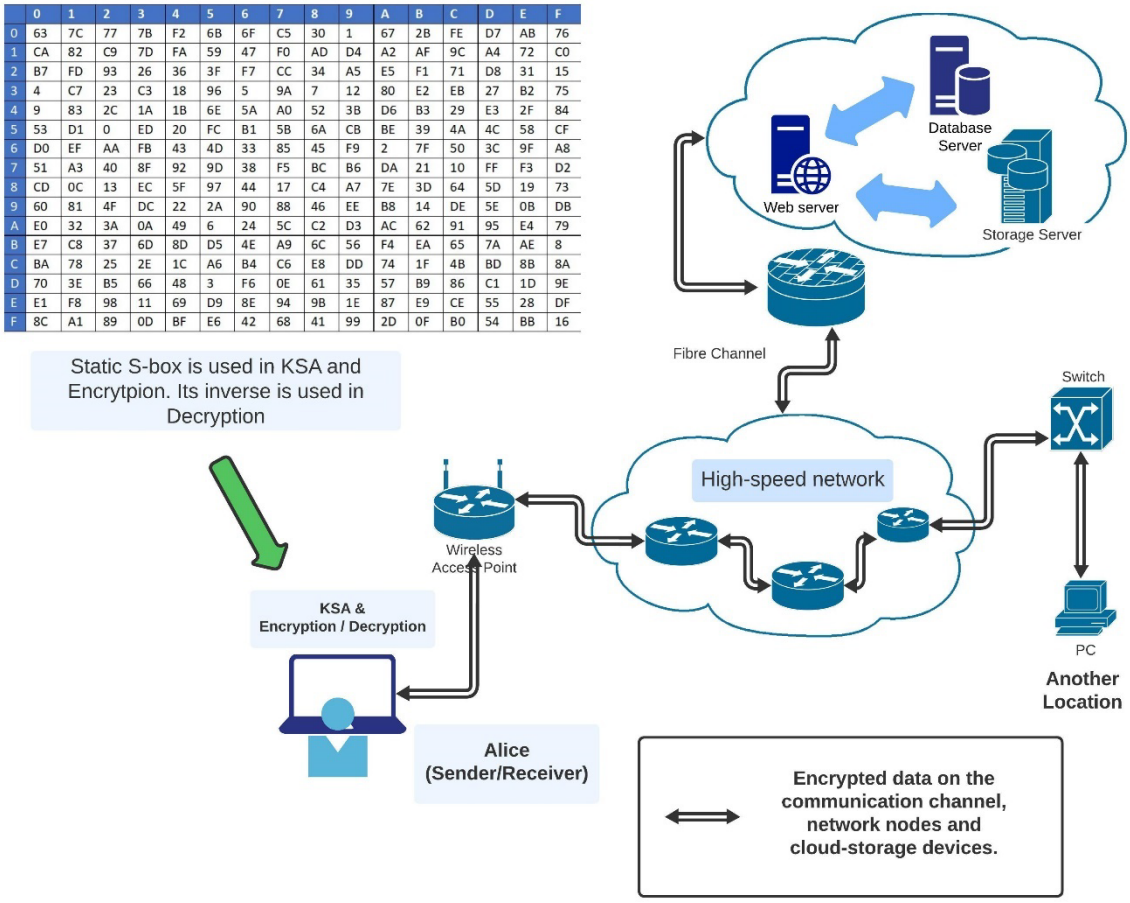
The static S-box is used to provide a substitution of values. The security strength of the system depends on how vulnerable is the key generated by the KSA and its component [7] for the cryptanalysis attack, which reserves the scope of research. In the last two decades, the concept of a dynamic S-box has become the focus area for providing the needed security. The static bits of an S-box make it vulnerable to attack and many researchers have focused on making it dynamic with different scientific approaches. This research focuses on improving the security by making the S-box of the KSA as dynamic using a QRNG and evolutionary computation. The KSA depends on a proposed dynamic S-box based on quantum random bits and the GA, an evolutionary computation to generate the key.

The GA [204] [205] is a well-known algorithm for search-oriented optimization inspired by biological evolution. It provides an analogy for the Darwinian theory of survival of the fittest. The GA dynamically changes the search process to reach an optimal solution based on crossover and mutation probability as a part of different phases of its cycle. The binary GA, as a search-oriented optimization, dynamically searches the solution within the generations of the population based on the binary format. The fitness function is used to evaluate the population that undergoes a crossover or mutation to find the best value.

The proposed work uses the transformation operations of the binary GA to provide efficient big-shuffling for the determination of the values of a dynamic S-box.

Convention AES keys are based on a PRNG [152] [93] to generate random data to enhance AES security. RK-AES [95] proposed a SRFG to achieve randomness in an AES key. However, the highest quality [206] and true random data [26] can be achieved using a QRNG based on the quantum principle.

IDQ invented a USB-based QRNG for generating true (non-pseudo) randomness. Quantis [26] is a state-of-the-art QRNG, exploiting an optical quantum process as the source of randomness.




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 0 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | 3A | 0A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 3 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

*Figure 6.2 AES static S-box in cloud-based storage encryption through high-speed network*

The GA [51] is also a search heuristic inspired by the natural evolution of Charles Darwin's theory. The algorithm corresponds to the natural selection of selecting the fittest individuals based on fitness function for reproduction to produce offspring of the next generation. A GA requires the following two factors of a domain solution: *a genetic representation* and *a fitness function to evaluate suitability.*

Initial population, fitness function, selection, crossover, mutation, and termination are phases of the GA that results in an optimal solution.

The proposed work uses a 128-bit secret key and 128-bits from the QRNG as the initial population and performs crossover and mutation to generate different values of the S-box. The S-box values are dynamic and unique during execution of the proposed method, in contrast to existing solutions.

The focus of this study is to create a key-dependent dynamic S-box based on quantum randomness and a bit-shuffling method. The concrete contributions of this work are:

- Designing an algorithm that takes a user-defined key and quantum random bits from a QRNG and generates a dynamic S-box using bit-shuffling with genetic evolution.
- Analysing the proposed algorithm corresponding to a range of cryptographic properties, including nonlinearity, randomness, linear and differential probabilities, bit-independence criteria, and balance.

The proposed method of generating a dynamic S-box includes a QRNG and bit-shuffling based on the operations of a GA to make the mathematical calculation simple, while at the same time ensuring that the values generated by this method are cryptographically strong and secure. The proposed QuantumGS-box values are generated dynamically with the user-defined key and quantum random bits. The design principle is discussed first, followed by the proposed methodology.

## 6.2. The design principle

The design principle of the proposed work uses algebraic techniques, quantum randomness, and optimization techniques to generate a dynamic, high-quality S-box. There are a variety of cryptographic properties associated with high quality. These properties include nonlinearity, balance, correlation immunity, algebraic degree, and differential and linear approximation probabilities. A diagram illustrating the design principles is shown in Figure 6.3. The RNGs-based S-box effectively affect the randomness of bits in the S-box. The best RNGs generate more random and unpredictable data to make keys resilient against a cryptanalysis attack. Randomness is generated using an entropy source, which can be either pseudo- or quantum-based. The PRNG- [170] and TRNG- [165] based S-boxes are proposed by some researchers affecting the cryptographic properties. However, the QRNG [25] ensures high entropy using quantum states of light from a quantum origin. A QRNG is superior to a traditional RNG because its source of quantum randomness (QR) is invulnerable to environmental perturbations such as temperature, voltage, or current and are provably secure RNGs [207][208].

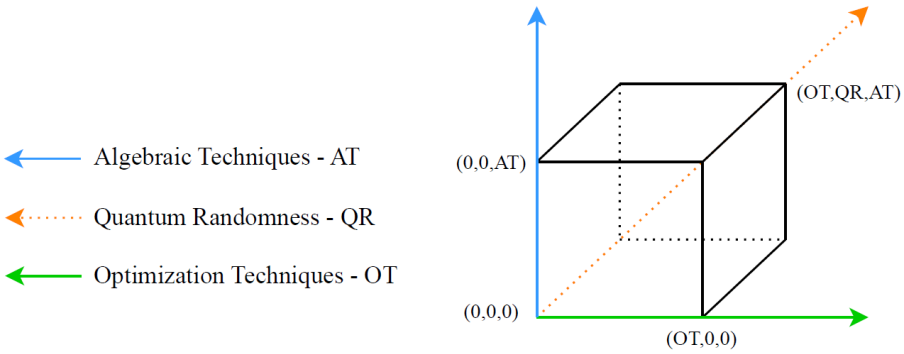



*Figure 6.3 Design principle of the proposed work*

Algebraic techniques (AT) are used to achieve various cryptographic properties in generating the S-box. These methods formulate cryptographic property and possess significant cryptographic qualities. However, with the advancement of cryptanalysis [209][210] over the last two decades, these methods are becoming more vulnerable to exploitation.

Another important criterion in the design of S-box dynamic bits is the speed at which the bits are generated. There needs to be an increase in the speed at which the S-box is constructed to make it more difficult to access the bits in the S-box. Dynamic S-boxes are the basis for generating the key and encrypting the data using that key. The higher speed of generating dynamic S-box values also contributes to the high-speed encryption of the data. Increasing the generation speed of dynamic S-box values reduces encryption time and data travel time on communication channels and data storage facilities.

The last criterion is the optimal values generated by optimization techniques (OT) based on all three above-mentioned criteria, which can be achieved through evolutionary computation (EC). There are various studies [211][212][213] by researchers who created dynamic S-boxes based on EC and optimized the most effective solution for creating S-box values.

## 6.3. Proposed methodology

Figure 6.4 shows the dynamic proposed S-box, a QuantumGS-box for the KSA and for a cloud-based storage encryption. The figure shows the encryption with the new QuantumGS-box. The QuantumGS-box is a 16 x 16 matrix consisting of dynamic bits. The user will generate the key with the KSA along with the S-box and encrypt the information. Alice is both sender and receiver in this scenario, encrypts the data before sending it to the high-speed network, and decrypts it once it is returned.

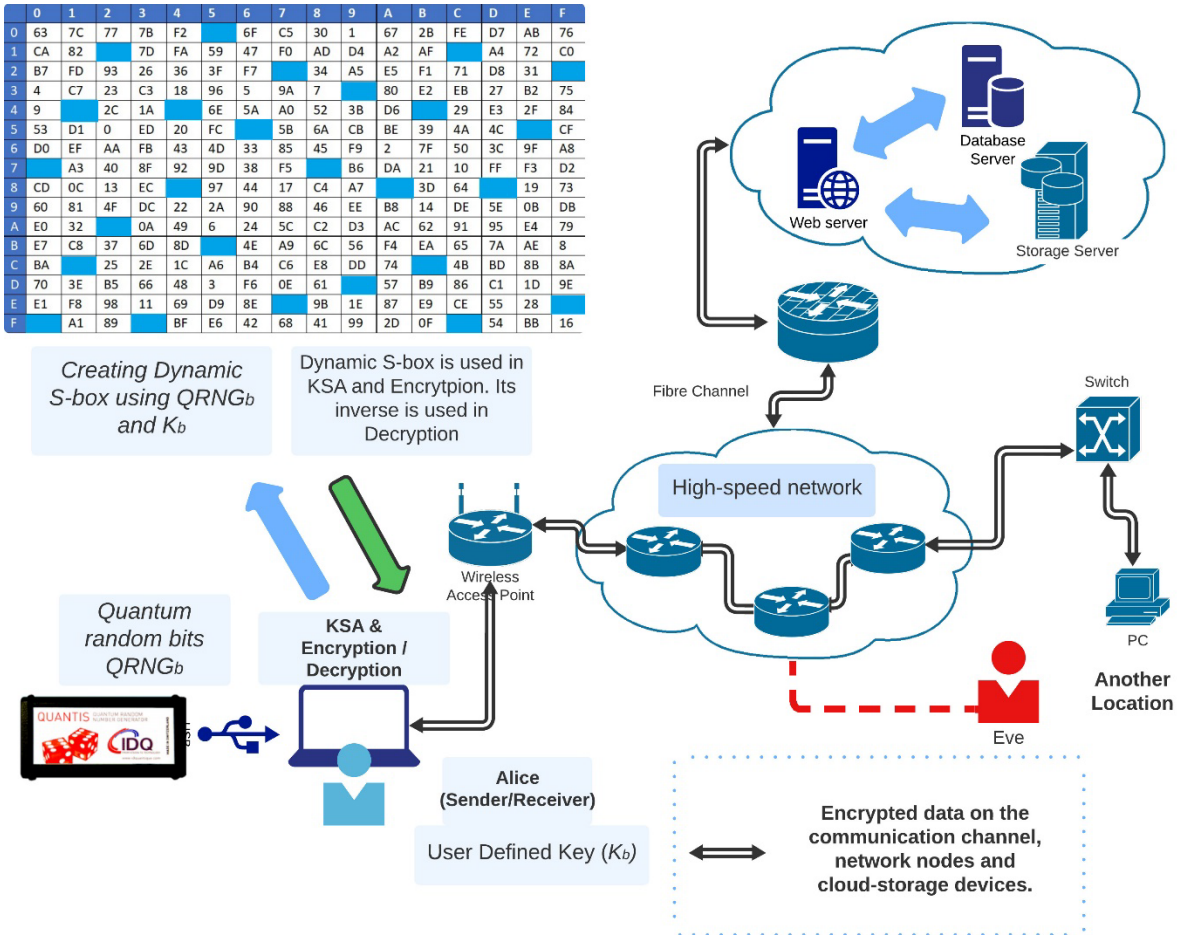



*Figure 6.4 Proposed QuantumGS-box in cloud-based storage encryption through high-speed network*

The QuantumGS-box provides a matrix of substitution values based on a QRNG and user-defined key for the KSA and on encrypting the data. Alice will send the encrypted information over a communication channel to the cloud. The encrypted format makes the data secure before it travels through various HSN nodes. This ensures that the data is protected from Eve, a potential malicious attacker, who could attempt to intercept it while it is in transit. Encrypted data is stored on multiple servers depending on their cloud application. The same encrypted data travels back over the network when Alice wants to access the data. Overall, this process ensures that Alice's data remains secure and confidential during data transmission to the cloud.

Figures 6.5 and 6.6 show how the proposed algorithm using the QRNG will be applied to the AES and key expansion along with EC. The AES key-schedule includes AES key generation and expansion. AES 128 generates a random key by 16 bytes, constituting $W_0$, $W_1$, $W_2$, and $W_3$ that incorporates QRNG bits from Quantis. The key expansion will expand these four words to 44 with g function and XOR operation. The total number of words required

depends on the number of round keys. AES 128, 192, and 256 required 11, 13, and 15 Round Keys, respectively. The total number of words needed is [$W_0$, $W_1$, $W_2$…….$W_{4R-1}$], where R is the number of round keys required.

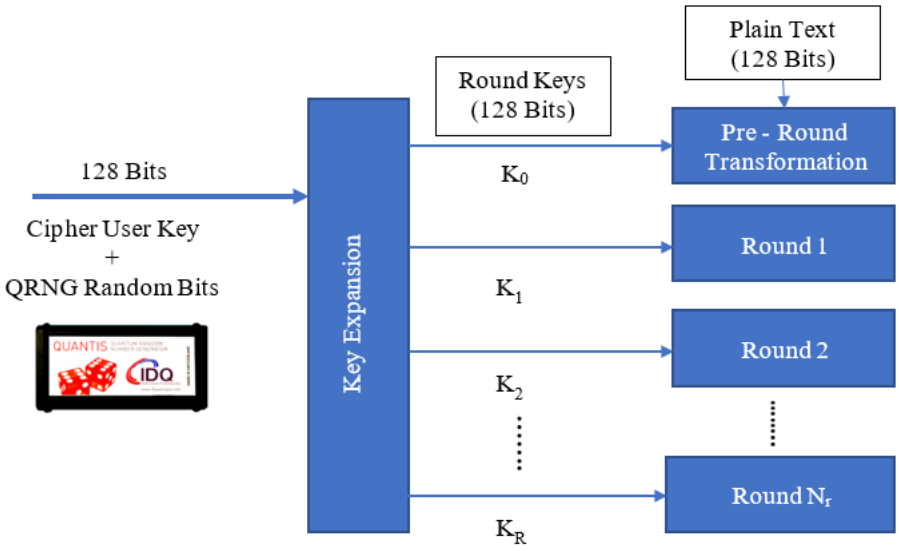



*Figure 6.5 QRNG applied to key expansion of AES*

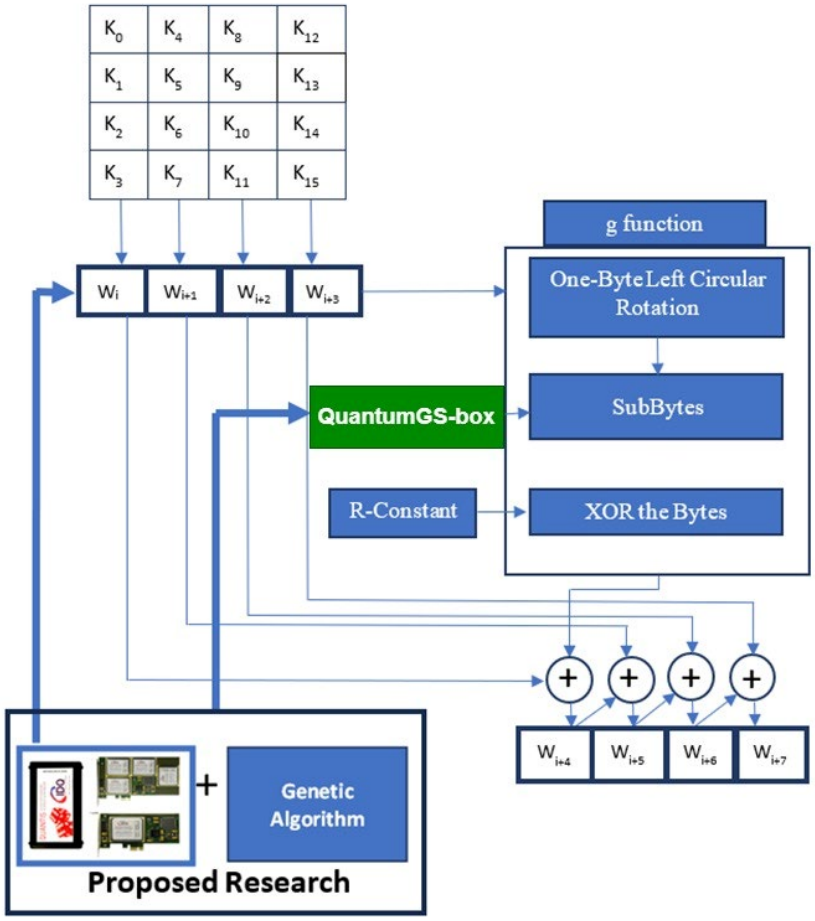



*Figure 6.6 Proposed research (QuantunGS-box) for AES key generation & expansion*

### 6.3.1. Step-by-step procedure and algorithm to generate QuantumGS-box

This section illustrates the mathematical calculation and the bit-shuffling based on the operations of a GA of the proposed method to generate dynamic S-box values. The step-by-step operation of the proposed method is shown with a control flow diagram in Figure 6.7.

The flowchart starts with taking a user key from the user and then random bits from a QRNG. The GA will process the QRNG and user key bits as the initial population, and the number of rounds as the generation, and will generate the binary string as chromosomes. A fitness function is used to select the best chromosomes in order to find the index to generate the values of the QuantumGS-box ($QGSbox$).

***Step1 (S1):*** Initially, the user inputs a 16-character key, denoted as $UKey$

***Step2 (S2):*** The QRNG generates the 128 quantum random bits, denoted as $QRNG(128)$

***Step3 (S3*)**: The user key UKey is converted into bits and denoted as $Key_b$

***Step4 (S4):*** Initial population generation ($P_{init}$)

$QRNG(128)$ - Quantum random bits from QRNG (128 bits)

$Key_b$ - User-defined key (128 bits)

$$P = P \cup (QRNG(128) \oplus Key_b) \quad 6\text{-}1$$

***Step5 (S5):*** This step gives the GA the function name optimize_GA and a parameter initial population ($P_{init}$), calculated at S4. Algorithm 5-2 shows the pseudo-code of the proposed algorithm for optimize_GA. Section 6.3.2 focuses on optimize_GA.

***Step 6 (S6):*** This step checks each element in the $QGSbox$ with row m and column n. The decision depends on the positions that have NaN ($\perp$) as their value.

The following steps will calculate the weight of that position, where there is a NaN (Not a Number) in the $QGSbox$. If there is no NaN, this will proceed to stop.

***Step 7 (S7) to Step 9 (S9):*** These steps will continue until $QGSbox$ has NaN ($\perp$).

The quantum random number bits will be generated from QRNG and as $QRNG(255)$. The value of temporary weight denoted as $weight$ is calculated by XORing the quantum random bits and user-defined key bits.

$$weight = QRNG(255) \oplus Key_b \quad 6\text{-}2$$

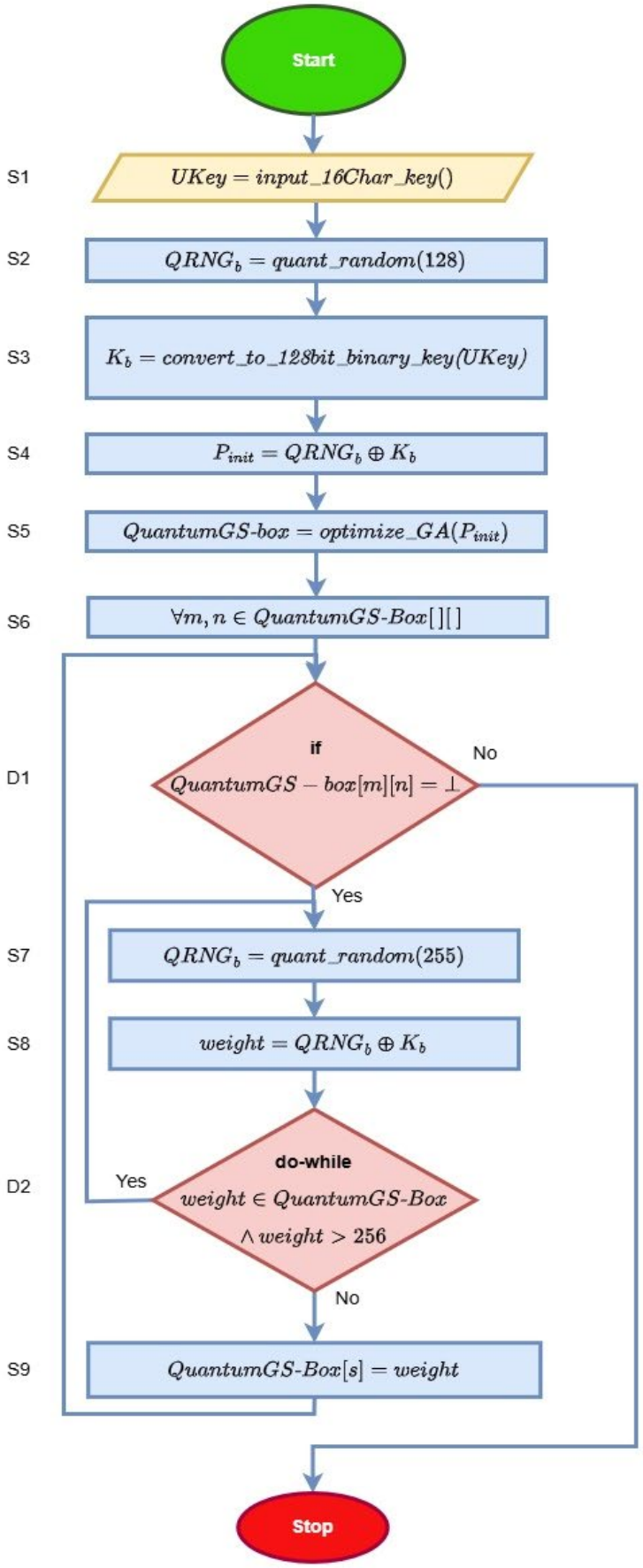



*Figure 6.7 Flowchart of the proposed QuantumGS-box*

The next block will check whether the value for $weight$ has already been stored in $QGSbox$, and if it is greater than 256. If it is, the value will be regenerated; if it is not, the value will be stored in the $QGSbox$.

When the algorithm reaches the stop, all $QGSbox$ values have been calculated and the algorithm terminates. Algorithm 6-1 shows the pseudo-code of the proposed algorithm.

*Algorithm 6-1 Algorithm to generate QuantumGS-box*

Algorithm to generate QuantumGS-box

**INPUT:** $UKey$ *// 16-character textual key provided by user*

**OUTPUT:** $QGSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$ *// QuantumGS-box*

```
1  begin
2      K_b = convert_to_128bit_binary_key( UKey )
3      P_init = ∅ // initial population
4      ∀i ∈ {1..PSIZE}. P = P ∪ (QRNG(128) ⊕ Key_b) // fill initial population
5      QGSbox = optimize_GA (P_init)
6      ∀m,n ∈ QGSBox[ ][ ]
7      If QGSbox[m][n] = ⊥
8          do
9              weight = QRNG(255) ⊕ Key_b
10         while weight ∈ QGSBox ∧ weight > 256
11         QGSbox[s] = weight
12 end
```

Note: ⊥– *Not a Number (NaN)*

### 6.3.2. Algorithm $optimize\_GA(P_{init})$:

Algorithm 6-2 shows the steps to calculate the positions for weights of $QGSbox$.

Initialize the generation maximum iteration of generation denoted by $RMAX$. This step will continue to execute from generation 1 to $RMAX$.

In a population of chromosomes, $fitness\_GA(C)$ computes the fitness of each chromosome. The fitness value that is highest across all chromosomes will be considered the most appropriate and its chromosome is denoted as $C_{fittest}$. Algorithm 5-6 focuses on $fitness\_GA(C)$ and how to calculate each fitness value.

This highest value chromosome $C_{fittest}$ tells the positions for the weights of $QGSbox$. Four middle bits are chosen together with four of the most significant bits to calculate the row and column index. The index positions $x$ and $y$ represent row and column positions. For

example, the 60, 61, 62 and 63 bits of $C_{fittest}$ are 0110 and 124, 125, 127, and 128 bits are 0100, so the $x = 0110$ in binary and $x = 6$ in hexadecimal. Similarly, $y = 0100$ and $y = 4$. These positions in $QGSbox$ as $QGSbox$ [4][6] and $QGSbox$ [6][4] will now have NaN.

The next step will calculate the next population ($P_{tmp}$) using crossover and mutation for the next iteration of the generation.

*Algorithm 6-2 Algorithm* $optimize\_GA(P_{init})$*: Calculate positions for weights of QuantumGS-box*

Algorithm to calculate positions for weights of QuantumGS-box

**INPUT:** $P_{init}$ *// initial population (with PSIZE chromosomes)*

$$SSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$$ *// static S-Box*

$RMAX$ *// max number of S-Box weights to replace:* $10 \leq RMAX \leq 250$

**OUTPUT:** $QGSbox_{tmp}$ *// interim QuantumGS-box*

1 ***begin:***
2 $P_{tmp} = P_{init}$ *// initialize current population with initial population*
3 $QGSbox_{tmp} = SSbox$ *// initialize output QuantumGS-box*
4 ***for*** $gen\_cnt$ ***is*** $1$ ***to*** $RMAX$ *// iterate RMAX times*
5 $C_{fittest} = \underset{C \in P_{tmp}}{\text{argmax}}\ fitness\_GA(C)$ *// chromosome with max fitness value*
6 $i = C_{fittest}[60], j = C_{fittest}[61], k = C_{fittest}[62], l = C_{fittest}[63]$
7 $p = C_{fittest}[124], q = C_{fittest}[125], r = C_{fittest}[126], s = C_{fittest}[127]$
8 $x = bin\_to\_hex(i, j, k, l)$ *//* $i \cdot 2^3 + j \cdot 2^2 + k \cdot 2^1 + l \cdot 2^0$
9 $y = bin\_to\_hex(p, q, r, s)$ *//* $p \cdot 2^3 + q \cdot 2^2 + r \cdot 2^1 + s \cdot 2^0$
10 $QGSbox_{tmp}[x][y] = \perp$
11 $QGSbox_{tmp}[y][x] = \perp$
12 $P_{tmp} = mutate(crossover(P_{tmp}))$ *// population of next generation*
13 *return* $QuantumGSbox_{tmp}$
14 ***end***

*Crossover:* $crossover(P_{tmp})$

This research study focuses on the crossover operation of a GA [214]. The chromosomes of each generation are combined via the crossover function to produce the chromosomes of the next generation. A one-point crossover has been used. In order to create new offspring, a crossover point is chosen at random and the tails of ($C_1$ *and* $C_2$) are swapped to create a new offspring. The crossover chromosome is calculated as follows:

$$crossover(P_{tmp}) = f(P_{tmp}) \qquad 6\text{-}3$$

where,

$crossover(P_{tmp})$ - Population chromosome after crossover

$n$ – a random number crossover point from QRNG.

Algorithm 6-3 shows the $crossover(P_{tmp})$ function to calculate the crossover for the current population. Algorithm 6-4 $\text{crossover}_C(C_1, C_2)$ shows the 1-point crossover with a quantum random number.

*Algorithm 6-3 Algorithm $crossover(P_{tmp})$: Calculate crossover for the current population*

| Algorithm to calculate crossover for the current population | |
|---|---|
| | **INPUT:** $P_{tmp}$ *// initial population (with PSIZE chromosomes)* |
| | **OUTPUT:** $P_{new}$ *// interim population for crossover* |
| 1 | ***begin:*** |
| 2 | $P_{new} = \emptyset$ *// initialize new population* |
| 3 | while$(\text{P}_{\text{tmp}} \neq \emptyset)$ |
| 4 | $C_1, C_2 = remove(P_{tmp})$ *// Two chromosomes from current population* |
| 5 | $P_{new} = P_{new} \cup \text{crossover}_C(C_1, C_2)$ |
| 6 | *return* $P_{new}$ |
| 7 | ***end*** |

*Algorithm 6-4 Algorithm $\text{crossover}_C(C_1, C_2)$: Calculate 1-point crossover.*

| Algorithm to calculate 1-point crossover | |
|---|---|
| | **INPUT:** $C_1, C_2$ *// two chromosomes from the current population* |
| | **OUTPUT:** $\text{C}_1'$ , $\text{C}_2'$ *// new crossover chromosomes* |
| 1 | ***begin:*** |
| 2 | $p = QRNG(0..127)$ *// a quantum random number from QRNG within the range* |
| 3 | $\text{C}_1' = \text{C}_1[0..(\text{p}-1)] + \text{C}_2[(\text{p})..127]$ |
| 4 | $\text{C}_2' = \text{C}_2[0..(\text{p}-1)] + \text{C}_1[(\text{p})..127]$ |
| 5 | *return* $(\text{C}_1', \text{C}_2')$ |
| 6 | ***end*** |

*Mutation:* $mutate(P_{tmp})$

This research focuses on the mutation operation of a GA [214]. In the GA, mutations can be used to maintain diversity among the chromosomes in a population. A mutation has not been incorporated into the population as a whole and is measured by a mutation probability that is assigned to each individual according to their fitness value. The mutation probability in the proposed work is 0.95. A random number is generated between 0 and 1. If it is greater than probability, the values will be generated by QRNG, otherwise not.

The mutation chromosome is calculated as follows:

$$mutate\_c(C) = \begin{cases} QRNG(128), Q_{rn}(0,1) > k \mid k = 0.95 \\ C, otherwise \end{cases} \quad 6\text{-}4$$

where,

$mutate\_c(C)$ - Population chromosome after mutation

C– Chromosome of initial population ($P_{init}$)

$Q_{rn}(n_1, n_2)$ - a random number from QRNG within the range $n_1 \leq Q_{rn}(n_1, n_2) \leq n_2$

The process will proceed to RMAX generation and the population for the next iteration will be as follows:

$$P_{tmp} = mutate(crossover(P_{tmp})) \qquad 6\text{-}5$$

Algorithm 6-5 shows the $mutate\_c(C)$ function to calculate the mutation for the current population.

*Algorithm 6-5 Algorithm $mutate\_c(P_{tmp})$: Calculate mutation for the current population*

| Algorithm to calculate mutation for the current population | |
|---|---|
| | **INPUT:** $P_{tmp}$ *// initial population (with PSIZE chromosomes)* |
| | **OUTPUT:** $P_{new}$ *// interim population for mutation* |
| 1 | ***begin:*** |
| 2 | $P_{new} = \emptyset$ *// initialize new population* |
| 3 | while$(\mathrm{P_{tmp}} \neq \emptyset)$ |
| 4 | $\mathrm{C} = \mathrm{remove}(P_{tmp})$ *// chromosome from current population* |
| 5 | $P_{new} = P_{new} \cup \mathrm{mutate}_C(C)$ |
| 6 | *return* $P_{new}$ |
| 7 | ***end*** |

### 6.3.3. Algorithm $fitness\_GA(C)$:

Algorithm 6-6 shows the pseudo-code of the proposed algorithm for $fitness\_GA(C)$ that calculates the fitness value of each chromosome.

*Algorithm 6-6 Algorithm $fitness\_GA(C)$: Calculate fitness value for a given chromosome (128-bit sequence)*

| Algorithm to calculate fitness value for a given chromosome (128-bit sequence) | |
|---|---|
| | **INPUT:** $C$ *// chromosome (128-bit sequence)* |
| | **OUTPUT:** $Fitness$ *// fitness value* |
| 1 | ***begin:*** |
| 2 | $R = C[0 \dots 63]$ *// least significant 64 bits of C* |
| 3 | $L = C[64 \dots 127]$ *// most significant 64 bits of C* |
| 4 | $H_{dist} = Hamming\_distance(R, L)$ |
| 5 | $JW_{dist} = Jaro\_Winkler\_distance\ (R, L)$ |
| 6 | $L_{dist} = Levenshtein\_distance\ (R, L)$ |
| 7 | $Fitness = H_{dist} + JW_{dist} + L_{dist}$ |
| 8 | *return* $Fitness$ |
| 9 | ***end*** |

The current chromosome splits into two parts.

*Chromosome:* (Split the $C$ into two halves $R$ and $L$)

$$R = C[0 \dots 63] \quad 6\text{-}6$$

$$L = C[64 \dots 127] \quad 6\text{-}7$$

Different distance values will be calculated using these two parts, leading to the fitness value, $Fitness$. The highest fitness value is the optimum to optimize chromosomes.

### 6.3.4. Calculation of distance values: $H_{dist}, JW_{dist}, L_{dist}$

$Fitness$ calculates the fitness function for QuantumGS-box as follows:

$$Fitness = (H_{dist} + JW_{dist} + L_{dist}) \quad 6\text{-}8$$

where,

$H_{dist}$ - Hamming Distance between $R$ and $L$: $H_{dist}(R, L)$

$JW_{dist}$ - Jaro–Winkler Distance between $R$ and $L$: $JW_{dist}(R, L)$

$L_{dist}$ - Levenshtein Distance between $R$ and $L$: $L_{dist}(R, L)$

*Hamming Distance* ($\mathrm{H_{dist}}$) :

$\mathrm{H_{dist}}(BS_1, BS_2)$, the hamming distance [200] between two bit sequences $BS_1$ and $BS_2$ is calculated as the number of bit positions $BS_{1,j}$ in $BS_1$ that are different to those $BS_{2,j}$ in $BS_2$, divided by the chromosome length:

$$H_{dist}(BS_1, BS_2) = \frac{\left|\left\{BS_{1,j} \mid BS_{1,j} \neq BS_{2,j} \wedge \left(0 \leq j < len(BS_1)\right)\right\}\right|}{len(BS_1)} \quad 6\text{-}9$$

*Jaro–Winkler Distance* ($\mathrm{JW_{dist}}$) :

The first step in calculating $\mathrm{JW_{dist}}(BS_1, BS_2)$, the Jaro–Winkler Distance between $BS_1$ and $BS_2$, is to obtain the Jaro similarity $\mathrm{JS}(BS_1, BS_2)$, a score between $BS_1$ and $BS_2$.

The $JS(BS_1, BS_2)$ score is 0 if the strings do not match at all, and 1 if they are an exact match. A $JS$ is calculated as follows:

$$JS(BS_1, BS_2) = \begin{cases} 0, if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|BS_1|} + \frac{m}{|BS_2|} + \frac{m-t}{m}\right), otherwise \end{cases} \quad 6\text{-}10$$

where,

$|BS_i|$ - is the length of the bitstring $BS_i$

$m$ - the number of matching bits

$t$ - the number of transpositions

The Jaro–Winkler similarity ($JWS$) uses a con for a more specific similarity with a defined length $len$ and is calculated as follows:

$$JWS(BS_1, BS_2) = JS(BS_1, BS_2) + len \times con\big(1 - JS(BS_1, BS_2)\big) \quad 6\text{-}11$$

where,

$JS$ is the Jaro similarity between bitstring $BS_1$ and $BS_2$

$len$ – the length of common prefix at the start of the string with up to a maximum of four characters

The final $JW_{dist}$ is

$$JW_{dist}(BS_1, BS_2) = 1 - JWS(BS_1, BS_2) \quad 6\text{-}12$$

*Levenshtein Distance* ($L_{dist}$)

The Levenshtein distance $L_{dist}\ (BS_1\ , BS_2)$ between two bitstring $BS_1$ and $BS_2$ is calculated as follows

$$L_{dist}(BS_1, BS_2) = \begin{cases} |BS_1|\ if\ |BS_2| = 0, \\ |BS_2|\ if\ |BS_1| = 0, \\ L_{dist}(tail(BS_1), tail(BS_2))\ if\ BS_1[0] = BS_2[0], \\ 1 + min\begin{cases} L_{dist}(tail(BS_1), BS_2) \\ L_{dist}(BS_1, tail(BS_2)) \\ L_{dist}(tail(BS_1), tail(BS_2)) \end{cases} \quad Otherwise \end{cases} \quad 6\text{-}13$$

The proposed algorithm generates different QuantumGS-boxes based on different generations and keys. Table 6.1 shows the QuantumGS-box with the 75 generation and the user-defined key – “The QuantumGS-box” . The following places are different with static S-box generated [0F], [1E], [25], [2C], [4B], [4F], [5E], [6D], [6F], [B4], [C2], [CF], [D2], [D6], [DE], [E1], [ED], [F0], [F4], [F6], and [FC].

*Table 6.1 QuantumGS-box generated by 75-th generation of the GA*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 25 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | F6 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | D9 | 8D | 31 | 15 |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | 72 | 29 | E3 | 2F | 8A |
| 5 | 53 | D1 | 1D | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | B3 | 9F | 42 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | B | DB |
| A | E0 | 32 | 3A | A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 84 | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | 78 | F8 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 3C |
| D | 70 | 3E | B5 | 66 | 48 | 3 | D8 | E | 61 | 35 | 57 | B9 | 86 | C1 | B0 | 9E |
| E | E1 | BF | 98 | 11 | 69 | 76 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 8C | 28 | DF |
| F | A8 | A1 | 89 | D | 71 | E6 | 55 | 68 | 41 | 99 | 2D | F | 0 | 54 | BB | 16 |

## 6.4. Summary and conclusion

This chapter focused on design principle, methodology, and pseudo-code, along with flow chart, explaining the step-by-step process of generating a QuantumGS-box. At the end it showed a QuantumGS-box generated by the use key – the QuantumGS-box. The next chapter is divided into parts. The first part will focus on the experimental details and analysis of the CryptoQRNG and the second part will discuss the comparison of QuantumGS-box with the existing dynamic S-box.

# Chapter 7. Experimental evaluation of CryptoQNRG and QuantumGSbox

This chapter focuses on the experimental evaluation of the proposed work. Initially, it evaluates CryptoQRNG. It then assesses QuantumGSbox.

## 7.1. The RNG-based KSAs tested with CryptoQNRG

This section focuses on the generation of different RNG-based KSAs and their evaluation based on the proposed criterion of CryptoQRNG.

### 7.1.1. Data collection

This study analysed the KSA of the following five block ciphers from a symmetric cryptosystem for an experimental purpose: AES, DES, CAST, Camellia, and GOST. The block cipher encrypts data in blocks of specified key size. The KSA key size taken is 128 bits for AES, Camellia, and CAST, 64 Bits for DES, and 256 bits for GOST. The subkeys are extended using the key expansion function of the KSA. These ciphers are proposed by different authors and with different KSA key size.

Rijndael [39] proposed an AES with three variants AES-128, 192, and 256, based on KSA key length sizes of 128, 192, and 256 respectively, all of which were approved by NIST. Endre Bangerter et al. [40] were able to recover AES-128 encryption keys in 2010. The second block cipher, DES [56], with a KSA key size of 64 bits, is based on the Balanced Feistel structure and was proposed by IBM. Biham and Shamir [146], who suggested a differential cryptanalysis attack on complete rounds of DES.

CAST [71] is based on the Feistel Network (FN) with a KSA key size of 40 to 128 bits. The fourth cipher, Camellia [81], was developed by Mitsubishi Electric and NTT of Japan, based on the FN algorithm and with a KSA key size of 128, 192, or 256 bits. The final cipher, GOST [84], supports KSA key sizes of 256 bits. Nicolas Courtois et al. [215] proposed a Contradiction Immunity to attack the complete 32 rounds of the GOST cipher in 2011.

The RNG-based key-schedule also depends on different entropy based on its generator. In the set-up, CSPRNG and QRNG are taken to evaluate the cryptographic strength. The entropy within the system is used to provide pseudo random bits for the key-schedule ($KSA_{PK}$) that is required to create the keys. In QRNG, photons are used to generate quantum random bits for the key-schedule ($KSA_{QK}$). The bits length depends on the key size of the KSA. The subkeys are generated using the cryptol [185] language. The results will demonstrate the

strength of the KSA in terms of cryptographic parameters for each block cipher. The same notation is used for result analyses:

$K_1$ ... the subkey obtained with the KSA based on QRNG

$K_2$ ... the subkey obtained with the KSA based on PRNG.

To generate subkey $K_1$ of size $L$ the bitstream of the QRNG is taken and combined with the user-defined key (Key) with multiple KSA iterations (rounds):

$$K_1 = KSA(Key, bs_{QRNG}, rounds, L)$$

Analogously, to generate subkey $K_2$ of size $L$ the bitstream of the PRNG is taken and combined with the user-defined key (Key) with multiple KSA iterations (rounds):

$$K_2 = KSA(Key, bs_{PRNG}, rounds, L)$$

The experiment uses a KSA with 11 iterations ($rounds$) and subkey-size $L = 2^N$ with $N \in \{6,7,8\}$:

$$K_1 = KSA(Key, bs_{QRNG}, 11, 2^N) \mid N \in \{6,7,8\}$$

$$K_2 = KSA(Key, bs_{PRNG}, 11, 2^N) \mid N \in \{6,7,8\}$$

where the subkey-size $L = 2^N$ depends on the concrete block cipher length. So for DES there is $N = 6$ ($L = 2^6 = 64$), for AES, Camellia, and CAST, $N = 7$ ($L = 2^7 = 128$), and for GOST, $N = 8$ ($L = 2^8 = 256$).

In this study, two sets of data are used, one for the Frequency and Bit-Entropy test, where random subkeys are used, and another set for Bit-Correlation and Bit-Interfold, where samples of subkeys are used to test the hypotheses.

### 7.1.2. Results and analysis

Finally, the KSEC $KSEC_{RNG}(K_1, K_2)$ is used to evaluate the cryptographic strength of $K_1$ and $K_2$.

#### *7.1.2.1. Frequency Test*

Table 7.1 displays the results of the frequency test. The number of bits is balanced in both $K_1$ and $K_2$. The numbers of bits in the Quantum $K_1$ and Pseudo-based $K_2$ key-schedule integrate the equal number of 0s and 1s. The table shows that the $FT(K_1, K_2)$ of each block cipher passes the frequency test. However, the frequency test alone cannot predict the strength of the RNG-based key-schedule.

*Table 7.1 Frequency analysis $FT(K_1, K_2)$ of bits in $K_1$ and $K_2$ schedule of five block ciphers.*

| | AES | Camellia | CAST | DES | GOST |
|---|---|---|---|---|---|
| Ratio of Percentage of 0:1 in $K_1$ | 50:50 | 50:50 | 50:50 | 50:50 | 50:50 |
| Ratio of Percentage of 0:1 in $K_2$ | 50:50 | 50:50 | 50:50 | 50:50 | 50:50 |

*The Number of 0s and 1s is compared in $K_1$ and $K_2$ schedules of five different block ciphers. The statistical analysis shows the bits are balanced in $K_1$ and $K_2$ for all the ciphers.*

#### *7.1.2.2. Bit-Correlation Test*

Bit-correlation tests the strength in terms of the correlation, while taking the Pearson's correlation hypothesis test in conjunction with the Rogers-Tanimoto distance measure. The graph in Figure 7.1 shows the changes in the Rogers-Tanimoto distance measure in $K_1$ and $K_2$. The results show that the dissimilarity index of CAST is the lowest of all, whereas the GOST key-schedule shows the highest. The index of DES is slightly less then Camellia and AES with 0.66589.

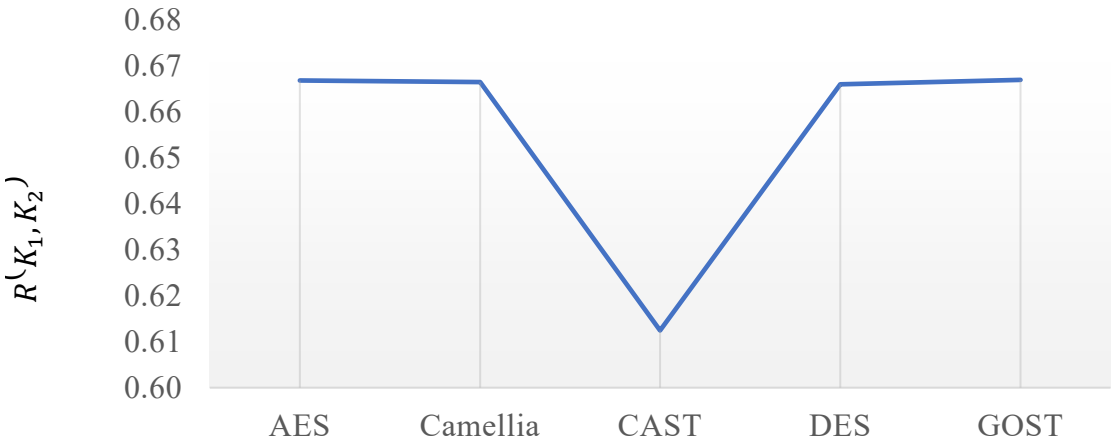



*Figure 7.1 Rogers-Tanimoto distance measure $R(K_1, K_2)$ of five block ciphers.*
*The dissimilarity index of $K_1$ and $K_2$ is compared for five different block ciphers. Statistical analysis shows the bits are nearly 30% similar in both the key-schedule for four ciphers and 40% in CAST; among all the ciphers.*

Table 7.2 shows the Pearson's Correlation hypothesis testing result of each block cipher. The values of the $r_{K_1,K_2}$ correspond to P-value statistics. The P-value of AES, Camellia, DES, and GOST subkeys passes the threshold value of 0.1; therefore, this study rejects the null hypothesis that the $K_1$ and $K_2$ key-schedules are dependent on each other for these ciphers. On the other hand, CAST subkeys failed to pass the threshold value. This means the $K_1$ and $K_2$ are

dependent on each other and do not pass the Bit-Correlation – $BCT(K_1, K_2)$ test. This shows that CAST keys are weak and susceptible for key-dependent [145] and correlation [216] attacks, with both quantum and pseudo random number-based key-schedules.

*Table 7.2 Correlation of bits of $K_1$ and $K_2$ of five block ciphers based on Pearson's Correlation Hypothesis Test.*

| | **AES** | **Camellia** | **CAST** | **DES** | **GOST** |
|---|---|---|---|---|---|
| $r_{K_1,K_2}$ | 0.979 | 0.889 | 0.076 | 0.745 | 0.808 |
| $H_0$ *or* $H_A$ | Independent | Independent | Dependent | Independent | Independent |

### *7.1.2.3. Bit-Interfold Test*

Table 7.3 shows the results of the Bit-Interfold test. The test first calculates the Hamming Distance of the $K_1$ and $K_2$ based on a 64, 128, and 256-bit key size with a sample of 400 keys. The result of the Hamming Distance measures dissimilarity between the $K_1$ and $K_2$, and the inverse of that is then passed to Z-Proportions hypothesis testing and the corresponding P-values are calculated.

Camellia and CAST result in the alternative hypothesis-$H_A$ (taken from below result Table 3), which means they pass the Bit-interfold test. Any KSA that fails this test will create a weak cipher that is vulnerable to an easy cryptoanalysis. For example, AES, DES, and GOST failed the $BIT(K_1, K_2)$ test and showed that they are weak and vulnerable to attacks such as related-key and side-channel [217] attacks.

*Table 7.3 Confusion and Diffusion of $K_1$ and $K_2$ of five block ciphers based on Hamming Distance and Z - Proportion Hypothesis Test*

| | **AES** | **Camellia** | **CAST** | **DES** | **GOST** |
|---|---|---|---|---|---|
| H($K_1, K_2$) | 27661 | 28502 | 31676 | 25623 | 27461 |
| $Z(K_1, K_2)$ | $H_0$ | $H_A$ | $H_A$ | $H_0$ | $H_0$ |

### *7.1.2.4. Bit-Entropy Test*

The most critical parameter for a key in order to be secure is unpredictability. The $K_1$ and $K_2$ are tested with two different entropy tests. The $K_1$ shows better entropy than the $K_2$, as shown in Figure 7.2 and Figure 7.3. The variations in entropy result in different values for the *entr* and Bi-Entropy tests, one for each $K_1$ and $K_2$. These were analysed against the threshold value. The

distinct entropy values of each block cipher exceed the threshold value of 1.0 for *entr* and 0.1 for Bi-Entropy; hence, all the $K_1$ and $K_2$ passed the entropy test – $BEnT(K_1, K_2)$.

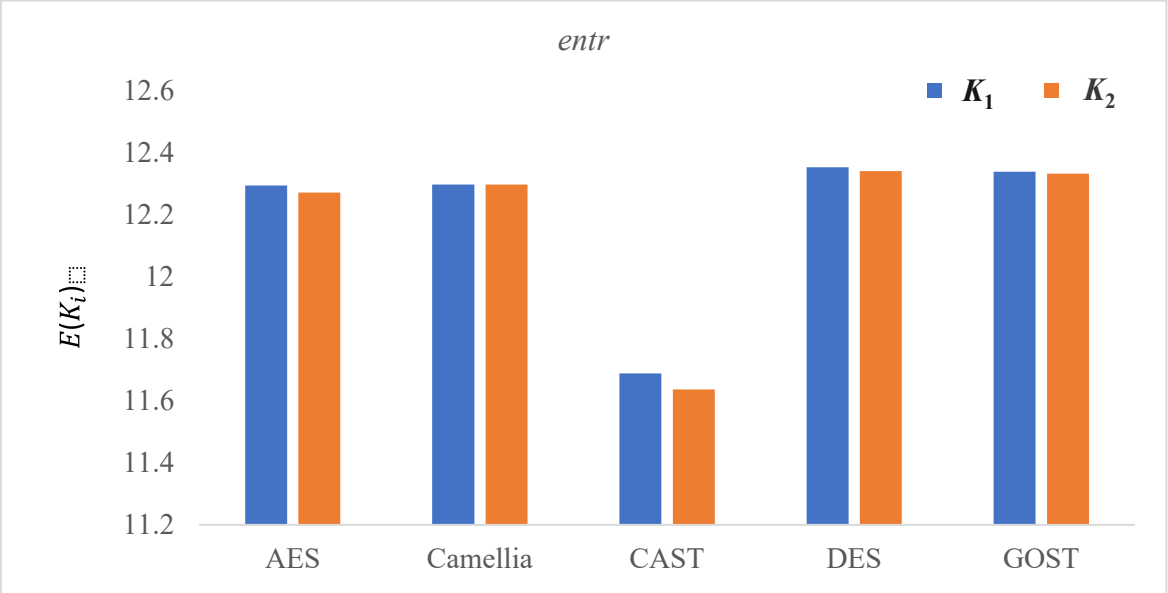



*Figure 7.2 Entropy analysis entr $E(K_i)$ for $K_1$ and $K_2$ using five block ciphers.*

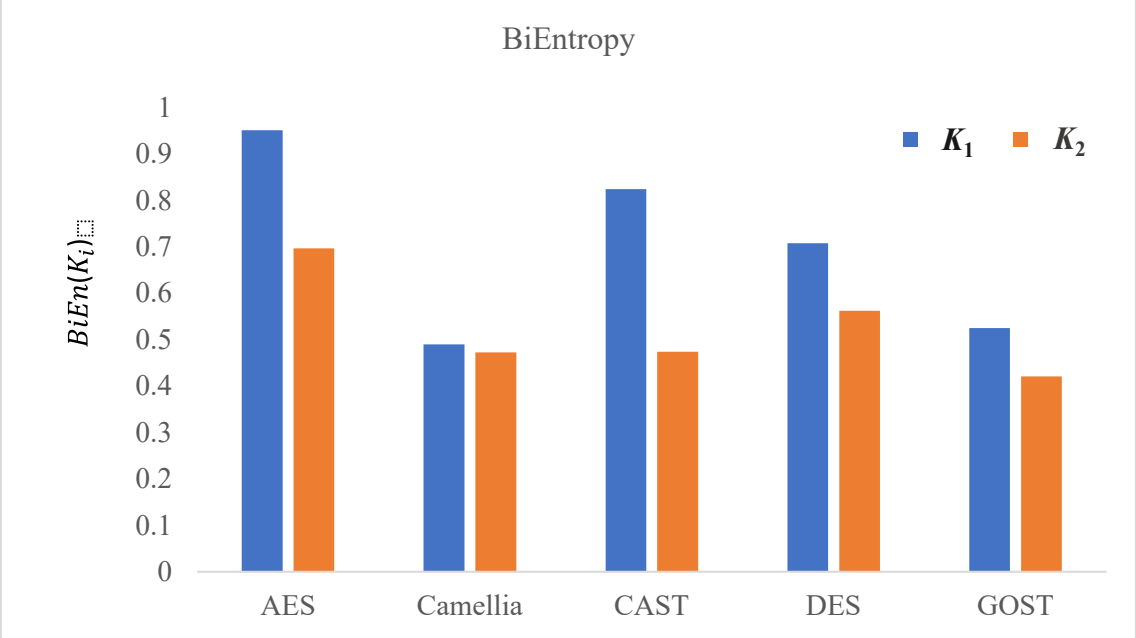



*Figure 7.3 Bi_ Entropy analysis $BiEn(K_i)$ for $K_1$ and $K_2$ using five block ciphers.*

*The $K_1$ and $K_2$ of five different block ciphers are compared with the 500 and 50 different subkeys using two entropy tests: entr and Bi_Entropy. Statistical analysis shows that the key-schedule generated by quantum random bits are more unpredictable than pseudo random for all the block ciphers.*

The analysis also proves that the quantum random number-based($K_1$) key-schedule is more unpredictable than the pseudo-random($K_2$) one. Unpredictability increases the $K_1$

schedule's strength, making it strong and hard to do cryptanalysis to partially access the key with Related-Key and Fault-Injection Attacks[218].

#### *7.1.2.5. Encryption Time*

The encryption time for all of the block ciphers was calculated to evaluate the impact of the $K_1$ and $K_2$. The research used two different file sizes, 8 MB and 16 MB, to illustrate the time required to convert plaintext to ciphertext. The encryption time for each file size can be seen in Figure 7.4. DES takes the longest time to encrypt, while GOST takes the second-longest time. The minimum time computation is by AES in both the $K_1$ and $K_2$ schedules. The analysis also shows that quantum- and pseudo-based key-schedules are taking nearly the same time for encryption. All the ciphers showed nearly the same transformation time with $K_1$ and $K_2$, with AES taking the least time among all the ciphers for both the schedules.

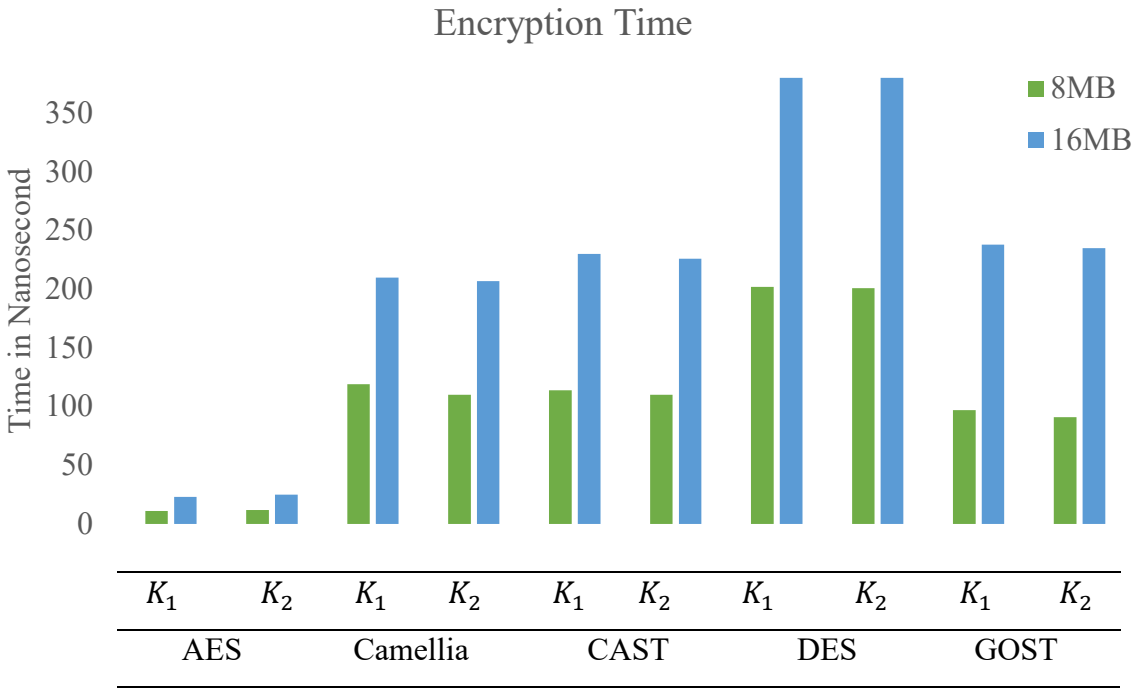



*Figure 7.4 Encryption Time of plaintext with $K_1$ and $K_2$ with file size of 8 MB and 16 MB.*

*The time of converting plaintext to ciphertext with the help of quantum- and pseudo-based key-schedules is measured in nanoseconds.*

Chapter 8 concludes the CryptoQNRG, a new framework in order to evaluate the strength of RNG-based key-schedules.

## 7.2. The QuantumGS-box comparison with existing dynamic S-box

This section covers results and data. The result is drawn on the strength of S-box measures such as nonlinearity, bit-independence, and avalanche properties including differential and

linear approximation analysis. There are several tools to evaluate the performance analysis of cryptographic properties of the S-box. The most common of them are MATLAB, Sage [219], SET [220] and the S-box performance analysis [221] tool. The results were evaluated using SET and the S-box performance tool.

### 7.2.1. Bijectivity property

An S-box of n x n size is said to be bijective if it has all possible output values from interval [0,2n-1]. If $f_i(1 \leq i \leq n)$ is a Boolean function of an S-box [36], it satisfies

$$\text{wt}(\sum_{i=1}^{n} a_i f_i) = 2^{n-1} \quad 7\text{-}1$$

where $a_i \in \{0,1\}, (a_1 a_2 \dots \dots a_n) \neq (0,0 \dots .0)$and wt(.) is the hamming weight, which indicates the number of 1s in a given vector. $f_i$ is to be balanced between 0 and 1. The proposed QuantumGS-box generates the n x n matrix of unique values, as shown in Table 5.3.

### 7.2.2. Nonlinearity

An S-box is strong if it has the Boolean function with a high nonlinearity value. The nonlinearity [222] Nf of a Boolean function f(x) is:

$$N_f = 2^{n-1}\left(1 - 2^{-n} \max\left|S_f(\omega)\right|\right) \quad 7\text{-}2$$

where $f_x \; if \; defined \; as$

$$S_f(\omega) = \sum_{\omega \in GF(2^n)}^{n} (-1)^{f(x)+x.\omega} \quad 7\text{-}3$$

where $\omega \in GF(2^n) \; and \; x.\omega$ represents the dot product of x and $\omega$.

Table 7.4 shows the nonlinearity of different S-boxes generated by different generations. The speed illustrates the time taken to generate the dynamic S-box bits based on generations.

Table 7.5 presents a comprehensive comparison of S-box security analysis of nonlinearity. The sample of QuantumGS-box ($75^{th}$ generation), generated by the proposed method, is compared with relevant dynamic S-boxes published in the last few years in different categories. In the proposed work, a maximum of 110 nonlinearity and an average of 108.75 are achieved. Figure 7.5 shows the graphical representation of comparison of nonlinearity.

*Table 7.4 Nonlinearity of the QuantumGS-box with three different generations and their generation time*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Generation 50, Time: 3.238 sec |
| Nonlinearity | 108 | 110 | 112 | 112 | 112 | 110 | 110 | 110 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Generation 75, Time: 4.560 sec |
| Nonlinearity | 106 | 108 | 108 | 110 | 108 | 110 | 110 | 110 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Generation 100, Time: 4.874 sec |
| Nonlinearity | 112 | 108 | 108 | 108 | 108 | 108 | 108 | 110 |

*Table 7.5 Comparison of QuantumGS-box with different research studies in terms of nonlinearity*

| **Design Methodology** | Ref | Non-Linearity | | |
|---|---|---|---|---|
| | | Min | Max | Avg |
| Algebraic | [223] | 106 | 108 | 107 |
| | [224] | 106 | 108 | 107.25 |
| | [225] | 104 | 108 | 106.8 |
| | [226] | 102 | 111 | 106.5 |
| | [160] | 106 | 110 | 107.75 |
| | [227] | 106 | 108 | 106.5 |
| Chaos Based Design | [228] | 108 | 110 | 109.5 |
| | [229] | 104 | 110 | 106.3 |
| | [230] | 108 | 112 | 109.25 |
| | [231] | 100 | 106 | 104 |
| | [232] | 100 | 110 | 103.8 |
| | [233] | 100 | 108 | 105 |
| Others | [165] | 99 | 108 | 103.5 |
| | [234] | 102 | 110 | 106.5 |
| | [211] | 102 | 110 | 107 |
| **QuantumGS-box 75 Gen** | this work | 106 | 110 | 108.75 |

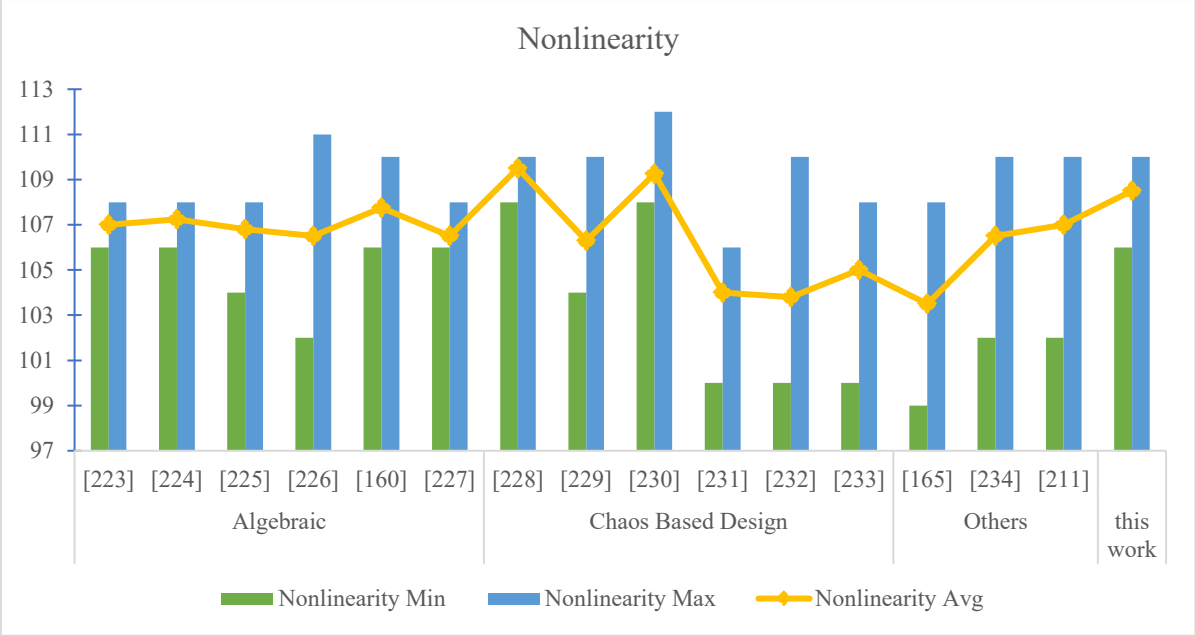



*Figure 7.5 Comparison of QuantumGS-box (this work) with different research studies in terms of nonlinearity.*

### 7.2.3. Bit- Independence criteria – Strict Avalanche criteria (BIC-SAC)

The Bit-Independence Criteria (BIC) tests measure the correlation between each pair of output bits when one input bit changes in response to the change in the output bit. In this test, the diagonal is excluded from the output matrix of n x n dimensions.

If a Boolean function satisfies the SAC [35], half of the output bits should change when there is a change in one bit. Any change in the input vector will significantly changes the output vector with a probability of ½. Table 7.6 shows the BIC-SAC values of the proposed QuantumGS-box. The research obtained a maximum, minimum, and average SAC value for QuantumGS-box equal to 0.525391, 0.46875, and 0.497070 correspondingly. Based on the results, the proposed QuantumGS-box fulfils the property of SAC as the average value for the QuantumGS-box is equal to the desired value of SAC (0.5).

*Table 7.6 BIC-SAC of QuantumGS-box*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0.511719 | 0.503906 | 0.525391 | 0.513672 | 0.486328 | 0.46875 | 0.509766 |
| 0.511719 | 0 | 0.511719 | 0.496094 | 0.498047 | 0.507813 | 0.507813 | 0.515625 |
| 0.503906 | 0.511719 | 0 | 0.527344 | 0.501953 | 0.507813 | 0.486328 | 0.505859 |
| 0.525391 | 0.496094 | 0.527344 | 0 | 0.519531 | 0.517578 | 0.503906 | 0.503906 |
| 0.513672 | 0.498047 | 0.501953 | 0.519531 | 0 | 0.519531 | 0.515625 | 0.5 |
| 0.486328 | 0.507813 | 0.507813 | 0.517578 | 0.519531 | 0 | 0.509766 | 0.488281 |
| 0.46875 | 0.507813 | 0.486328 | 0.503906 | 0.515625 | 0.509766 | 0 | 0.517578 |
| 0.509766 | 0.515625 | 0.505859 | 0.503906 | 0.5 | 0.488281 | 0.517578 | 0 |

### 7.2.4. Linear approximation probability

The LAP assesses the security of S-boxes against linear cryptanalysis. An S-box provides diffusion and confusion of bits through linear mappings between inputs and outputs. The LP of an event determines a maximum imbalance.[109][220]

$$LAP = \max_{p_x, q_x \neq 0} \left| \frac{|\{ x \mid (x \in R) \wedge (x.p_x = S(x).q_x) \}|}{2^n} - \frac{1}{2} \right| \qquad 7\text{-}4$$

where,

$p_x$ and $q_x$ are the input and output values respectively and $R = \{1,2,3,4 \ldots 255\}$

A small linear probability in an S-box makes the box very resistant to linear cryptanalysis. In the proposed work a LAP value of 0.1015 is achieved. Table 7.7 compares the maximum LP of different researchers with the proposed work. Figure 7.6 shows the graphical representation of the comparison data.

*Table 7.7 Comparison of QuantumGS-box with different research studies in terms of LAP.*

| Design Methodology | Ref | Linear Approximation Probability (LAP) |
|---|---|---|
| Algebraic | [223] | 0.1560 |
| | [224] | 0.1094 |
| | [225] | 0.1400 |
| | [226] | 0.1090 |
| | [160] | 0.1172 |
| | [227] | 0.1250 |
| Chaos Based Design | [228] | 0.1328 |
| | [229] | 0.1250 |
| | [230] | 0.1250 |
| | [231] | 0.1328 |
| | [232] | 0.1250 |
| | [233] | 0.1250 |
| Others | [165] | 0.1406 |
| | [234] | 0.1484 |
| | [211] | 0.1172 |
| **QuantumGS-box 75 Gen** | this work | 0.1015 |

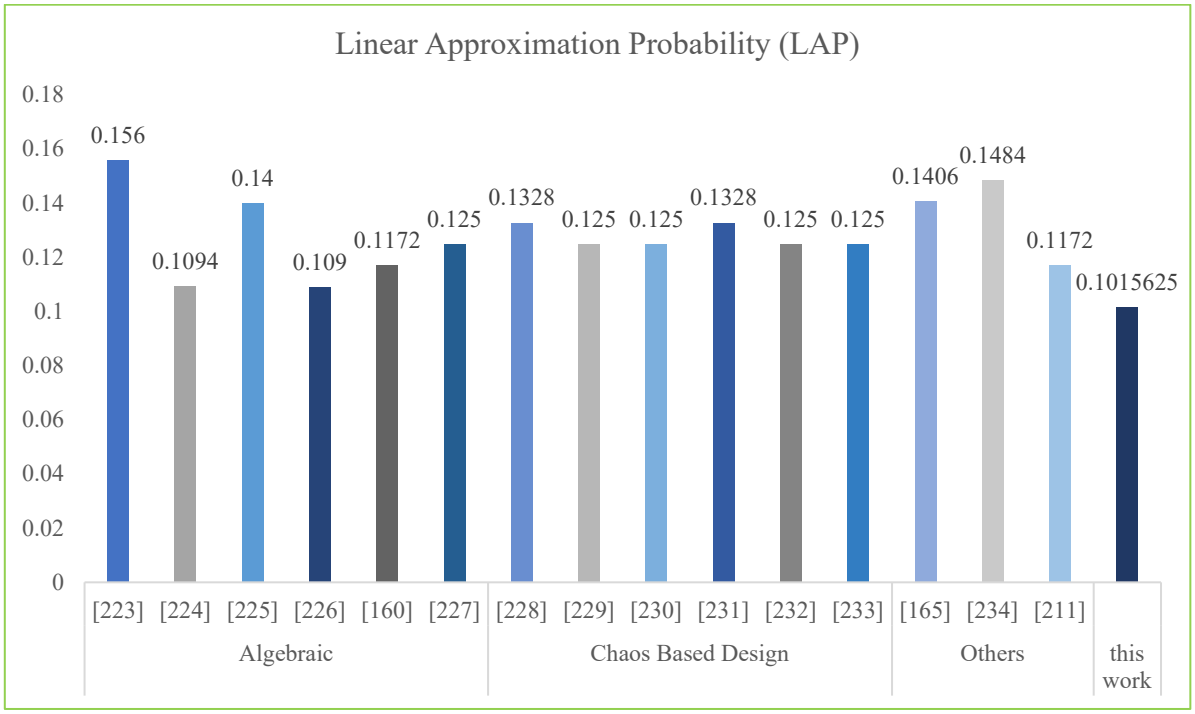



*Figure 7.6 Comparison of QuantumGS-box (this work) with different research studies in terms of LAP.*

### 7.2.5. Differential approximation probability

An effective way to assess S-box resistance to differential attacks is to use the DP [109][235]. DP indicates the probability that a particular change in output bits will occur due to a change in input bits.

The DP is calculated as follows

$$DP(\Delta x \rightarrow \Delta y) = \left( \frac{\left| \left\{ x \mid (x \in X) \ \wedge \left( \left( S(x) \oplus S(x \oplus \Delta x) \right) = \Delta y \right) \right\} \right|}{2^n} \right) \qquad 7\text{-}5$$

where X represents the set of all possible input values, 2n is a total number of all the elements. An S-box with a small differential value is strongly resistant to differential cryptoanalysis. In the proposed work a DAP value of 0.03125 is achieved.

### 7.2.6. Balanced output

An S-box with n input bits and m output bits, m ≤ n, is balanced if each output occurs 2n−m times. For the S-box to be balanced [35], it should have the same number of 0s and 1s. The result from the SET [37] tool is that the QuantumGS-box is balanced.

## 7.3. Summary and conclusion

This chapter discussed the evaluation and analysis of both CryptoRNG and the QuantumGS-box. It first discussed the data collection for the CryptoRNG based on pseudo and true random numbers. It then evaluated and differentiated the strengths and weaknesses of the KSAs of five different block ciphers. (AES, Camellia, CAST, DES, and GOST ).

The chapter also focused on evaluating the various cryptographic properties of the QuntumGS-box and compared them to existing research studies in a similar area.

The next chapter discusses the conclusion of the proposed work. This is of great significance because it not only summarizes the findings and results but also provides a comprehensive understanding of the research implications and potential contributions to the field. It reflects the hard work and dedication put into the project, ultimately paving the way for future studies and advancements in the subject matter.

# Chapter 8. Conclusion

## 8.1. Discussion of objectives and research question

A HSCSN provides the flexibility of storing data remotely and ensures its availability as required. The data travel from different layers of networks, exhibiting a risk of information disclosure by various attackers. Cloud model mapping to the security model leads to data security with cryptosystems. The cryptosystem provides a KSA to generate the key for encryption (that converts the plaintext into ciphertext) and decryption (that converts ciphertext into plaintext). The research focuses on sending the data from the cloud user to cloud storage, passing through different layers of high-speed networks. This research is based on the secured data travelling in an encrypted form before being transferred to the network from the user and stored in an encrypted form. The encrypted data is based on the KSA and its components. The literature review focused on various cryptosystems *(Section 3.1.1)* and attacks on cryptosystems *(Section 3.1.2),* focusing on the KSA, its cryptographic parameters, and its components. The security of components deals with randomization, which includes RNGs in this study.

The RNG generates random numbers that possess randomness in different components of the cryptosystems. The different RNGs were classified (PRNGs, CDRNGs, QRNGs) with their different properties and focused towards the true random number based on quantum principles, the QRNG for this study. *Section 3.2* discussed various cryptosystems based on RNGs that affected the various properties of the cryptosystems. The RNGs generate numbers based on different sources of entropy and were classified into different categories with their properties. *Section 3.3* discussed various RNGs and focused on the properties or objectives for using QRNGs. This study uses a USB-based QRNG to focus on the components of the cryptosystems.

This research proposed CryptoQNRG *(Chapter 5),* a new framework to enable the evaluation of the strength of RNG-based key-schedules using four tests: Frequency, Bit-Correlation, Bit-Interfold, and Entropy. The test suite evaluated the resilience of subkeys of the KSA in terms of the balance of 0s and 1s, the correlation of bits, confusion and diffusion, and an essential parameter of security, uncertainty.

The proposed CryptoQNRG was evaluated *(Chapter 7)* and assessed the subkeys of the most common KSA with quantum and pseudo random numbers. The main focus of the thesis was to compare the strength of the KSA based on RNGs, as compared to Afzal et al. [155], who evaluated the subkeys without considering them. The results indicate the strength of a

quantum- and pseudo-based key-schedules and their cryptographic properties, and showed that CAST did not pass the Bit-Correlation test and that keys are prone to cipher attacks. The analysis also indicated that the AES, DES, and GOST did not pass the Bit-Interfold test, whereas CAST and Camellia did. However, the computational time required to generate a cipher with a quantum random number-based($K_1$) key-schedule and pseudo-random($K_2$) of AES was much faster than the others. The results also revealed that a quantum-based key is less predictable than a pseudo random number-based key.

This research concluded that the quantum random number-based GA generated dynamic QuantamGS-boxes with different random S-box values at different arbitrary positions for cloud-based storage. The dynamic feature of the algorithm based on bit-shuffling with the operations of a GA also made it more resilient to modern cryptographic attacks. QuantumGS-boxes generated by the proposed algorithm were majorly nonlinear. The QuantumGS-box also had low differential and linear approximation properties in addition to exhibiting bijectivity cryptography properties, making it resistant to differential and linear attacks. Low differential and linear approximation properties meant that guessing the algorithm's output from its inputs was not possible and made it resistant to attacks that sought to exploit patterns in the input or output of the cryptographic system. Moreover, the different generations of the proposed QuantumGS-box were generated at a speed of a few seconds. Therefore, the bits of the S-box could be generated quickly and efficiently, increasing the security of the encryption system. The overall strengthened design of the proposed S-box ensured that cloud-based storage over high-speed networks was more resilient to various cryptographic attacks. *Chapter 6* focused on generating the QuantumGS-box and *Chapter 7* analysed and compared its different cryptographic properties with those of existing S-boxes.

In conclusion, this research proposed and investigated the random number-based KSA evaluation CryptoQNRG and a dynamic QRNG and GA-based S-box QuantumGS-box.

## 8.2. Research limitation

The research focused on the data security of an HSCSN, where the sender, Alice, sends data encrypted by a QRNG based on the source of quantum randomness and the evolution computation of a GA stored on the cloud. The stored encrypted data is fetched from the cloud whenever required by the local machine and decrypted by Alice to access the contents of the data in the original form. The research includes the quantum randomness and genetic computation in a key generated for symmetric encryption.

However, asymmetric encryption is also capable of utilizing a quantum source for key generation, but creating and storing a pair of public and private keys still requires further exploration and research.

## 8.3. Summary and conclusion

This chapter concluded the research questions and highlighted the research outcomes, specifically in terms of CryptoQRNG and the QuantumGS-box. The detailed analysis in the previous chapter helped to draw conclusions and answer the research questions. The next chapter focuses on the future work of the QuantumGS-box.

# Chapter 9. Future work

## 9.1. The QuantumGSbox with IoT smart home system

Future work will focus on proposing the QuantumGS-box in the IoT Smart Home System. Using symmetric cryptosystems as part of IoT security enhances the security of IoT devices. The implementation of a smart home involves connecting devices in the home to sensors, and controlling them with IoT devices such as Raspberry Pi. These sensors collect data and store it on the cloud for future use.

Another focus is to incorporate a specific machine learning algorithm for a QuantumGSbox and analyse the impact on different cryptographic properties.

The future work also includes designing a physical-digital device comprised of Raspberry Pi, and QRNG. The device will be connected with a LAN connection to any computer or laptop.

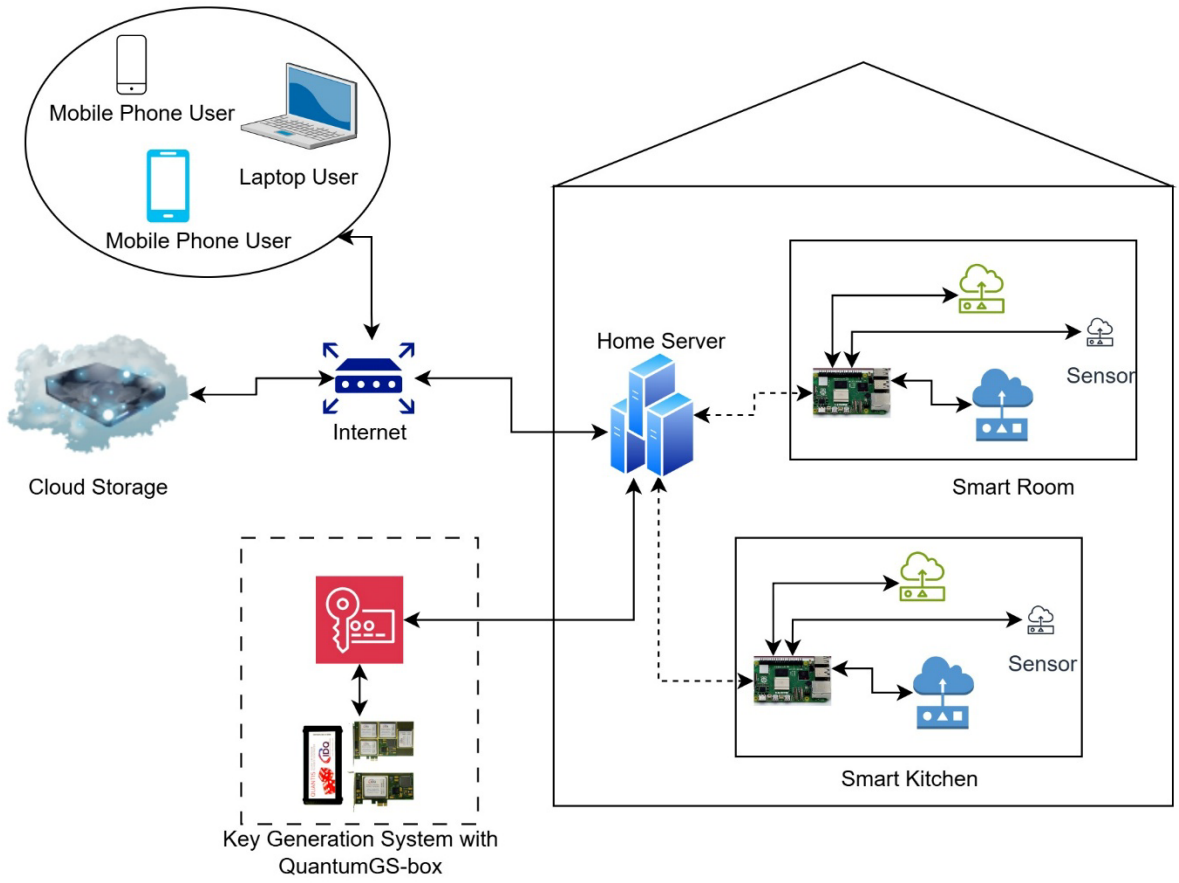



*Figure 9.1 IoT-based Smart Home System with QuantumGS-box.*

The encryption process of the cryptosystem using the QuantumGS-box encrypts the user's data. The device sends the encrypted data over the high-speed network on cloud storage and receives the same encrypted data. The received encrypted data will be decrypted by the device with the key entered by the user. The device encrypts the user's data in front of them at their premises. The data will be encrypted before being sent over to the internet, and the user will receive the same encrypted data from the internet. The device is then used to decrypt the data. The device also eliminates the requirement for a key storage management system.

Figure 9.1 [236] illustrates a smart home system based on IoT technology, where different rooms are connected via separate IoT networks equipped with sensors. The home server generates keys using a device called an S-box, encrypts the data collected by the various sensors, and stores it on the cloud. The encrypted data can be retrieved later by the home server and will then be decrypted to analyse or control the other sensors in the home.

## 9.2. Final thoughts

This study focused on the data security of a high-speed cloud storage network, where a user of cloud storage encrypts the data with the KSA and S-box of the cryptosystem. The cryptographic characteristics indicate that the QuantumGS-box is a promising candidate for real-world cloud-based storage encryption applications, where the sender and receiver are the same. Also, $KSEC_{RNG}$, the framework of the KSA, evaluates the strength of an RNG-based KSA.

Future work will focus on the use of QuantumGS-box in IoT-based smart home systems and storing encrypted sensor data in cloud storage. The future work of the study also includes testing the KSA of lightweight cryptographic algorithms that play a major role in the field of the Internet of Things (IoT).

# Bibliography


[1] J. Wang and Z. Wang, "A Survey on Personal Data Cloud," *Scientific World Journal*, vol. 2014, 2014, doi: 10.1155/2014/969150.

[2] S. H. Eck, "Challenges in data storage and data management in a clinical diagnostic setting," *Journal of Laboratory Medicine*, vol. 42, no. 6, pp. 219–224, 2018, doi: 10.1515/labmed-2018-0054.

[3] M. A. Alyami, M. Almotairi, L. Aikins, A. R. Yataco, and Y.-T. Song, "Managing personal health records using meta-data and cloud storage," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017, pp. 265–271, doi: 10.1109/ICIS.2017.7960004.

[4] P. Yang, N. Xiong, and J. Ren, "Data Security and Privacy Protection for Cloud Storage: A Survey," *IEEE Access*, vol. 8, pp. 131723–131740, 2020, doi: 10.1109/ACCESS.2020.3009876.

[5] Santiago Lucas Obrutsky, "Cloud Storage: Advantages, Disadvantages and Enterprise Solutions for Business Santiago," *EIT New Zealand*, no. July, 2016, [Online]. Available: https://www.researchgate.net/profile/Santiago-Obrutsky/publication/305508410_Cloud_Storage_Advantages_Disadvantages_and_Enterprise_Solutions_for_Business/links/5792976508ae33e89f7cc136/Cloud-Storage-Advantages-Disadvantages-and-Enterprise-Solutions-for-Bu.

[6] I. Odun-Ayo, O. Ajayi, B. Akanle, and R. Ahuja, "An Overview of Data Storage in Cloud Computing," in *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, 2017, pp. 29–34, doi: 10.1109/ICNGCIS.2017.9.

[7] P. A. Abdalla and A. Varol, "Advantages to disadvantages of cloud computing for small-sized business," *7th International Symposium on Digital Forensics and Security, ISDFS 2019*, pp. 1–6, 2019, doi: 10.1109/ISDFS.2019.8757549.

[8] G. Gonçalves, A. B. Vieira, I. Drago, A. P. C. Da Silva, and J. M. Almeida, "Cost-Benefit Tradeoffs of Content Sharing in Personal Cloud Storage," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2017, pp. 32–42, doi: 10.1109/MASCOTS.2017.32.

[9] M. Albahar, "The Insider Threats," *International Journal of New Technology and Research (IJNTR)*, vol. 1, no. 1, pp. 16–21, 2015.

[10] E. Bertino, *Data Protection from Insider Threats*, vol. 4, no. 4. 2012.

[11] S. Namasudra, P. Roy, and B. Balusamy, "Cloud computing: Fundamentals and research issues," *Proceedings - 2017 2nd International Conference on Recent Trends and Challenges in Computational Models, ICRTCCM 2017*, pp. 7–12, 2017, doi: 10.1109/ICRTCCM.2017.49.

[12] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami, and M. Ayaz, "A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies," *IEEE Access*, vol. 9, pp. 57792–57807, 2021, doi: 10.1109/ACCESS.2021.3073203.

[13] N. Attar and M. Shahin, "A Proposed Architecture for Data Security in Cloud Storage Space," *Journal of Biostatistics and Biometric Applications*, vol. 3, no. 2, pp. 1–7, 2018.

[14] A. Albugmi, M. O. Alassafi, R. Walters, and G. Wills, "Data security in cloud computing," *5th International Conference on Future Generation Communication Technologies, FGCT 2016*, pp. 55–59, 2016, doi: 10.1109/FGCT.2016.7605062.

[15] CSA, "SECURITY GUIDANCE FOR CRITICAL AREAS OF FOCUS IN CLOUD COMPUTING V3.0," 2011.

[16] W. Stallings, "Cryptography and Network Security: Principles and Practices," *Cryptography and Network Security*. Pearson, USA, 2005.

[17] X. Zhang, W. Li, H. Hu, and N. K. Dutta, "High-speed all-optical encryption and decryption based on two-photon absorption in semiconductor optical amplifiers," *Journal of Optical Communications and Networking*, vol. 7, no. 4, pp. 276–285, 2015, doi: 10.1364/JOCN.7.000276.

[18] P. Chinnasamy and P. Deepalakshmi, "Design of Secure Storage for Health-care Cloud using Hybrid Cryptography," *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, no. Icicct, pp. 1717–1720, 2018, doi: 10.1109/ICICCT.2018.8473107.

[19] L. Gong, J. Zhang, H. Liu, L. Sang, and Y. Wang, "True Random Number Generators Using Electrical Noise," *IEEE Access*, vol. 7, pp. 125796–125805, 2019, doi: 10.1109/ACCESS.2019.2939027.

[20] J. Lee, Y. Seo, and J. Heo, "Analysis of random number generated by quantum noise source and software entropy source," *In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Jeju, Korea (South), pp. 729–732, Oct. 17, 2018, doi: 10.1109/ICTC.2018.8539618.

[21] S. Sinha, S. H. Islam, and M. S. Obaidat, "A comparative study and analysis of some pseudorandom number generator algorithms," *Security and Privacy*, vol. 1, no. 6, p. e46, 2018, doi: 10.1002/spy2.46.

[22] A. I. S. Oy, "Araneus Alea II," 2022. https://www.araneus.fi/products/alea2/en/ (accessed Aug. 09, 2021).

[23] Ubldit, "TrueRNG v3," *ubldit*, 2022. https://ubld.it/truerng_v3 (accessed Aug. 09, 2021).

[24] C. Supply, "Infinite noise TRNG," *Crowd Supply*, 2022. https://www.crowdsupply.com/leetronics/infinite-noise-trng (accessed Jan. 02, 2022).

[25] M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum random number generators," *Reviews of Modern Physics*, vol. 89, no. 1, p. 015004, Feb. 2017, doi:

10.1103/RevModPhys.89.015004.

[26] ID Quantique, "What is the Q in QRNG ?" 2020, Accessed: Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/random-number-generation/overview/.

[27] G. Shaw, S. R. Sivaram, and A. Prabhakar, "Quantum Random Number Generator with One and Two Entropy Sources," *In Proceedings of the National Conference on Communications (NCC)*. IEEE, Bangalore, India, pp. 1–4, Feb. 20, 2019, doi: 10.1109/NCC.2019.8732222.

[28] G. Mogos, "Quantum Random Number Generator vs. Random Number Generator," in *IEEE International Conference on Communications*, Jun. 2016, pp. 423–426, doi: 10.1109/ICComm.2016.7528306.

[29] N. Xiao and M. Armstrong, "Genetic Algorithms and Evolutionary Computing," *Geographic Information Science & Technology Body of Knowledge*, vol. 2020, no. Q1, pp. 1–7, 2020, doi: 10.22224/gistbok/2020.1.1.

[30] F. F. Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud computing challenges and opportunities: A survey," *2015 International Conference on Telematics and Future Generation Networks, TAFGEN 2015*, pp. 34–38, 2015, doi: 10.1109/TAFGEN.2015.7289571.

[31] M. Jones, "Anatomy of a cloud storage infrastructure," *IBM*, 2010. https://developer.ibm.com/articles/cl-cloudstorage/.

[32] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, "Strengthening the key schedule of the AES," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2384, pp. 226–240, 2002, doi: 10.1007/3-540-45450-0_19.

[33] R. Hughes and J. Nordholt, "Strengthening the Security Foundation of Cryptography With Whitewood ' S Quantum-Powered Entropy Engine." Whitewood Encryption Systems, Inc, Boston, MA, 2016, Accessed: Aug. 03, 2021. [Online]. Available: http://www.whitewoodencryption.com.

[34] M. Ahmad, N. Mittal, P. Garg, and M. Maftab Khan, "Efficient cryptographic substitution box design using travelling salesman problem and chaos," *Perspectives in Science*, vol. 8, pp. 465–468, 2016, doi: 10.1016/j.pisc.2016.06.001.

[35] G. Jacob, A. Murugan, and I. Viola, "Towards the Generation of a Dynamic Key-Dependent S-Box to Enhance Security.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 92, 2015, [Online]. Available: http://dblp.uni-trier.de/db/journals/iacr/iacr2015.html#JacobMV15.

[36] M. Ahmad, H. Haleem, and P. M. Khan, "A new chaotic substitution box design for block ciphers," *2014 International Conference on Signal Processing and Integrated Networks, SPIN 2014*, pp. 255–258, 2014, doi: 10.1109/spin.2014.6776958.

[37] A. Nitaj, W. Susilo, and J. Tonien, "A New Improved AES S-box with Enhanced Properties," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial*

*Intelligence and Lecture Notes in Bioinformatics)*, vol. 12248 LNCS, no. Acisp 2020, pp. 125–141, 2020, doi: 10.1007/978-3-030-55304-3_7.

[38] J. Daemen and V. Rijmen, *The Design of Rijndael*. 2002.

[39] J. Daemen and V. Rijmen, *The Design of Rijndael*. Springer Berlin Heidelberg, 2002.

[40] D. Gullasch, E. Bangerter, and S. Krenn, "Cache games - Bringing access-based cache attacks on AES to practice," in *IEEE Symposium on Security and Privacy*, May 2011, pp. 490–505, doi: 10.1109/SP.2011.22.

[41] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in *Advances in Cryptology – ASIACRYPT Lecture Notes in Computer Science*, Springer, 2009, pp. 1–18.

[42] T. Lunghi *et al.*, "Self-Testing Quantum Random Number Generator," *Phys. Rev. Lett.*, vol. 114, no. 15, p. 150501, Apr. 2015, doi: 10.1103/PhysRevLett.114.150501.

[43] H. Xu, D. Perenzoni, A. Tomasi, and N. Massari, "A 16 × 16 Pixel Post-Processing Free Quantum Random Number Generator Based on SPADs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 627–631, 2018, doi: 10.1109/TCSII.2018.2821904.

[44] R. C. Pooser, P. G. Evans, and T. S. Humble, "Self correcting quantum random number generators using tapered amplifiers," *In Proceedings of the IEEE Photonics Society Summer Topical Meeting Series*. IEEE, Waikoloa, HI, USA, pp. 147–148, Jul. 08, 2013, doi: 10.1109/PHOSST.2013.6614471.

[45] J. M. Wang, T. Y. Xie, H. F. Zhang, D. X. Yang, C. Xie, and J. Wang, "A bias-free quantum random number generation using photon arrival time selectively," *IEEE Photonics Journal*, vol. 7, no. 2, 2015, doi: 10.1109/JPHOT.2015.2402127.

[46] Y.-H. Li *et al.*, "Quantum random number generation with uncharacterized laser and sunlight," *npj Quantum Information*, vol. 5, no. 1. p. 97, Dec. 14, 2019, doi: 10.1038/s41534-019-0208-1.

[47] C. Abellán *et al.*, "Ultra-fast quantum randomness generation by accelerated phase diffusion in a pulsed laser diode," *Optics Express*, vol. 22, no. 2, p. 1645, 2014, doi: 10.1364/oe.22.001645.

[48] Z. Cao, H. Zhou, X. Yuan, and X. Ma, "Source-independent quantum random number generation," *Physical Review X*, vol. 6, no. 1, pp. 1–11, 2016, doi: 10.1103/PhysRevX.6.011020.

[49] J. Liu *et al.*, "117 Gbits/s Quantum Random Number Generation with Simple Structure," *IEEE Photonics Technology Letters*, vol. 29, no. 3, pp. 283–286, 2017, doi: 10.1109/LPT.2016.2639562.

[50] M. Siswanto and B. Rudiyanto, "Designing of quantum random number generator (QRNG) for security application," *In Proceedings of the 3rd International Conference on Science in*

*Information Technology (ICSITech)*, vol. 2018-Janua. IEEE, Bandung, Indonesia, pp. 273–277, Oct. 25, 2017, doi: 10.1109/ICSITech.2017.8257124.

[51] S. Picek, E. Marchiori, L. Batina, and D. Jakobovic, "Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8672, pp. 822–831, 2014, doi: 10.1007/978-3-319-10762-2_81.

[52] E. Wirsansky, *Hands-on genetic algorithms with Python- Applying Genetic Algorithms to Solve Real-World Deep Learning and Artificial Intelligence Problems*, 1st ed. Packt Publishing, Limited, 2020.

[53] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, 2021, doi: 10.1007/s11042-020-10139-6.

[54] Y. Wang, K. W. Wong, C. Li, and Y. Li, "A novel method to design S-box based on chaotic map and genetic algorithm," *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 376, no. 6–7, pp. 827–833, 2012, doi: 10.1016/j.physleta.2012.01.009.

[55] Y. Wang, Z. Zhang, L. Y. Zhang, J. Feng, J. Gao, and P. Lei, "A genetic algorithm for constructing bijective substitution boxes with high nonlinearity," *Information Sciences*, vol. 523, pp. 152–166, 2020, doi: 10.1016/j.ins.2020.03.025.

[56] A. Biryukov and C. Cannière, "Data encryption standard (DES)," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, USA, Oct. 1999, doi: 10.1007/0-387-23483-7_94.

[57] J. L. Massey, "SAFER K-64: A byte-oriented block-ciphering algorithm," *In Proceedings of the International Workshop on Fast Software Encryption. Lecture Notes in Computer Science*. Springer: Berlin/Heidelberg, Germany, Cambridge, UK, pp. 1–17, Dec. 09, 1993, doi: 10.1007/3-540-58108-1_1.

[58] J. Daemen, R. Govaerts, and J. Vandewalle, "A new approach to block cipher design," *In Proceedings of the International Workshop on Fast Software Encryption. Lecture Notes in Computer Science.* Springer: Berlin/Heidelberg, Germany, Cambridge, UK, pp. 18–32, Dec. 09, 1993, doi: 10.1007/3-540-58108-1_2.

[59] R. Anderson, E. Biham, and L. Knudsen, "Serpent : A proposal for the advanced encryption standard," *NIST AES Proposal*. pp. 1–23, 1998, Accessed: Oct. 09, 2021. [Online]. Available: https://bitbucket.org/nicholascapo/network-security-project/src/fcbc6e93e555/Literature/serpent.pdf.

[60] R. J. J. Jr., "ISAAC and RC4," 1993. http://burtleburtle.net/bob/rand/isaac.html (accessed Jun. 12, 2022).

[61] J. Quirke, "Security in the GSM system," *AusMobile, May*, no. May. 2004, Accessed: Oct. 09, 2020. [Online]. Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.1509&rep=rep1&type=pdf.

[62] Security Algorithms Group of Experts, “Etr 278 - ETSI TECHNICAL REPORT.” 1996, Accessed: May 04, 2021. [Online]. Available: https://cryptome.org/espy/ETR278e01p.pdf.

[63] X. Lai and J. L. Massey, “A Proposal for a New Block Encryption Standard,” *In Proceedings of the Advances in Cryptology—EUROCRYPT ’90, Workshop on the Theory and Application of Cryp-tographic Techniques*, vol. 473 LNCS. Lecture Notes in Computer Science; Damgård, I.B., Ed.; Springer: Berlin/Heidelberg, Germany, Aarhus, Denmark, pp. 389–404, May 21, 1991, doi: 10.1007/3-540-46877-3_35.

[64] B. Schneier, “Description of a new variable-length key, 64-bit block cipher (Blowfish),” *In Proceedings of the International Workshop on Fast Software Encryption*. Springer: Berlin/Heidelberg, Germany, Cambridge, UK, pp. 191–204, Dec. 09, 1993, doi: 10.1007/3-540-58108-1_24.

[65] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, “Rabbit: A New High-Performance Stream Cipher,” *In Proceedings of the 10th International Workshop, Fast Software Encryption 2003*, vol. 2887. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, Lund,, Sweden, pp. 307–329, 2003, doi: 10.1007/978-3-540-39887-5_23.

[66] P. Hawkes and G. G. Rose, “Primitive Specification for SOBER-128.,” *IACR Cryptology ePrint Archive*, vol. 2003. p. 81, 2003, Accessed: Jan. 02, 2020. [Online]. Available: http://dblp.uni-trier.de/db/journals/iacr/iacr2003.html#HawkesR03a.

[67] C. Berbain, H. Gilbert, and J. Patarin, “QUAD: A multivariate stream cipher with provable security,” *Journal of Symbolic Computation*, vol. 44, no. 12, pp. 1703–1723, 2009, doi: 10.1016/j.jsc.2008.10.004.

[68] D. C. Christophe and B. Preneel, “Trivium Specifications,” *Trivium Specification*, vol. 507932. 2006, Accessed: Jan. 02, 2020. [Online]. Available: https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[69] D. J. Bernstein, “The salsa20 family of stream ciphers,” *In New Stream Cipher Designs; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4986 LNCS. Springer: Berlin/Heidelberg, Germany, pp. 84–97, 2008, doi: 10.1007/978-3-540-68351-3_8.

[70] D. J. Wheeler and R. M. Needham, “The Tiny Encryption Algorithm (TEA),” *In Proceedings of the Fast Software Encryption, Second International Workshop, Leuven, Belgium*, vol. 1008. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer: Berlin/Heidelberg, Germany, pp. 363–366, Dec. 14, 1994, doi: 10.1007/3-540-60590-8_29.

[71] C. Adams, “The CAST-128 Encryption Algorithm.” 1997, Accessed: Jun. 12, 2021. [Online].

Available: https://www.rfc-editor.org/info/rfc2144.

[72] P. Ekdahl and T. Johansson, "A new version of the stream cipher SNOW," *In Proceedings of the Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, NL, Canada,* vol. 2595. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, pp. 47–61, Aug. 15, 2003, doi: 10.1007/3-540-36492-7_5.

[73] L. R. Knudsen, V. Rijmen, R. L. Rivest, and M. J. B. Robshaw, "On the Design and Security of RC2," *In Proceedings of the Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France*. Springer: Berlin/Heidelberg, Germany, pp. 206–221, Mar. 23, 1998.

[74] "Crypto-1." https://en.wikipedia.org/wiki/Crypto-1 (accessed Jun. 12, 2021).

[75] R. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 Block Cipher," *In Proceedings of the First Advanced Encryption Standard (AES)*. Ventura, CA, 1998.

[76] C. Burwick *et al.*, "MARS - a Candidate Cipher for AES," *NIST AES Proposal*. pp. 8–23, 1998, Accessed: Jun. 12, 2021. [Online]. Available: http://cryptosoft.de/docs/Mars.pdf.

[77] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, and C. Hall, "Twofish : A 128-Bit Block Cipher," *NIST AES Proposal*, vol. 15, no. 1. pp. 1–27, 1998, Accessed: Jul. 07, 2021. [Online]. Available: https://www.schneier.com/wp-content/uploads/2016/02/paper-twofish-paper.pdf.

[78] H. Wu, "A new stream cipher HC-256," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3017, pp. 226–244, 2004, doi: 10.1007/978-3-540-25937-4_15.

[79] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, 2007, doi: 10.1504/IJWMC.2007.013798.

[80] S. Babbage and M. Dodd, "The stream cipher MICKEY 2.0," *ECRYPT Stream Cipher Project, Report*. pp. 1–12, 2006, Accessed: Jun. 12, 2021. [Online]. Available: https://www.cosic.esat.kuleuven.be/ecrypt/stream/p2ciphers/mickey/mickey_p2.pdf.

[81] "Japan's First 128-bit Block Cipher 'Camellia' Approved as a New Standard Encryption Algorithm in the Internet." NTT News Release, Accessed: Jul. 17, 2021. [Online]. Available: https://www.ntt.co.jp/news/news05e/0507/050720.html.

[82] C. Fontaine, "SEAL," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, USA, pp. 543–543, 2011, doi: 10.1007/0-387-23483-7_371.

[83] N. Ferguson *et al.*, "Threefish." https://www.schneier.com/academic/skein/threefish/ (accessed Aug. 02, 2021).

[84] C. Cannière, "GOST," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, 2011, doi: 10.1007/978-1-4419-5906-5_579.

[85] D. Whiting, B. Schneier, S. Lucks, and M. F, "Phelix." 2004, Accessed: Apr. 07, 2021. [Online]. Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.8097&rep=rep1&type=pdf.

[86] NIST, “Post-Quantum Cryptography.” https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions (accessed Jun. 12, 2022).

[87] R. L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[88] T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” *In Proceedings of the Advances in Cryptology*, vol. 196 LNCS, no. 4. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 10–18, 1985, doi: 10.1007/3-540-39568-7_2.

[89] FIPS, “Digital Signature Standard (DSS),” *National Institute of Standards and Technology: Gaithersburg, MD, USA*. 1994.

[90] Certicom Research, “Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography,” *Standards for Efficient Cryptography*, vol. 1, no. Sec 1, pp. 1–22, 2009.

[91] S. Josefsson and I. Liusvaara, “Edwards-Curve Digital Signature Algorithm (EdDSA),” *Journal of Chemical Information and Modeling*, vol. 53, no. 9. pp. 1689–1699, 2017, Accessed: Dec. 12, 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8032.

[92] Python, “random,” 2020. https://docs.python.org/3/library/random.html (accessed Jun. 23, 2020).

[93] Oracle, “SecureRandom,” 2020. https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#SecureRandom (accessed Jun. 23, 2020).

[94] T. Sivakumar and P. Li, “A secure image encryption method using scan pattern and random key stream derived from laser chaos,” *Optics and Laser Technology*, vol. 111, pp. 196–204, 2019, doi: 10.1016/j.optlastec.2018.09.048.

[95] R. Saha, G. Geetha, G. Kumar, and T. H. Kim, “RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys,” *Security and Communication Networks*, vol. 2018, pp. 1–11, Nov. 2018, doi: 10.1155/2018/9802475.

[96] P. V. Mankar, “Key updating for leakage resiliency with application to Shannon security OTP and AES modes of operation,” *IEEE International Conference on IoT and its Applications, ICIOT 2017*, vol. 10, no. 3, pp. 519–528, 2017, doi: 10.1109/ICIOTA.2017.8073631.

[97] S. Picek, C. Carlet, D. Jakobovic, J. F. Miller, and L. Batina, “Correlation immunity of Boolean functions: An evolutionary algorithms perspective,” *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, pp. 1095–1102, 2015, doi: 10.1145/2739480.2754764.

[98] N. Nedjah and L. De Macedo Mourelle, “Evolutionary resilient substitution boxes for secure cryptography using nash equilibrium,” *2006 International Conference on Computational Intelligence and Security, ICCIAS 2006*, vol. 2, pp. 1295–1298, 2006, doi:

10.1109/ICCIAS.2006.295266.

[99] F. L. Shi and H. Bin, "Propagation properties of symmetric Boolean functions," in *International Conference on Intelligent Computation Technology and Automation*, May 2010, pp. 947–950, doi: 10.1109/ICICTA.2010.614.

[100] J. Husa, "Hamming Weight Using Various Genetic Programming Methods," pp. 342–343, 2019.

[101] D. Tang, C. Carlet, X. Tang, and Z. Zhou, "Construction of highly nonlinear 1-resilient boolean functions with optimal algebraic immunity and provably high fast algebraic immunity," *IEEE Transactions on Information Theory*, vol. 63, no. 9, pp. 6113–6125, 2017, doi: 10.1109/TIT.2017.2725918.

[102] J. Wang, B. Hu, and Q. Wang, "The Walsh spectrum property of resilient Boolean functions," *CIS 2009 - 2009 International Conference on Computational Intelligence and Security*, vol. 1, pp. 409–413, 2009, doi: 10.1109/CIS.2009.21.

[103] J. Yang, "Constructions of Highly Nonlinear Resilient Vectorial Boolean Functions via Perfect Nonlinear Functions," *IEEE Access*, vol. 5, no. 1, pp. 23166–23170, 2017, doi: 10.1109/ACCESS.2017.2764919.

[104] Y. Chen, "A construction of balanced odd-variable boolean function with optimum algebraic immunity," *Proceedings - 2011 7th International Conference on Computational Intelligence and Security, CIS 2011*, pp. 852–855, 2011, doi: 10.1109/CIS.2011.192.

[105] Y. Zhang, M. Liu, and D. Lin, "On the immunity of rotation symmetric Boolean functions against fast algebraic attacks," *Discrete Applied Mathematics*, vol. 162, pp. 17–27, 2014, doi: 10.1016/j.dam.2013.04.014.

[106] C. Carlet, "On the nonlinearity of monotone Boolean functions," *Cryptography and Communications*, vol. 10, no. 6, pp. 1051–1061, 2018, doi: 10.1007/s12095-017-0262-5.

[107] K. Verma and D. K. Sharma, "Calculation of non-linearity and algebraic degree of constructed boolean function," in *2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, May 2017, pp. 501–505, doi: 10.1109/RTEICT.2017.8256647.

[108] Bharti, "Survey on the nonlinearity of Boolean functions," *Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2016*, pp. 882–884, 2016, doi: 10.1109/WiSPNET.2016.7566258.

[109] M. S. Mahmood Malik *et al.*, "Generation of Highly Nonlinear and Dynamic AES Substitution-Boxes (S-Boxes) Using Chaos-Based Rotational Matrices," *IEEE Access*, vol. 8, pp. 35682–35695, 2020, doi: 10.1109/ACCESS.2020.2973679.

[110] S. M. Kareem and A. M. S. Rahma, "A novel approach for the development of the Twofish algorithm based on multi-level key space," *Journal of Information Security and Applications*, vol. 50, Feb. 2020, doi: 10.1016/j.jisa.2019.102410.

[111] A. Vuppala, R. S. Roshan, S. Nawaz, and J. V. R. Ravindra, “An Efficient Optimization and Secured Triple Data Encryption Standard Using Enhanced Key Scheduling Algorithm,” in *Procedia Computer Science*, Jun. 2020, vol. 171, pp. 1054–1063, doi: 10.1016/j.procs.2020.04.113.

[112] Z. Cao, F. Chen, B. Chen, and X. Zhang, “Research on the balanced boolean functions satisfying strict avalanche criterion,” *Proceedings - 2015 International Conference on Computational Science and Computational Intelligence, CSCI 2015*, no. 1, pp. 680–684, 2016, doi: 10.1109/CSCI.2015.14.

[113] P. K. Behera and S. Gangopadhyay, “Analysis of Cost function using Genetic algorithm to construct balanced Boolean function,” *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, vol. 2018-Octob, no. October, pp. 1445–1450, 2019, doi: 10.1109/TENCON.2018.8650309.

[114] P. S. Foundation, “secrets,” 2022. https://docs.python.org/3/library/secrets.html (accessed Mar. 10, 2021).

[115] P. Nannipieri *et al.*, “True random number generator based on fibonacci-galois ring oscillators for fpga,” *Applied Sciences (Switzerland)*, vol. 11, no. 8, 2021, doi: 10.3390/app11083330.

[116] L. Crocetti, S. Di Matteo, P. Nannipieri, L. Fanucci, and S. Saponara, “Design and Test of an Integrated Random Number Generator with All-Digital Entropy Source,” *Entropy*, vol. 24, no. 2, 2022, doi: 10.3390/e24020139.

[117] “European Processor Initiative (EPI).” https://www.european-processor-initiative.eu/ (accessed Jun. 12, 2022).

[118] IDQ, “quantis-random-number-generator,” 2020. https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator (accessed Jul. 07, 2020).

[119] QRANGE, “qrng,” 2020. https://qrange.eu/project/qrng (accessed Jul. 27, 2020).

[120] A. QRNG, “ANU QRNG,” *ANU QRNG*, 2022. https://qrng.anu.edu.au/ (accessed Sep. 05, 2020).

[121] A. H. Kashmar and E. S. Ismail, “Blostream: A high speed stream cipher,” *Journal of Engineering Science and Technology*, vol. 12, no. 4, pp. 1111–1128, 2017.

[122] T. R. Patnala, D. Jayanthi, S. Majji, M. Valleti, S. Kothapalli, and S. R. Karanam, “A Modernistic way for KEY Generation for Highly Secure Data Transfer in ASIC Design Flow,” *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, Coimbatore, India, pp. 892–897, Mar. 06, 2020, doi: 10.1109/ICACCS48705.2020.9074200.

[123] A. Amro and E.-S. M. El-Alfy, “Known-plaintext attack and improvement of PRNG-based text encryption,” *2016 7th International Conference on Information and Communication Systems (ICICS)*. IEEE, Irbid, Jordan, pp. 233–238, May 05, 2016, doi: 10.1109/IACS.2016.7476117.

[124] T. Kowsalya, R. Ganesh Babu, B. D. Parameshachari, A. Nayyar, and R. M. Mehmood, "Low area PRESENT cryptography in FPGA using TRNG-PRNG key generation," *Computers, Materials and Continua*, vol. 68, no. 2, pp. 1447–1465, 2021, doi: 10.32604/cmc.2021.014606.

[125] M. Mishra and V. H. Mankar, "Text Encryption Algorithms based on Pseudo Random Number Generator," *International Journal of Computer Applications*, vol. 111, no. 2, pp. 1–6, 2015, doi: 10.5120/19507-0756.

[126] G. Mogos, "Use quantum random number generator in Diffie-Hellman key exchange protocol," *2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, Cluj-Napoca, Romania, pp. 1–6, May 19, 2016, doi: 10.1109/AQTR.2016.7501290.

[127] A. Arab, M. J. Rostami, and B. Ghavami, "An image encryption method based on chaos system and AES algorithm," *Journal of Supercomputing*, vol. 75, no. 10, pp. 6663–6682, 2019, doi: 10.1007/s11227-019-02878-7.

[128] A. Kr.Banthia and N. Tiwari, "Image Encryption using Pseudo Random Number Generators," *International Journal of Computer Applications*, vol. 67, no. 20, pp. 1–8, 2013, doi: 10.5120/11508-7226.

[129] J. Sen Teh and A. Samsudin, "A chaos-based authenticated cipher with associated data," *Security and Communication Networks*, vol. 2017, 2017, doi: 10.1155/2017/9040518.

[130] Ü. Çavuşoğlu, A. Akgül, A. Zengin, and I. Pehlivan, "The design and implementation of hybrid RSA algorithm using a novel chaos based RNG," *Chaos, Solitons & Fractals*, vol. 104, pp. 655–667, Nov. 2017, doi: 10.1016/j.chaos.2017.09.025.

[131] H. Mads, "RANDOM.ORG," 1998. https://www.random.org/ (accessed Jun. 12, 2022).

[132] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, doi: 10.1137/S0097539795293172.

[133] W. Wei and H. Guo, "Quantum random number generator based on the photon number decision of weak laser pulses," *2009 Conference on Lasers & Electro Optics & The Pacific Rim Conference on Lasers and Electro-Optics*. IEEE, Shanghai, China, pp. 1–2, Aug. 30, 2009, doi: 10.1109/CLEOPR.2009.5292482.

[134] J. A. Miszczak, "Employing online quantum random number generators for generating truly random quantum states in Mathematica," *Computer Physics Communications*, vol. 184, no. 1, pp. 257–258, 2013, doi: 10.1016/j.cpc.2012.08.012.

[135] E. De Jesus Lopes Soares, F. A. Mendonca, and R. V. Ramos, "Quantum random number generator using only one single-photon detector," *IEEE Photonics Technology Letters*, vol. 26, no. 9, pp. 851–853, 2014, doi: 10.1109/LPT.2014.2302436.

[136] X. Ma, X. Yuan, Z. Cao, B. Qi, and Z. Zhang, "Quantum random number generation," *npj*

*Quantum Information*, vol. 2, no. 1, Nov. 2016, doi: 10.1038/npjqi.2016.21.

[137] M. Avesani, D. G. Marangon, G. Vallone, and P. Villoresi, "Quantum Random Number Generator at 17 Gbps," *Nature Communications*, pp. 1–7, 2018, [Online]. Available: http://dx.doi.org/10.1038/s41467-018-07585-0.

[138] R. S. Hasan, S. K. Tawfeeq, N. Q. Mohammed, and A. I. Khaleel, "A true random number generator based on the photon arrival time registered in a coincidence window between two single-photon counting modules," *Chinese Journal of Physics*, vol. 56, no. 1, pp. 385–391, 2018, doi: 10.1016/j.cjph.2017.11.008.

[139] B. Septriani, O. de Vries, and M. Gräfe, "Quantum random number generation (QRNG) by phase diffusion process in a gain-switched semiconductor laser - new insights," *In Proceedings of the Conference on Lasers and Electro-Optics*. IEEE, San Jose, CA, USA, May 05, 2019, doi: 10.1364/CLEO_QELS.2019.FM2M.2.

[140] A. Tontini, L. Gasparini, N. Massari, and R. Passerone, "SPAD-Based Quantum Random Number Generator with an Nth-Order Rank Algorithm on FPGA," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 12, pp. 2067–2071, 2019, doi: 10.1109/TCSII.2019.2909013.

[141] M. Nicola, L. Gasparini, A. Meneghetti, and A. Tomasi, "A SPAD-Based Random Number Generator Pixel Based on the Arrival Time of Photons," *In Proceedings of the New Generation of CAS (NGCAS)*. IEEE, Genova, Italy, pp. 213–216, Sep. 2017, doi: 10.1109/NGCAS.2017.27.

[142] A. Sarkar and C. M. Chandrashekar, "Multi-bit quantum random number generation from a single qubit quantum walk," *Scientific Reports*, vol. 9, no. 1, pp. 1–11, 2019, doi: 10.1038/s41598-019-48844-4.

[143] H. McCabe, S. M. Koziol, G. L. Snider, and E. P. Blair, "Tunable, Hardware-Based Quantum Random Number Generation Using Coupled Quantum Dots," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 292–296, 2020, doi: 10.1109/TNANO.2020.2978859.

[144] M. Imran, V. Sorianello, F. Fresi, L. Potì, and M. Romagnoli, "Quantum Random Number Generator based on Phase Diffusion in Lasers using an On-chip Tunable SOI Unbalanced Mach-Zehnder Interferometer (uMZI)," *In Proceedings of the Optical Fiber Communication Conference (OFC) 2020*. OSA Technical Digest (Optica Publishing Group), Washington, D.C., 2020, doi: 10.1364/OFC.2020.M1D.5.

[145] K. B. Jithendra and T. K. Shahana, "New Results in Related Key Impossible Differential Cryptanalysis on Reduced Round AES-192," in *2018 International Conference On Advances in Communication and Computing Technology, ICACCT 2018*, Feb. 2018, pp. 291–295, doi: 10.1109/ICACCT.2018.8529666.

[146] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, Jan. 1991, doi: 10.1007/BF00630563.

[147] N. P. Smart, V. Rijmen, B. Warinschi, and G. Watson, “Algorithms, Key Sizes and Parameters Report,” *Report*. ENISA, Nov. 2014, Accessed: Sep. 09, 2021. [Online]. Available: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014.

[148] A. R. Hakim and Z. Z. Nusron, “An improved Lblock-s key schedule algorithm,” in *International Conference on Information and Communications Technology*, Jul. 2019, pp. 232–236, doi: 10.1109/ICOIACT46704.2019.8938569.

[149] S. Sulaiman, Z. Muda, J. Juremi, R. Mahmod, and S. M. Yasin, “A New ShiftColumn Transformation : An Enhancement of Rijndael Key Scheduling,” *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 3, pp. 160–166, 2013.

[150] J. Huang, H. Yan, and X. Lai, “Transposition of AES key schedule,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10143 LNCS, pp. 84–102, 2017, doi: 10.1007/978-3-319-54705-3_6.

[151] R. Shahzadi, S. M. Anwar, F. Qamar, M. Ali, and J. J. P. C. Rodrigues, “Chaos based enhanced RC5 algorithm for security and integrity of clinical images in remote health monitoring,” *IEEE Access*, vol. 7, Apr. 2019, doi: 10.1109/ACCESS.2019.2909554.

[152] S. Sahmoud, W. Elmasry, and A. Shadi, “Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher,” *International Arab Journal of e-Technology*, vol. 3, no. 1, pp. 17–26, Jan. 2013.

[153] B. Maram and J. M. Gnanasekar, “A Block Cipher Algorithm to Enhance the Avalanche Effect Using Dynamic Key-Dependent S-Box and Genetic Operations,” *International Journal of Pure and Applied Mathematics*, vol. 119, no. 10, pp. 399–418, 2018.

[154] G. Leurent and C. Pernot, “New Representations of the AES Key Schedule,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12696 LNCS. pp. 54–84, 2021, doi: 10.1007/978-3-030-77870-5_3.

[155] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, “Cryptographic Strength Evaluation of Key Schedule Algorithms,” *Security and Communication Networks*, pp. 1–9, May 2020, doi: 10.1155/2020/3189601.

[156] S. Afzal, U. Waqas, M. A. Mir, and M. Yousaf, “Statistical Analysis of Key Schedule Algorithms of Different Block Ciphers,” *Science International - Report*. Jun. 2015.

[157] J. Juremi, S. Sulaiman, N. H. M. Saad, and ..., “A survey on various dynamic S-box implementation in block cipher encryption algorithm,” *Journal of Applied …*, vol. 3, no. 1, pp. 2–5, 2019, [Online]. Available: https://jati.sites.apiit.edu.my/files/2019/05/A-Survey-on-Various-Dynamic-S-box-in-Block-Cipher-Encryption-Algorithm.pdf.

[158] N. Siddiqui, H. Khalid, F. Murtaza, M. Ehatisham-Ul-Haq, and M. A. Azam, “A novel algebraic technique for design of computational substitution-boxes using action of matrices on

galois field," *IEEE Access*, vol. 8, pp. 197630–197643, 2020, doi: 10.1109/ACCESS.2020.3034832.

[159] A. R. Alharbi, S. S. Jamal, M. F. Khan, M. A. Gondal, and A. A. Abbasi, "Construction and Optimization of Dynamic S-Boxes Based on Gaussian Distribution," *IEEE Access*, vol. 11, no. February, pp. 35818–35829, 2023, doi: 10.1109/ACCESS.2023.3262313.

[160] S. Ibrahim and A. M. Abbas, "Efficient key-dependent dynamic S-boxes based on permutated elliptic curves," *Information Sciences*, vol. 558, pp. 246–264, 2021, doi: 10.1016/j.ins.2021.01.014.

[161] A. H. Zahid *et al.*, "Efficient Dynamic S-Box Generation Using Linear Trigonometric Transformation for Security Applications," *IEEE Access*, vol. 9, pp. 98460–98475, 2021, doi: 10.1109/ACCESS.2021.3095618.

[162] Q. Lu, C. Zhu, and X. Deng, "An Efficient Image Encryption Scheme Based on the LSS Chaotic Map and Single S-Box," *IEEE Access*, vol. 8, pp. 25664–25678, 2020, doi: 10.1109/ACCESS.2020.2970806.

[163] S. Ibrahim, A. M. Abbas, A. A. Alharbi, and M. A. Albahar, "A New 12-Bit Chaotic Image Encryption Scheme Using a 12 × 12 Dynamic S-Box," *IEEE Access*, vol. 12, no. March, pp. 37631–37642, 2024, doi: 10.1109/ACCESS.2024.3374218.

[164] F. Özkaynak, "On the effect of chaotic system in performance characteristics of chaos based s-box designs," *Physica A: Statistical Mechanics and its Applications*, vol. 550, 2020, doi: 10.1016/j.physa.2019.124072.

[165] M. F. Khan *et al.*, "Construction and Optimization of TRNG Based Substitution Boxes for Block Encryption Algorithms," *Computers, Materials and Continua*, vol. 73, no. 2, pp. 2679–2696, 2022, doi: 10.32604/cmc.2022.027655.

[166] H. Ibrahim and F. Ozkaynak, "A Substitution-Box Structure Based on Crowd Supply Infinite Noise TRNG," *9th International Symposium on Digital Forensics and Security, ISDFS 2021*, 2021, doi: 10.1109/ISDFS52919.2021.9486317.

[167] G. Kim, H. Kim, Y. Heo, Y. Jeon, and J. Kim, "Generating Cryptographic S-Boxes Using the Reinforcement Learning," *IEEE Access*, vol. 9, pp. 83092–83104, 2021, doi: 10.1109/ACCESS.2021.3085861.

[168] M. Yang, Z. Wang, Q. Meng, and L. Han, "Evolutionary design of S-box with cryptographic properties," *Proceedings - 9th IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops, ISPAW 2011 - ICASE 2011, SGH 2011, GSDP 2011*, pp. 12–15, 2011, doi: 10.1109/ISPAW.2011.59.

[169] A. F. Kadhim and Z. A. Kamal, "Generating dynamic S-BOX based on Particle Swarm Optimization and Chaos Theory for AES," *Iraqi Journal of Science*, vol. 59, no. 3, pp. 1733–1745, 2018, doi: 10.24996/IJS.2018.59.3C.18.

[170] M. D. Gupta and R. K. Chauhan, "Secure image encryption scheme using 4D-Hyperchaotic

systems based reconfigurable pseudo-random number generator and S-Box," *Integration*, vol. 81, no. July, pp. 137–159, 2021, doi: 10.1016/j.vlsi.2021.07.002.

[171] M. Zhao, H. Liu, and Y. Niu, "Batch generating keyed strong S-Boxes with high nonlinearity using 2D hyper chaotic map," *Integration*, vol. 92, no. February, pp. 91–98, 2023, doi: 10.1016/j.vlsi.2023.05.006.

[172] S. Arrag, A. Hamdoun, A. Tragha, and E. Khamlich Salah, "Implementation of stronger AES by using dynamic S-box dependent of master key," *Journal of Theoretical and Applied Information Technology*, vol. 53, no. 2, pp. 196–204, 2013.

[173] Alamsyah, "Improving the Quality of AES S-box by Modifications Irreducible Polynomial and Affine Matrix," *2020 5th International Conference on Informatics and Computing, ICIC 2020*, 2020, doi: 10.1109/ICIC50835.2020.9288567.

[174] Alamsyah, B. Prasetiyo, and M. N. Ardian, "Enhancement security AES algorithm using a modification of transformation ShiftRows and dynamic S-box," *Journal of Physics: Conference Series*, vol. 1567, no. 3, 2020, doi: 10.1088/1742-6596/1567/3/032025.

[175] G. Manjula and H. S. Mohan, "Constructing key dependent dynamic S-Box for AES block cipher system," *Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016*, no. 10, pp. 613–617, 2017, doi: 10.1109/ICATCCT.2016.7912073.

[176] P. Agarwal, A. Singh, and A. Kilicman, "Development of key-dependent dynamic S-Boxes with dynamic irreducible polynomial and affine constant," *Advances in Mechanical Engineering*, vol. 10, no. 7, pp. 1–18, 2018, doi: 10.1177/1687814018781638.

[177] B. B. Cassal-Quiroga and E. Campos-Cantón, "Generation of Dynamical S-Boxes for Block Ciphers via Extended Logistic Map," *Mathematical Problems in Engineering*, vol. 2020, 2020, doi: 10.1155/2020/2702653.

[178] H. Aguirre, H. Okazaki, and Y. Fuwa, "An evolutionary multiobjective approach to design highly non-linear Boolean functions," *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*, pp. 749–756, 2007, doi: 10.1145/1276958.1277112.

[179] S. Picek, D. Jakobovic, and M. Golub, "Evolving cryptographically sound boolean functions," *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion*, pp. 191–192, 2013, doi: 10.1145/2464576.2464671.

[180] K. Kalaiselvi and A. Kumar, "A Novel Method to Design S-box Based on Genetic Algorithm and Particle Swarm Optimizationin AES-128 Cryptosystem," vol. 118, no. 18, pp. 1443–1457, 2018.

[181] H. Alsaif, R. Guesmi, A. Kalghoum, B. M. Alshammari, and T. Guesmi, "A Novel Strong S-Box Design Using Quantum Crossover and Chaotic Boolean Functions for Symmetric Cryptosystems," *Symmetry*, vol. 15, no. 4, pp. 1–23, 2023, doi: 10.3390/sym15040833.

[182] ID Quantique, "Understanding Quantum Cryptography." ID Quantique SA, 2020, Accessed:

Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/quantum-safe-security/quantum-key-distribution/.

[183] G. Sosa-Gómez, O. Rojas, and O. Páez-Osuna, "Using hadamard transform for cryptanalysis of pseudo-random generators in stream ciphers," *EAI Endorsed Transactions on Energy Web*, vol. 7, no. 27, pp. 1–7, 2020, doi: 10.4108/eai.13-7-2018.163980.

[184] S. Nakov, "secure-random-generators." https://cryptobook.nakov.com/secure-random-generators (accessed Sep. 03, 2021).

[185] "Cryptol." Galois, Inc., [Online]. Available: https://cryptol.net/.

[186] C. L. Chiang, *Statistical Methods of Analysis*. 2003.

[187] D. Bogdanov, L. Kamm, S. Laur, and V. Sokk, "Rmind: A tool for cryptographically secure statistical analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 481–495, 2018, doi: 10.1109/TDSC.2016.2587623.

[188] M. Masdari and M. Jalali, "A survey and taxonomy of DoS attacks in cloud computing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3724–3751, 2016, doi: 10.1002/sec.1539.

[189] M. Bazm, M. Lacoste, M. Südholt, and J. Menaud, "Side Channels in the Cloud : Isolation Challenges , Attacks , and Countermeasures," 2017.

[190] R. Kumar and M. P. S. Bhatia, "A systematic review of the security in cloud computing: Data integrity, confidentiality and availability," *2020 IEEE International Conference on Computing, Power and Communication Technologies, GUCON 2020*, pp. 334–337, 2020, doi: 10.1109/GUCON48875.2020.9231255.

[191] A. Markandey, P. Dhamdhere, and Y. Gajmal, "Data access security in cloud computing: A review," *2018 International Conference on Computing, Power and Communication Technologies, GUCON 2018*, pp. 633–636, 2019, doi: 10.1109/GUCON.2018.8675033.

[192] A. J. Duncan, S. Creese, and M. Goldsmith, "Insider attacks in cloud computing," *Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*, pp. 857–862, 2012, doi: 10.1109/TrustCom.2012.188.

[193] N. Rakotondravony *et al.*, "Classifying malware attacks in IaaS cloud environments," *Journal of Cloud Computing*, vol. 6, no. 1, 2017, doi: 10.1186/s13677-017-0098-8.

[194] V. E. Urias, B. Van Leeuwen, W. M. S. Stout, and H. Lin, "Applying a Threat Model to Cloud Computing," *Proceedings - International Carnahan Conference on Security Technology*, vol. 2018-Octob, 2018, doi: 10.1109/CCST.2018.8585471.

[195] H. Demirhan, "CryptRndTest: Statistical Tests for Cryptographic Randomness." https://cran.r-project.org/web/packages/CryptRndTest/index.html.

[196] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *Nist Special Publication*, vol. 22, no.

April, pp. 1/1--G/1, 2010, [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf.

[197] G. Marsaglia, "Diehard: A Battery of Tests of Randomnes," 1996.

[198] M. M. Chatzimichailidou and I. M. Dokas, "RiskSOAP: On the Relationship between Systems Safety and the Risk SA Provision Capability," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1148–1157, 2018, doi: 10.1109/JSYST.2016.2614953.

[199] P. Socha, V. Miskovsky, H. Kubatova, and M. Novotny, "Optimization of Pearson correlation coefficient calculation for DPA and comparison of different approaches," in *International Symposium on Design and Diagnostics of Electronic Circuit and Systems*, Apr. 2017, pp. 184–189, doi: 10.1109/DDECS.2017.7934563.

[200] T. S. Community, "hamming." https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html (accessed Jul. 09, 2020).

[201] E. Volchok, "Clear-Sighted Statistics: Module 14: One-Sample Hypothesis Tests (slides)." City University of New York (CUNY)., 2020.

[202] S. Vajapeyam, "Understanding Shannon's Entropy metric for Information," no. March, pp. 1–6, Mar. 2014, doi: https://doi.org/10.48550/arXiv.1405.2061.

[203] G. J. Croll, "Bientropy, TriEntropy and primality," *Entropy*, vol. 22, no. 3, Mar. 2020, doi: 10.3390/e22030311.

[204] S. Tiun, M. Ayob, M. A. AL-DhAlbadr, and F. Ief, "Genetic Algorithm Based on Natural Selection Theory for Optimization Problems," *Symmetry*, vol. 12, no. 11, p. 1758, 2020.

[205] D. Singh and R. Prakash, "Understanding Evolutionary Based Genetic Algorithm and Implementation using R," *SSRN Electronic Journal*, 2022, doi: 10.2139/ssrn.3970486.

[206] M. Jane, H. Bruno, R. Moulds, W. Nino, and F. Anthony, "Quantum-Safe Security Working Group Quantum Random Number Generators." Cloud Security Alliance, Bellingham, WA, USA, 2016.

[207] IDQ, "Quantum versus Classical Random Number Generators." Switzerland, May 2020.

[208] ID Quantique, "gaming-and-lotteries." Accessed: Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/random-number-generation/applications/gaming-and-lotteries/.

[209] M. R. Albrecht, "Algebraic Techniques in Cryptanalysis," *Ecrypt2[Ppt]*, vol. 3, no. 61070244, pp. 120–141, 2011.

[210] N. G. Bardeh and S. Rønjom, "Practical attacks on reduced-round AES," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11627 LNCS, pp. 297–310, 2019, doi: 10.1007/978-3-030-23696-0_15.

[211] B. Abd-El-Atty, "Efficient S-box construction based on quantum-inspired quantum walks with PSO algorithm and its application to image cryptosystem," *Complex and Intelligent Systems*,

2023, doi: 10.1007/s40747-023-00988-7.

[212] K. Kalaiselvi and A. Kumar, “Enhanced AES cryptosystem by using genetic algorithm and neural network in S-box,” *2016 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2016*, 2016, doi: 10.1109/ICCTAC.2016.7567340.

[213] D. Zhu, M. Zhang, X. Tong, and Z. Wang, “A Novel S-box Optimization Method Based on Immune Genetic Algorithm,” *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, vol. 2020-Octob, pp. 32–35, 2020, doi: 10.1109/ICSESS49938.2020.9237665.

[214] D. Zhu, X. Tong, M. Zhang, and Z. Wang, “A new s-box generation method and advanced design based on combined chaotic system,” *Symmetry*, vol. 12, no. 12, pp. 1–17, 2020, doi: 10.3390/sym12122087.

[215] N. T. Courtois, J. A. Gawinecki, and G. Song, “Contradiction Immunity and Guess-Then-Determine Attacks on Gost,” *Tatra Mountains Mathematical Publications*, vol. 53, no. 1, pp. 65–79, Dec. 2013, doi: 10.2478/v10127-012-0039-3.

[216] N. N. Anandakumar and S. Dillibabu, “Correlation power analysis attack of AES on FPGA using customized communication protocol,” in *International Conference on Computational Science, Engineering and Information Technology*, Oct. 2012, pp. 683–688, doi: 10.1145/2393216.2393330.

[217] Y. Niu, J. Zhang, A. Wang, and C. Chen, “An Efficient Collision Power Attack on AES Encryption in Edge Computing,” *IEEE Access*, vol. 7, pp. 18734–18748, 2019, doi: 10.1109/ACCESS.2019.2896256.

[218] Y. Li, M. Chen, Z. Liu, and J. Wang, “Reduction in the number of fault injections for blind fault attack on SPN block ciphers,” *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, pp. 1–20, Apr. 2016, doi: 10.1145/3014583.

[219] T. S. D. Team, “Sage.” https://doc.sagemath.org/html/en/reference/cryptography/index.html (accessed Nov. 14, 2022).

[220] S. Picek *et al.*, “S-box , SET , Match : A Toolbox for S-box Analysis To cite this version : HAL Id : hal-01400936,” pp. 0–10, 2016.

[221] D. YongWang, “S-Box Performance Analysis.” http://42.192.236.76/Login (accessed Mar. 17, 2023).

[222] M. Ahmad, N. Mittal, P. Garg, and M. Maftab Khan, “Efficient cryptographic substitution box design using travelling salesman problem and chaos,” *Perspectives in Science*, vol. 8, pp. 465–468, Sep. 2016, doi: 10.1016/j.pisc.2016.06.001.

[223] A. H. Zahid, M. J. Arshad, and M. Ahmad, “A novel construction of efficient substitution-boxes using cubic fractional transformation,” *Entropy*, vol. 21, no. 3, p. 245, Mar. 2019, doi: 10.3390/e21030245.

[224] Attaullah, S. S. Jamal, and T. Shah, “A Novel Algebraic Technique for the Construction of

Strong Substitution Box," *Wireless Personal Communications*, vol. 99, no. 1, pp. 213–226, 2018, doi: 10.1007/s11277-017-5054-x.

[225] A. H. Zahid and M. J. Arshad, "An innovative design of substitution-boxes using cubic polynomial mapping," *Symmetry*, vol. 11, no. 3, 2019, doi: 10.3390/sym11030437.

[226] A. Ejaz, I. A. Shoukat, U. Iqbal, A. Rauf, and A. Kanwal, "A secure key dependent dynamic substitution method for symmetric cryptosystems," *PeerJ Computer Science*, vol. 7, pp. 1–29, 2021, doi: 10.7717/PEERJ-CS.587.

[227] W. Gao, B. Idrees, S. Zafar, and T. Rashid, "Construction of Nonlinear Component of Block Cipher by Action of Modular Group PSL(2, Z) on Projective Line PL(GF(2 8))," *IEEE Access*, vol. 8, pp. 136736–136749, 2020, doi: 10.1109/ACCESS.2020.3010615.

[228] A. A. Alzaidi, M. Ahmad, H. S. Ahmed, and E. Al Solami, "Sine-Cosine Optimization-Based Bijective Substitution-Boxes Construction Using Enhanced Dynamics of Chaotic Map," *Complexity*, vol. 2018, 2018, doi: 10.1155/2018/9389065.

[229] Q. Lu, C. Zhu, and G. Wang, "Compound Chaotic System," pp. 1–15, 2019.

[230] M. Ahmad, M. N. Doja, and M. M. S. Beg, "ABC Optimization Based Construction of Strong Substitution-Boxes," *Wireless Personal Communications*, vol. 101, no. 3, pp. 1715–1729, 2018, doi: 10.1007/s11277-018-5787-1.

[231] X. Wang and J. Yang, "A novel image encryption scheme of dynamic S-boxes and random blocks based on spatiotemporal chaotic system," *Optik*, vol. 217, no. February, p. 164884, 2020, doi: 10.1016/j.ijleo.2020.164884.

[232] A. Belazi and A. A. A. El-Latif, "A simple yet efficient S-box method based on chaotic sine map," *Optik*, vol. 130, pp. 1438–1444, 2017, doi: 10.1016/j.ijleo.2016.11.152.

[233] H. Liu, A. Kadir, and C. Xu, "Cryptanalysis and constructing S-Box based on chaotic map and backtracking," *Applied Mathematics and Computation*, vol. 376, 2020, doi: 10.1016/j.amc.2020.125153.

[234] R. Soto, B. Crawford, F. G. Molina, and R. Olivares, "Human Behaviour Based Optimization Supported with Self-Organizing Maps for Solving the S-Box Design Problem," *IEEE Access*, vol. 9, pp. 84605–84618, 2021, doi: 10.1109/ACCESS.2021.3087139.

[235] M. F. Khan, K. Saleem, T. Shah, M. M. Hazzazi, I. Bahkali, and P. K. Shukla, "Block Cipher's Substitution Box Generation Based on Natural Randomness in Underwater Acoustics and Knight's Tour Chain," *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/8338508.

[236] U. Satapathy, B. K. Mohanta, D. Jena, and S. Sobhanayak, "An ECC based Lightweight Authentication Protocol for Mobile Phone in Smart Home," *2018 13th International Conference on Industrial and Information Systems, ICIIS 2018 - Proceedings*, pp. 303–308, 2018, doi: 10.1109/ICIINFS.2018.8721417.

# Appendix I : Published paper I - Quantum Randomness in Cryptography—A Survey of Cryptosystems, RNG-Based Ciphers, and QRNGs

information

IMPACT FACTOR 2.9

CITESCORE 6.5

Review

# Quantum Randomness in Cryptography—A Survey of Cryptosystems, RNG-Based Ciphers, and QRNGs

Anish Saini, Athanasios Tsokanos and Raimund Kirner

MDPI

https://doi.org/10.3390/info13080358

information

*Review*

# Quantum Randomness in Cryptography—A Survey of Cryptosystems, RNG-Based Ciphers, and QRNGs

**Anish Saini *, Athanasios Tsokanos and Raimund Kirner**

School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK; a.tsokanos@herts.ac.uk (A.T.); r.kirner@herts.ac.uk (R.K.)
* Correspondence: a.saini@herts.ac.uk

**Abstract:** Cryptography is the study and practice of secure communication with digital data and focuses on confidentiality, integrity, and authentication. Random number generators (RNGs) generate random numbers to enhance security. Even though the cryptographic algorithms are public and their strength depends on the keys, cryptoanalysis of encrypted ciphers can significantly contribute to the unveiling of the cipher's key. Therefore, to ensure high data security over a network, researchers need to improve the randomness of keys as they develop cryptosystems. Quantum particles have a leading edge in advancing RNG technology as they can provide true randomness, unlike pseudo-random numbers generators (PRNGs). In order to increase the level of the security of cryptographic systems based on random numbers, this survey focuses on three objectives: Cryptosystems with related cryptographic attacks, RNG-based cryptosystems, and the design of quantum random number generators (QRNGs). This survey aims to provide researchers with information about the importance of RNG-based ciphers and various research techniques for QRNGs that can incorporate quantum-based true randomness in cryptosystems.

**Keywords:** cryptosystems; cryptoanalysis; RNG-based cipher; QRNG

check for updates

**Citation:** Saini, A.; Tsokanos, A.; Kirner, R. Quantum Randomness in Cryptography—A Survey of Cryptosystems, RNG-Based Ciphers, and QRNGs. *Information* **2022**, *13*, 358. https://doi.org/10.3390/info13080358

Academic Editor: Willy Susilo

Received: 10 May 2022
Accepted: 6 July 2022
Published: 27 July 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Information security is the principal concern whenever data is transmitted over a network, a confined physical space of connected digital equipment, or a public network, such as the Internet. The information is considered secure if it cannot be understood by someone other than the intended recipient. The unintended person trying to steal the information is referred to as a hacker. In cryptography [1], one part of cryptology, such information is termed encrypted data, and the cryptosystem is designed with the primary concern of confidentiality and a security measure to safeguard the information from unauthorized access. The cryptosystem includes three significant processes: the Key Schedule Algorithm (KSA), the Encryption Algorithm, and the Decryption Algorithm.

The KSA is a process of generating keys for encryption and decryption. It takes the user-defined or original key as a set of bits, such as 40, 128, 192, 256, or more significant bits, to expand them based on its processing steps or the number of rounds designed for the encryption algorithm. The purpose of the KSA is to make the key so strong that it is not vulnerable to attack, and hackers cannot find the original key.

The Encryption Algorithm (EA) encrypts data by using a key and converts it to an unreadable format, and this process is called encryption. On the other hand, a Decryption Algorithm (DA) involves using the same or different key for decoding the cipher and converting the cipher back into the original data, which is called decryption. Figure 1a,b shows the three processes.

Encryption transforms plaintext into the ciphertext and secure ciphers by associating various cryptographic properties such as nonlinearity, propagation criteria, correlation, and algebraic immunity. However, the security of a cipher depends on how vulnerable the

used key is to a cryptanalysis attack [2]. In addition to KSA, which divides the key into subkeys, random numbers can be used to make the key more robust and complex.

Figure 2 shows the KSA and encryption of a cryptosystem with a random number generator (RNG). RNGs provide the random numbers or bits used in achieving randomness in a cryptosystem. Encryption encrypts the plain text into ciphertext with advancement in an RNG-based cryptosystem; the KSA key also incorporates the random number bits generated by an RNG.

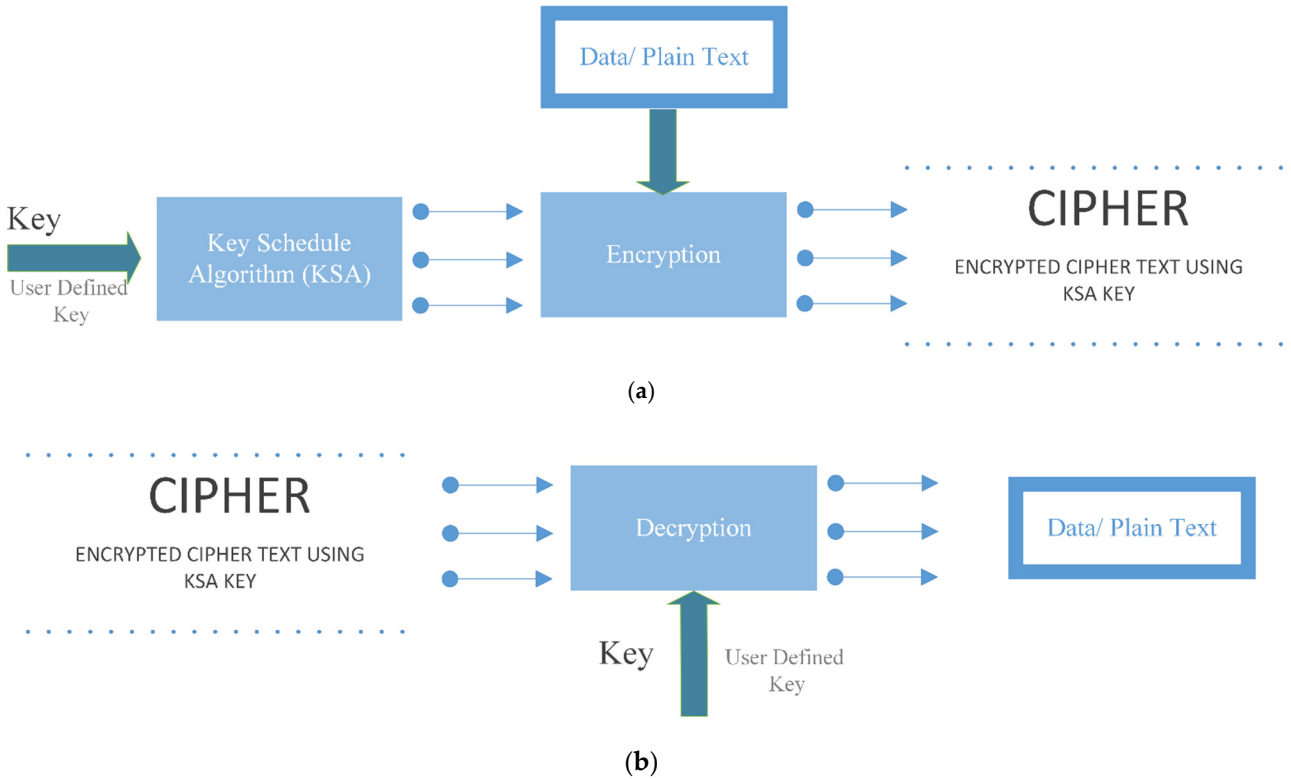



**Figure 1.** (**a**) Key Schedule Algorithm (KSA) and encryption of a cryptosystem; (**b**) decryption of a cryptosystem.

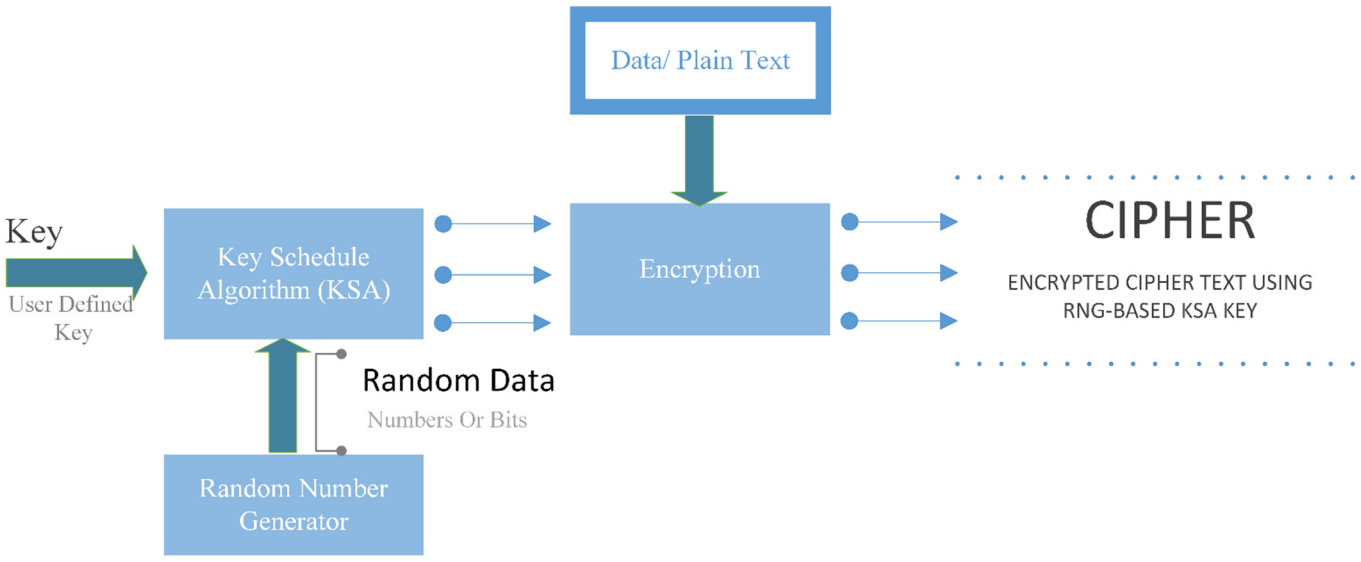



**Figure 2.** Key Schedule Algorithm (KSA) and encryption of an RNG-based cryptosystem.

RNGs' randomness critically depends on the type of RNG.

Pseudorandom number generators (PRNGs) [3,4] are based on algorithms for generating seemingly random numbers, which are determined by their seed, or initial value to enhance security. Shobhit Sinha et al. [5] compared various PRNGs based on their statistical randomness and cryptographic security, concluding that some PRNGs performed significantly worse than others.

Hardware and portable USB-based true random number generators (TRNGs) generate a truly random number based on digital noise intervention. The entropy or randomness source for TRNGs is usually electrical noise and an unpredictable component of electronic systems. Lishuang et al. [6] compared the advantages and challenges of entropy sources that use electrical noise.

True (non-pseudo) random data [7,8] can also be achieved by using a quantum random number generator (QRNG) based on quantum particles. Random number generators with quantum [9] randomness are superior to pseudo or true RNGs, as their source of randomness is invulnerable to environmental disturbance, such as temperature, voltage, or current and provides the highest level of entropy [10].

In this survey, we discuss and categorise various studies on cryptography and RNGs. In the first category, we study cryptosystems and cryptographic cryptoanalysis of the ciphers. Our second category focuses on different RNG-based cryptosystems in light of the RNGs' robustness. This category results help analyze the enhancement in the security of a cryptosystem. We will discuss in the last category the multiple features of QRNGs, analyzing how the QRNGs differ from each other and which of the QRNG features are best suited for any given application.

Lastly, this paper discusses and proposes open research problems for a cipher function, which relates to the randomness of quantum random numbers (QRN).

This research aims to contribute to the future of cryptography and to become a part of the open quantum-safe project [11]. The survey is organised as follows: Section II provides a survey methodology, categorised into three subsections. Section III focuses on open research problems of ciphers based on QRNGs. The paper is concluded in Section IV.

## 2. Survey Methodology

The survey methodology is subdivided into three categories:

A. Category I: Cryptosystems and cryptographic attacks
B. Category II: RNG-based cryptosystems
C. Category III: Research objectives of quantum-RNGs for cryptosystems

### *2.1. Category I: Cryptosystems and Cryptographic Attacks*

In cryptology, a cryptography algorithm is an art of creating codes or algorithms that turn plain text into ciphertext and ciphertext into the original text. It establishes secure communication between two entities in the public domain, such as the Internet, where unauthorized users and information hackers are present.

The main difference between the cryptography algorithm is the relationship between the encryption and the decryption key. Logically, in any cryptographic system known as a cryptosystem, keys generated and expanded by KSA play an essential role in the cryptographic process.

A private key is used for both encryption and decryption, and algorithms are classified as symmetric key cryptography, whereas asymmetric key cryptography uses a pair of public and private keys for encryption and decryption or signing and verification.

Figure 3a,b visualizes symmetric key cryptography and asymmetric key cryptography, respectively.

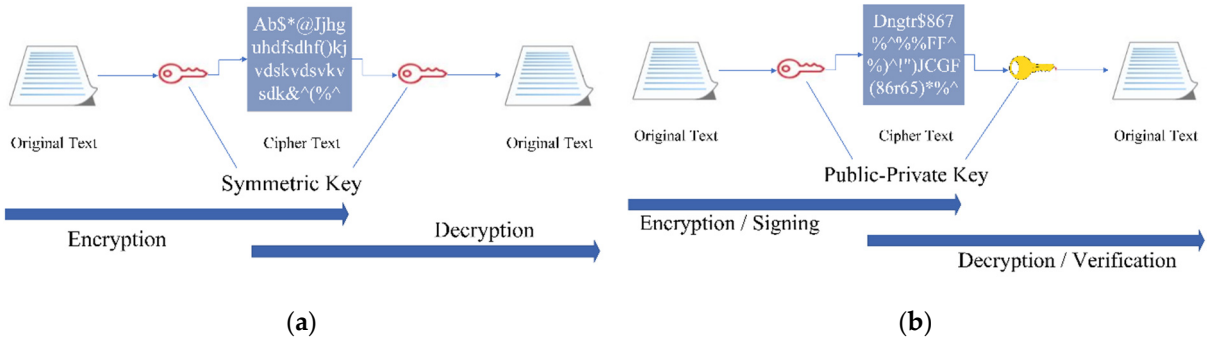



**Figure 3.** (**a**) Symmetric key cryptography; (**b**) Asymmetric key cryptography.

There are two types of symmetric key cryptography, block cipher and stream cipher.

Block cipher: A block cipher encrypts or decrypts a block of data. Encryption transforms plaintext into an equal length of ciphertext block, whereas the decryption performs the reverse process on the same block of ciphertext block. It primarily uses either of the following two network structures.

Feistel (F)-Network: In F-Network, the input data splits into two parts—the left and the right. The right part remains unchanged, and the left part goes through the operation with the key. After processing several rounds of operations designed by the algorithm, the combination of the left and right parts constitutes the ciphertext.

Substitution-permutation (SP) network: In the SP network, the substitution layer consists of a substitution (S)-box and operates on short data segments. Due to the layer's highly nonlinear property, it creates confusion in the internal ciphertext. The permutation layer diffuses the effect of substitution across the entire block.

Stream cipher: A stream cipher performs encryption or decryption on a digital data stream of 1 byte or 1 bit. When data is to be transmitted over a communication channel or through a web browser, this can be helpful. It depends on different structures broadly categorized as follows: arithmetic and XOR(AX), linear feedback shift register (LFSR), a combination of LFSR and nonlinear-feedback shift register (LFSR+NFSR), and pseudo-random function (PF)

Arithmetic and XOR(AX): Arithmetic operations include all bitwise operations (XOR, NAND, OR, Modulo operator) for cryptographic functions.

LFSR: LFSR is a shift register where the input bit is a previous state's linear function (exclusive-or (XOR)). The input is affected by previous stage values, so it operates as a feedback mechanism.

NFSR: NFSR is a shift register where the input bit is a previous state's non-linear function. The stream cipher also uses the LFSR+NFSR structure, which combines the LFSR and the NFSR.

PF: These functions are based on pseudo-random numbers generated by PRNGs.

Table 1 summarises the symmetric key cryptography categorized into block cipher and stream ciphers.

Equally important, asymmetric key cryptography, based on a pair of keys, is categorized into two approaches: public key cryptography and digital signature.

Public key cryptography: A public key cryptography system uses a pair of public and private keys. The KSA generates both the keys—a public key for the encryption and a private key for the decryption. Encryption is performed by the receiver's public key, generating the ciphertext, and the ciphertext can only be decrypted with the sender's private key. Factorization and Diffie–Hellman's (DH) key exchange are the two main techniques of public key cryptography.

Factorization: In factorization, a composite number, part of mathematical computation, is transformed into a product of smaller integers, whereas when the integers are primes, the process is known as prime factorization.

**Table 1.** List of symmetric key cryptography systems.

| *Symmetric-Key Cryptography* **(A Secret Key Generated by KSA for Encryption & Decryption)** | | | |
|---|---|---|---|
| Block Ciphers | | Stream Ciphers | |
| F Network | SP Network | AX | LFSR |
| 1900's | 1900's | 1900's | 1900's |
| — DES [12] | | — RC4 [17] | — A5/1 [18] |
| — 3 DES [12] | | | — A5/2 [19] |
| — IDEA [20] | | 2000's | |
| — Blowfish [21] | | | 2000's |
| — Tea [27] | — Safer-K [13] | | |
| — CAST-128 [28] | — 3-Way [14] | — Rabbit [22] | |
| | — SHARK [15] | — SOBER-128 [23] | — SNOW 2 [29] |
| — XTEA [27] | — AES [16] | — QUAD [24] | |
| | | — Trivium [25] | |
| — RC2 [30] | | — Salsa20 [26] | — Crypto-1 [31] |
| — RC6 [32] | | | |
| — CAST-256 [28] | | LFSR+NFSR | PF |
| — MARS [33] | | | |
| — Twofish [34] | | 2000's | 1900's |
| 2000's | | | — ISAAC [17] |
| — Camellia, 2000 [38] | | — HC-256 [35] | — SEAL [39] |
| | | — Grain [36] | |
| — Threefish [40] | | — MICKEY [37] | 2000's |
| — GOST [41] | | | — Phelix [42] |

Diffie–Hellman (DH) key exchange: DH is a cryptographic set of rules for exchanging the private key over an unsecured channel. The KSA of both the sender and receiver generates the public and private key pair. Both the parties share their public key. Once they get the public key, they calculate their secret or private key and use it for sending data securely.

Post-quantum public key cryptography: The cryptoanalysis of classical ciphers led researchers to develop quantum and classical computer-resistant cryptosystems. Moreover, these systems can communicate with an existing communication protocol, allowing their practical implementation for applications. The National Institute of Standards and Technology (NIST) [43] announced the four post-quantum public key encryption and key-establishment algorithms in round 3 compared to seventeen algorithms in round 2. NIST has started the process of standardizing these algorithms.

Digital signature: The digital signature is an electronic signature in which the sender signs or encrypts the document with a private key. The receiver will decrypt or verify that document with the sender's public key. It can be incorporated into cryptography by using the digital signature standard or elliptic curve cryptography.

Digital signature standard (DSS): DSS is a standard established by NIST to generate digital signatures.

Elliptic curve cryptography (ECC): A cryptographic algorithm based on elliptic curves generates smaller key sizes while providing the same level of security as those without them.

Post-quantum digital signature: NIST announced the three post-quantum digital signature algorithms in round 3 compared to nine algorithms in round 2. These are

another part of post-quantum cryptography. Furthermore, NIST has started the process of standardizing these algorithms.

Table 2 shows the asymmetric key approaches used by public key cryptography and digital signatures.

**Table 2.** List of asymmetric key cryptography systems.

| *Asymmetric-Key Cryptography* **(A Public and a Private Key Generated by KSA for Encryption/Signing & Decryption/Verification)** | | | |
|---|---|---|---|
| Public-Key cryptography | | Digital Signature | |
| Factorization | DH-Key Exchange | DSS | ECC |
| | | | 2000's |
| — RSA, [44] 1978 | — ElGamal, [45] 1985 | — DSA, [46] 1994 | — ECIES, [47]<br>— ECDSA, [47]<br>— EdDSA, [48] |
| Post-Quantum Public-Key cryptography [43] | | Post-Quantum Digital Signature [43] | |
| — BIKE | | — CRYSTALS-DILITHIUM | |
| — CRYSTALS-KYBER | — Classic McEliece | — GeMSS | |
| — HQC | — FrodoKEM | — MQDSS | |
| — LEDAcrypt | — LAC | — qTESLA | |
| — NTRU | — NewHope | — SPHINCS+ | |
| — NTS-KEM | — NTRU Prime | — FALCON | |
| — SIKE | — ROLLO | — LUOV | |
| — RQC | — Three Bears | — Picnic | |
| — Round5 | — SABER | — Rainbow | |

The security of symmetric key or asymmetric key depends on the vulnerability of their key to the cryptographic attacks. Cryptoanalysis analyzes cryptosystems to look for weak points or opportunities for information leaks to access or find the key. Cryptoanalysis is another part of cryptology that differentiates different cryptographic attacks. We have categorised the cryptographic attacks into six categories: differential cryptoanalysis, linear cryptoanalysis, meet-in-the-middle, side-channel attacks, related key attacks and other attacks.

Differential cryptoanalysis (DC): DC refers to a chosen-plaintext attack, where the attacker can choose a plaintext and find the corresponding ciphertext in order for them to obtain the key. By computing the differences between the ciphertexts, the attacker can detect statistical patterns in their distribution. These differences constitute a differential attack.

Linear cryptoanalysis (LC): LC finds affine approximations to the cipher's action based on linear equations. Depending on linear equations, it comes close to a plaintext–ciphertext pair and attempts to find the key.

Meet-in-the-middle (Mt-in-M): An Mt-in-M attack is a known-plaintext attack that breaks the long chain of encryption blocks into half to analyze the blocks independently and find the key more accurately. This accuracy cost depends on breaking the encryption chain into smaller parts requiring more storage. However, it is still more efficient than a brute force attack regarding time and computational complexity.

Side-channel attacks (SCA): In side-channel attacks, information leaks from a physical cryptosystem. In addition to timing, power consumption and electromagnetic emissions can also be exploited.

Related key Attacks (RKA): An attack model called related key is a subset of cryptoanalytic attacks in which the attacker is able to select or identify the relationship between multiple keys. These keys are then used for both encryption and decryption operations.

Other attacks: Other attacks, apart from DC, LC, M-in-M, SCA, and RKA, are included in this category.

Table 3 shows the survey of cryptographic attacks on different ciphers with the key size of KSA.

**Table 3.** Cryptanalysis of cryptography ciphers.

| | | ***Cryptoanalysis*** | | | | | |
|---|---|---|---|---|---|---|---|
| ***Ciphers*** | ***KSA Key Size (Bits)*** | ***DC*** | ***LC*** | ***Mt-in-M*** | ***SCA*** | ***RKA*** | ***Others*** |
| DES | 56 | √ | √ | √ | | | Brute-Force |
| 3 DES | 112 or 168 | | | | | | Sweet32 |
| IDEA | 128 | | | Bicliques Attack | | | |
| Blowfish | 32–448 | | | | | | Birthday Attack |
| Tea | 128 | √ | | | | √ | |
| GOST | 256 | √ | | Reflection | | √ | |
| CAST 128 | 40 to 128 | √ | | | | | |
| XTEA | 128 | √ | | √ | | √ | |
| RC2 | 1–128 Bytes | | | | | √ | |
| RC6 | 128, 192, 256 | | | | | | Statistics Attack |
| CAST 256 | 128, 160, 192, 224, 256 | √ | | | | | |
| Mars | 128, 192, 256 | | | √ | | | |
| Twofish | 128, 192, 256 | Truncated, Impossible | | | Power Analysis | √ | |
| Camellia | 128, 192, 256 | | | | Cache Timing | | Square Attack |
| Threefish | 256, 512, 1024 | Boomerang Attack | | | | | |
| Safer-K | 64,128 | Boomerang Attack, Impossible | | | | √ | |
| 3-Way | 96 | | | | | √ | |
| Serpent | 128, 192, 256 | Differential-Linear | | | Power Analysis | | |
| AES | 128, 192, 256 | | | Bicliques Attack | | √ | Brute-Force |
| RSA | 2048 to 4096 | | | | | | Shor's Algorithm |

The survey shows that different attacks on a cryptosystem are possible, even on an advanced encryption system (AES), a highly secure block cipher. These attacks illustrate a requirement to modify various cryptosystems to make them more secure against cryptographic attacks. Figure 4 represents the graphic view of the cipher's cryptoanalysis by the specific cryptographic attacks.

In the next category, we have discussed different RNGs and have focused on cryptosystems based on RNGs.

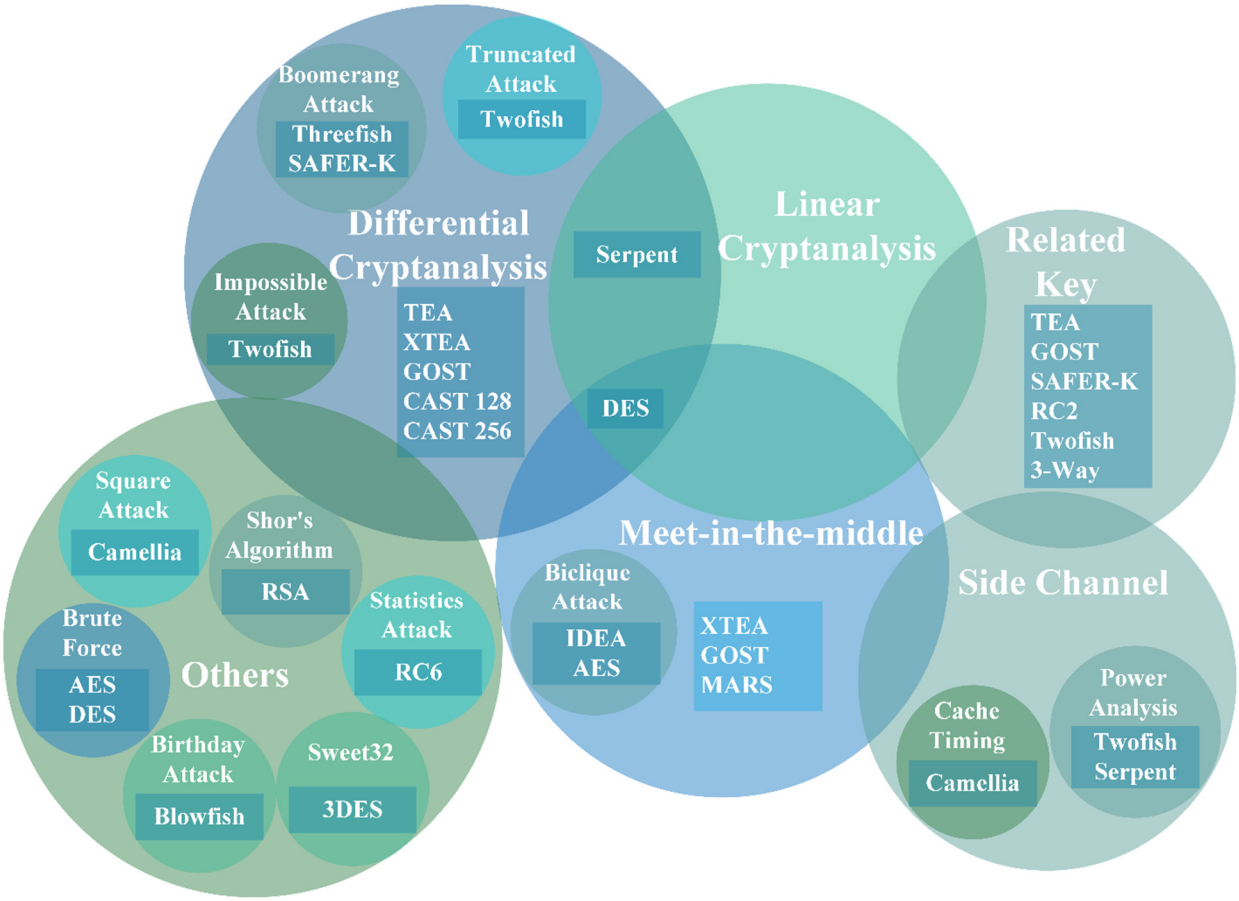



**Figure 4.** Graphic review of cryptographic cryptoanalysis.

### *2.2. Category II: RNG-Based Cryptosystems*

The cryptosystems are affected by the objectives and operations to perform their functions. The RNGs can play a significant role in providing cryptography's fundamental function, randomness, and enhancement of the security of the cryptosystem. The randomness can be achieved by different random method generators.

- Pseudo-random number generator (PRNG)
- True random number generator (TRNG)
- Circuit design-based random number generator (CDRNG)
- Quantum random number generator (QRNG)

Pseudo-random number generator (PRNG): PRNGs are computer-based algorithms that use mathematical computation to generate random numbers. A PRNG sequence of the random number is periodic, which means the sequence repeats itself periodically, and deterministic, in that the sequence reproduction is possible if the initial value is known. Mathematical computations like linear congruential generators, LFSRs, and chaos provide pseudo-randomness.

A PRNG is a cryptographic secure PRNG, or CSPRNG if it passes the next-bit test (if the attacker knows the first k bits, the attacker cannot predict the k + 1 bits) and withstands the state compromise extensions (if any stream of bits is guessed correctly, the last bits cannot be predicted).

Programming languages have techniques for generating CSPRNG. The secrets [49] module from Python and SecureRandom [4] class from Java support CSPRNG.

True random number generator (TRNG): Random numbers generated by TRNGs are unbiased, independent, and unpredictable as produced by different physical phenomena

such as infinite noise, voltage fluctuation, clock jitter, atmospheric radio noise, continuous and discrete time chaotic system.

There are various USB interface-based TRNGs that help to quickly connect with a laptop to work with any application. Alea II [50] by Araneus Information Systems, TrueRNG v3 [51] by ubldit, and infinite noise TRNG [52] by Crowd Supply are some of the USB-based TRNGs. On the other hand, RANDOM.ORG is a service-oriented online platform that provides TRNGs for various applications such as games, lotteries, and web pages.

Circuit design-based random number generator (CDRNG): Circuit design, including various electronic components, such as gates, integrated circuits, etc., is also helpful in generating RNGs. Pietro Nannipieri et al. [53] proposed TRNGs using the Fibonacci–Galois ring oscillator for cryptographically secure applications on a field programmable gate array (FPGA), resulting in the best entropy. Luca Crocetti et al. [54] proposed an all-digital RNG with high portability and entropy. As a result, it is ready to integrate with the European Processor Initiative (EPI) [55] chip.

Quantum random number generator (QRNG): The third classification of RNG is quantum-based and derived from quantum mechanics. Different QRNGs generate established and robust outcomes by applying different entropy sources, the source of randomness, based on quantum physics.

USB-based QRNGs are well tested and certified for generating high-quality and unique random numbers such as Quantis [56] by IDQ. They also invented a chip-based QRNG that can fit into small devices like smartphones. Another project named QRANGE [57] under the quantum flagship focused on CMOS technology generating quantum random numbers. ANU QRNG [58] is an online platform where anybody can generate quantum random numbers with just one click and use them.

Figure 5 illustrates the categories of RNGs and their implementation methodology or practical approach for integrating them into a cryptosystem.

These RNGs can be used in the RNG-based cryptosystem to enhance the security in the KSA or encryption. Table 4 shows the survey of RNG-based cryptosystems, improving different parameters due to the random numbers generated by incorporating RNGs. Moreover, the survey indicates that RNGs can be used with any cryptosystem and make it more secure than the system without RNGs.

**Table 4.** Survey of RNG-based cryptosystems.

| Cipher Design | RNG | Type of Cryptosystem | Result | Parameters Improved | Ref. |
|---|---|---|---|---|---|
| Blostream | PRNG | Symmetric | Immune to Brute-Force, Statistical, Deferential, Distinguishing and Correlations Attacks | ➢ Speed,<br>➢ Memory Requirements | [59] |
| Hybrid Cryptosystem | PRNG + TRNG | Symmetric | Strong Key | ➢ Non-deterministic<br>➢ Unpredictable<br>➢ Reproducible<br>➢ Periodic | [60] |
| Text Encryption Stream | PRNG | Symmetric | Immune To All Known-Plaintext Cryptanalysis Attacks | ➢ Large Plaintext<br>➢ Key Sensitivity | [61] |
| Present | PRNG + TRNG | Symmetric | Better Performance with High Security | ➢ Slices<br>➢ LookUp Table<br>➢ Frequency<br>➢ Power | A [62] |

**Table 4.** *Cont.*

| Cipher Design | RNG | Type of Cryptosystem | Result | Parameters Improved | Ref. |
|---|---|---|---|---|---|
| Text Encryption Algorithms | PRNG | Symmetric | Strength against Linear, Differential and Statistical Attacks | ➢ Plaintext sensitivity<br>➢ Key sensitivity<br>➢ Robustness against known plaintext attack | [63] |
| Diffie-Hellman Key Exchange—Using QRNG | QRNG | Asymmetric | Non-Vulnerable Cryptographic System | ➢ Key Entropy<br>➢ Plaintext Entropy<br>➢ Ciphertext Entropy | B [64] |
| CCAES—Chaos-based | PRNG | Symmetric | More Secure and Effective Resistant to Differential Attacks | ➢ Histogram<br>➢ Correlation<br>➢ NPCR and UACI<br>➢ Information Entropy | [65] |
| Image Encryption Algorithm—Chaos-based | PRNG | Symmetric | Security Enhancement | ➢ Entropy<br>➢ Cross Correlation<br>➢ Mean Square Error<br>➢ PSNR | [66] |
| Authenticated Encryption with Associated Data (AEAD)—Chaos-based | PRNG | Symmetric | Highly Secure for Ciphertext and Authentication Tag Resistant to Differential, Linear, Algebraic, and Timing Attacks. | ➢ Privacy and Integrity<br>➢ Measured by Statistical Test suite NIST, DIEHARD and ENT. | C [67] |
| Hybrid RSA | PRNG | Asymmetric | Strong Encryption | ➢ Histogram<br>➢ Correlation<br>➢ NPCR and UACI<br>➢ Key Sensitivity<br>➢ Key Space<br>➢ Information Entropy | [68] |

NPCR—Number of Changing Pixel Rate, UACI—Unified Averaged Changed Intensity, PSNR—Peak Signal to Noise Ratio. Note: Publication project supported by: A—The Xiamen University Malaysia Research Fund (XMUMRF); B—Prometeo Project of the Ministry of Education Superior, Science, Technology and Innovation of the Republic of Ecuador; C—Fundamental Research Grant Scheme funded by the Ministry of Higher Education of Malaysia (MOHE).

In the next category, we have focused on QRNG and its research design, which is the basis of incorporating quantum randomness into a cryptosystem.

### *2.3. Category III: Research Objectives of Quantum RNGs for Cryptosystems*

QRNGs as an external device are best suited when the sender and receiver, e.g., Alice and Bob, are the same. Alice wants to store data like personal files (driving license, passport, other identity proofs) and highly secure work files like military data on the cloud. Consequently, the data should be encrypted before sending it to the network and stored in the cloud. Here, KSA comprises the QRNG bits to make the encrypted key stronger and more complex.

We have discussed classifications of RNGs in category II, and the survey shows the improvement in various cryptosystem ciphers even using pseudo-randomness, which is not truly random.

It is impossible to consider true randomness in PRNG-generated [5] sequences as they are implemented by software, based on mathematical algorithms and determined by an initial (seed) value. In contrast, the TRNG and QRNG, based on unpredictable physical means, generate the true randomness in the sequence of random numbers. Furthermore, there are two significant differences between TRNGs and PRNGs. TRNGs [69] are non-deterministic and use a physical mechanism, whereas PRNGs are deterministic and utilize mathematical algorithms. Although TRNGs are unpredictable, they are still vulnerable to attack because if a failure occurs in the TRNG hardware system, it is hard to detect it.

Dr. Mads Haahr [70] introduced the service of providing different TRNGs online. Some are freely available like number-based TRNGs, lists, strings, and map-based TRNGs; some are paid and used for random drawings. Web tools and widget- based TRNGs are

also freely available for the website to be integrated into the webpage and provide the TRNG-based random number. As atmospheric noise creates these random numbers, they are better than pseudo-random numbers.

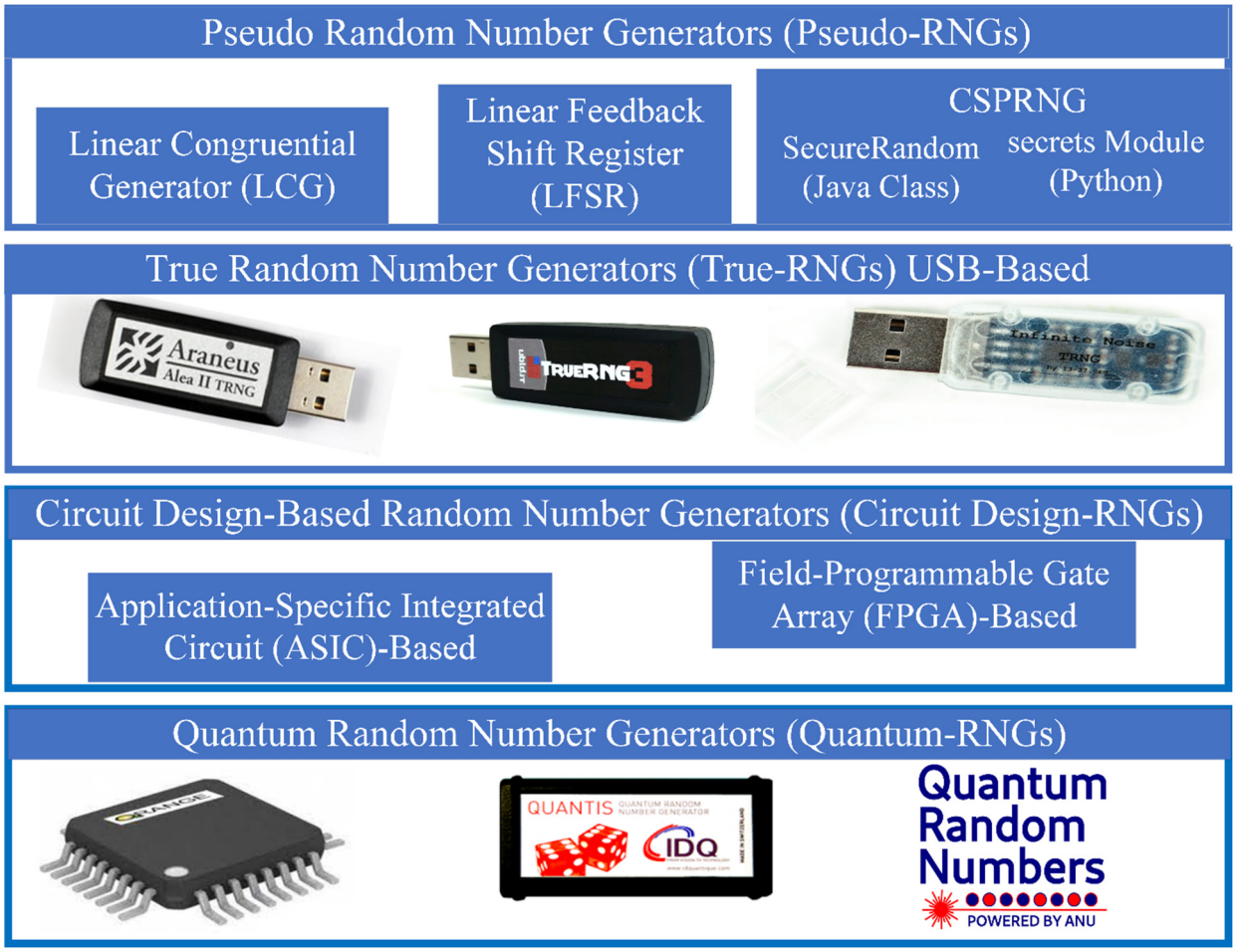



**Figure 5.** Categorization of random number generators: Implementation with applications.

In some cases of TRNGs, the entropy generated by an entropy source may be confused with thermal noise or shot noise [71]. However, a highly secure and unique random number is based on the quantum principle and generated by a QRNG [4]. A QRNG-based cryptosystem incorporates quantum randomness to generate a more robust key of the KSA or enhance encryption security.

Furthermore, Shor's algorithm [72] concluded that quantum computers could easily break public key cryptography. It proposed to solve the prime factorization of RSA and results with high probability. The quantum random numbers generated by QRNGs are also quantum-based and might secure the new cryptosystem without a prime number. Also, incorporating quantum randomness challenges this type of quantum algorithm for cryptoanalysis of the cipher.

In this category, we discuss the QRNG, which generates unbiased, high-speed, and unpredictable random numbers by various methods such as laser pulses, phase diffusion, photon-based methods, and other quantum mechanics techniques. Figure 6 shows the difference between RNGs in terms of their generation methods, properties, disadvantages, and advantages.

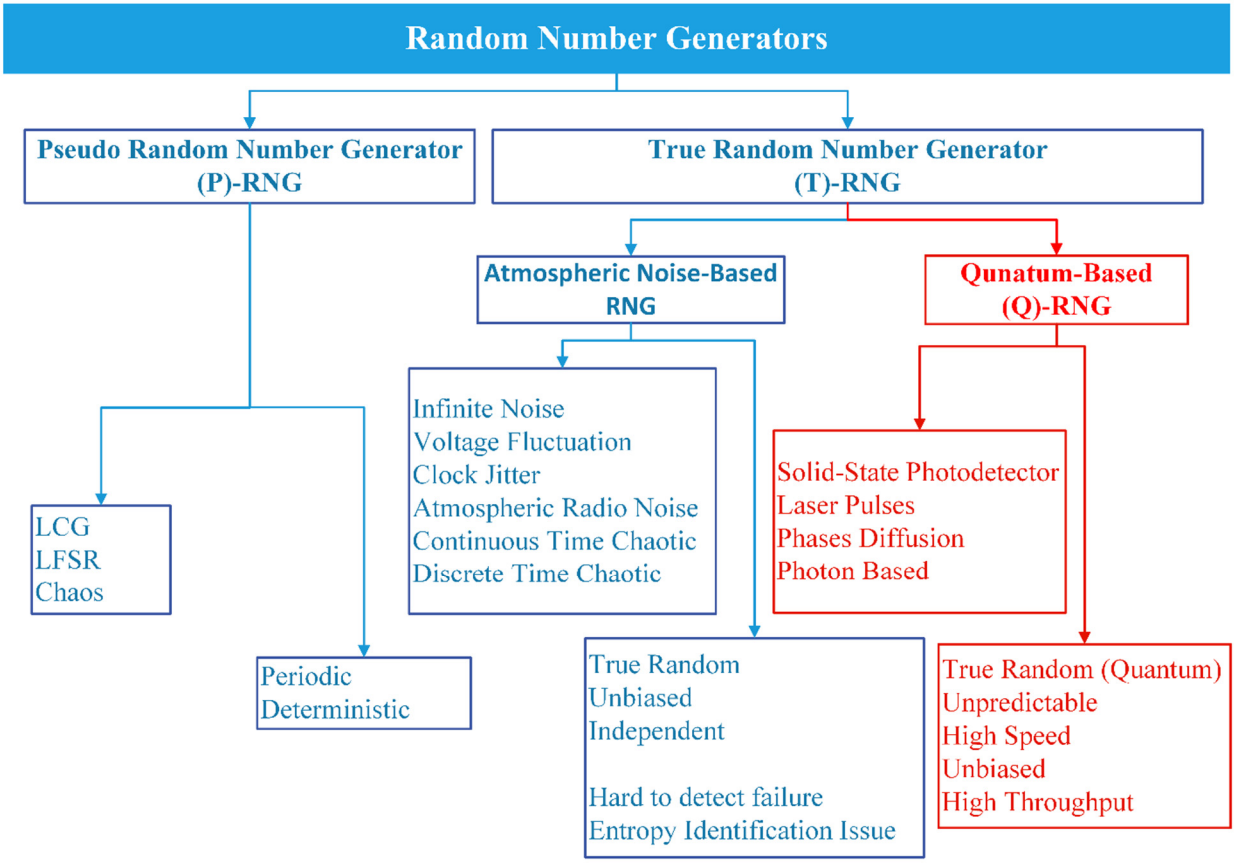



**Figure 6.** Categorization of Random Number Generators: generation methods, properties, disadvantages advantages.

We have reviewed 20 QRNGs and divided them into three important research objectives: High speed (HS), bias improvement (BI) and high efficiency (HE).

High speed (HS): The speed of the RNG device is indicated by the rate at which random bits are generated per second by the device. Speed is critical in determining the optimal trade-off for the RNG device. Different research methodologies attempt to identify the shortest time it takes for the device to generate the most random bits.

Bias improvement (BI): Random bits should be bias-free. Bias occurs when a significantly more zeros than ones, or vice versa, are generated. If the difference between zeros and ones is small enough, it should not introduce bias into the quantum random bits. Numerous research proposals have been designed to improve the bias in order to achieve balanced quantum random bits.

High efficiency (HE): The performance of a QRNG can be improved by researching different technologies on which the generator can be built. The efficiency of a device is defined in terms of extracting random bits or maximising the percentage of bits with specific physical photon-based conditions.

Table 5 shows the survey of different quantum random number generation techniques and their research objectives.

The next section addresses some open research challenges specific to QRNG-based cryptosystems.

**Table 5.** Literature review of QRNGs.

| QRNG Proposal Techniques | Ref. | HS | BI | HE | Other Properties |
|---|---|---|---|---|---|
| Weak laser pulses | [73] | √ | √ | | |
| Random quantum states in mathematica | [74] | √ | | | |
| Tapered amplifiers | [75] | | √ | | Self-Detecting |
| One single-photon detector | [76] | | √ | | |
| Photon arival time selectively | [77] | | √ | | |
| Three-types QRNG | [78] | √ | | | Self-testing |
| Ultra-fast QRNG—pulsed laser diode | [79] | √ | | | |
| Heterodyne-based | [80] | √ | | | Security |
| 16 × 16 pixel QRNG based on SPADs | [81] | √ | | √ | |
| Laser phase fluctuations | [82] | √ | | | |
| One and two entropy sources | [83] | | | √ | |
| Phase diffusion process-based | [84] | √ | | | |
| SPAD-based QRNG using FPGA | [85] | | | √ | |
| SPAD-based QRNG pixel-based | [86] | | | √ | |
| Multi-bit | [87] | | | √ | |
| Uncharacterized laser andsunlight | [88] | √ | | | |
| Coupled quantum dots | [89] | | √ | | |
| Phase diffusion in lasers | [90] | | √ | | Interference Quality, Input/Output Monitoring |
| Quantis—USB | [8] | √ | √ | | Autocalibration Status Monitoring |
| Qrange | [57] | √ | | | Security |

Note: HS, high speed; BI, bias improvement; HE, high efficiency.

## 3. Open Research Problems

In Section II, we reviewed the literature, and our analysis identified some open research problems to address and enhance the security of cryptosystems. RNGs play a significant role in securing systems that use random numbers. QRNs [9,32] are based on true randomness and can be used to strengthen the security of existing cryptography systems.

Future research will be needed to address the following challenges:

(a) The analysis of incorporating the quantum randomness in stream cipher operations compared to pseudo-based ciphers.

Boolean arithmetic-based cryptography mainly uses cryptographic-secure pseudo-random number generators (CSPRNG). A pseudo-Hadamard transform based on the modular operation of boolean arithmetic provides diffusion in cryptographic cipher. The diffusion depends on the change of an input bit. RNGs can generate the input bit to affect the cryptanalysis of the cipher.

Guillermo Sosa-Gómez et al. [91] showed the cryptanalysis of PRNGs for cryptographic ciphers. Also, a correlation analysis of the entropy of PRNGs and Hadamard values indicate a strong correlation. On the other hand, QRNG based on the photon can mitigate this cryptoanalysis.

For most CSPRNGs, the entropy is derived from the operating system, along with a PRNG generator [92]. The underlying PRNG often "reseeds", which means that when an entropy is supplied by the operating system (e.g., from user input, system interruptions, disk I/O, or hardware random generators), it changes its internal state. However, quantum randomness is derived from quantum particles without the intervention of reseeding.

(b) Design a KSA by using different entropy sources for quantum random bits in order to randomize the keyspace for differential attacks.

The differential attacks try to find the key with the chosen plaintext and corresponding ciphertext. The subkeys of KSA are vulnerable if the keyspace between them is not random or large. Incorporating QRNGs can generate random subkeys, increasing or enhancing the randomness of their keyspace. The stronger the keyspace, the more challenging it is to detect even a single subkey. As a result, a QRNG-based KSA may produce a strong cipher that is not vulnerable to differential attack.

(c) An in-depth study is needed to analyse the effects of key-related attacks on QRN-based ciphers.

In key-related attacks, the attacker knows (or chooses) a relation between several keys (up to 256 in some recent attacks) and is given access to encryption functions with such related keys. Keys and plaintexts can be selected with specific differences when using differential related-key attacks.

The QRNG will support the true randomness in the cryptography key against key-related attacks. The effect of the true randomness on the cryptanalysis of the cipher is to be analysed to secure the key and to make it truly random.

(d) Designing a new high-speed encryption cipher based on different research designs (high-speed and bias-free) by using QRNs to enhance the security of the cryptosystem.

The encryption speed of the block cipher is one of the major primary concerns when designing a new cipher. However, the security of the cipher must not be compromised on the basis of time, but the time complexity of an algorithm can make it more competitive against alternatives. The QRNG will add true randomness into the key by generating the quantum random bits from an external photon device. These random bits are highly important to secure the cipher, but might come with a time cost. New cryptographic primitives are sought after that reduce this time and introduce quantum randomness, leading to a highly secure cryptosystem.

(e) Research analysis on storing and exchanging the QRNG-based keys generated for asymmetric cryptosystem over the cloud.

The QRNGs are based on devices that can generate photons. These devices can be with sender or receiver, and are hence important when both have the same key, i.e., symmetric cryptosystem. They help generate two keys, a pair of public and private keys, but transfer them with the knowledge of quantum random bits and store them on the cloud as staging post needs further research.

## 4. Conclusions

We have surveyed cryptosystems, different cryptanalysis techniques, and RNGs. The analysis shows that cryptoanalysis is possible for various ciphers and, therefore, requires modification in the cryptosystem to secure them. RNGs provide random numbers in the cryptosystem and enhance the cipher's security by making the encryption robust or resistant to various cryptographic attacks. In addition, QRNGs provide true randomness, compared to PRNGs and atmospheric noise-based TRNGs, because they are based on quantum principles by different theories of quantum physics. Finally, the identification of open research problems guides researchers in the cryptography security field to improve the security based on the quantum randomness in a cryptosystem.

**Funding:** This research review work received funding by the University of Hertfordshire, United Kingdom.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Stallings, W. *Cryptography and Network Security: Principles and Practices*, 4th ed.; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 2005.
2. Smart, N.P.; Rijmen, V.; Warinschi, B.; Watson, G. Algorithms, Key Sizes and Parameters Report. ENISA, Nov. 2014. Available online: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014 (accessed on 9 September 2021).

3. Sahmoud, S.; Elmasry, W.; Shadi, A. Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher. *Int. Arab. J. e-Technol.* **2013**, *3*, 17–26.
4. Oracle, SecureRandom. 2020. Available online: https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#SecureRandom (accessed on 23 June 2020).
5. Sinha, S.; Islam, S.H.; Obaidat, M.S. A comparative study and analysis of some pseudorandom number generator algorithms. *Secur. Priv.* **2018**, *1*, e46. [CrossRef]
6. Gong, L.; Zhang, J.; Liu, H.; Sang, L.; Wang, Y. True Random Number Generators Using Electrical Noise. *IEEE Access* **2019**, *7*, 125796–125805. [CrossRef]
7. Melia, J.; Huttner, B.; Moulds, R.; Walenta, N.; Fuller, A.; Quantum-Safe Security Working Group. *Quantum Random Number Generators*; Cloud Security Alliance: Bellingham, WA, USA, 2016.
8. ID Quantique, What Is the Q in QRNG ? 2020. Available online: https://www.idquantique.com/random-number-generation/overview/ (accessed on 7 July 2020).
9. ID Quantique, Understanding Quantum Cryptography. ID Quantique SA. 2020. Available online: https://www.idquantique.com/quantum-safe-security/quantum-key-distribution/ (accessed on 7 July 2020).
10. ID Quantique, Gaming-and-Lotteries. Available online: https://www.idquantique.com/random-number-generation/applications/gaming-and-lotteries/ (accessed on 7 July 2020).
11. Open Quantum Safe. Available online: https://openquantumsafe.org/ (accessed on 9 May 2022).
12. Biryukov, A.; de Cannière, C. Data encryption standard (DES). In *Encyclopedia of Cryptography and Security*; Springer: Boston, MA, USA, 1999. [CrossRef]
13. Massey, J.L. SAFER K-64: A byte-oriented block-ciphering algorithm. In Proceedings of the International Workshop on Fast Software Encryption, Cambridge, UK, 9–11 December 1993; Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany, 1994; Volume 809, pp. 1–17. [CrossRef]
14. Daemen, J.; Govaerts, R.; Vandewalle, J. A new approach to block cipher design. In Proceedings of the International Workshop on Fast Software Encryption, Cambridge, UK, 9–11 December 1993; Springer: Berlin/Heidelberg, Germany, 1994; pp. 18–32.
15. Anderson, R.; Biham, E.; Knudsen, L. Serpent: A Proposal for the Advanced Encryption Standard. NIST AES Proposal. 1998, pp. 1–23. Available online: https://bitbucket.org/nicholascapo/network-security-project/src/fcbc6e93e555/Literature/serpent.pdf (accessed on 9 October 2021).
16. Daemen, J.; Rijmen, V. *The Design of Rijndael*; Springer: Berlin/Heidelberg, Germany, 2002.
17. Jenkins, R.J., Jr. ISAAC and RC4. 1993. Available online: http://burtleburtle.net/bob/rand/isaac.html (accessed on 12 June 2022).
18. Quirke, J. Security in the GSM System. AusMobile. 1 May 2004. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.1509&rep=rep1&type=pdf (accessed on 9 October 2021).
19. Security Algorithms Group of Experts, ETR 278—ETSI Technical Report. 1996. Available online: https://cryptome.org/espy/ETR278e01p.pdf (accessed on 4 May 2021).
20. Lai, X.; Massey, J.L. A Proposal for a New Block Encryption Standard. In Proceedings of the Advances in Cryptology—EUROCRYPT '90, Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, 21–24 May 1990; Lecture Notes in Computer Science. Damgård, I.B., Ed.; Springer: Berlin/Heidelberg, Germany, 1991; Volume 473, pp. 389–404. [CrossRef]
21. Schneier, B. Description of a new variable-length key, 64-bit block cipher (Blowfish). In Proceedings of the International Workshop on Fast Software Encryption, Cambridge, UK, 9–11 December 1993; Springer: Berlin/Heidelberg, Germany, 1994; pp. 191–204.
22. Boesgaard, M.; Vesterager, M.; Pedersen, T.; Christiansen, J.; Scavenius, O. Rabbit: A new high-performance stream cipher. In Proceedings of the 10th International Workshop, Fast Software Encryption 2003, Lund, Sweden, 24–26 February 2003; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer: Berlin/Heidelberg, Germany, 2003; Volume 2887, pp. 307–329. [CrossRef]
23. Hawkes, P.; Rose, G.G. Primitive Specification for SOBER-128. IACR Cryptology ePrint Archive. 2003, p. 81. Available online: http://dblp.uni-trier.de/db/journals/iacr/iacr2003.html#HawkesR03a (accessed on 2 January 2020).
24. Berbain, C.; Gilbert, H.; Patarin, J. QUAD: A multivariate stream cipher with provable security. *J. Symb. Comput.* **2009**, *44*, 1703–1723. [CrossRef]
25. Christophe, D.C.; Preneel, B. Trivium Specifications. 2006, Volume 507932. Available online: https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf (accessed on 2 January 2020).
26. Bernstein, D.J. The salsa20 family of stream ciphers. In *New Stream Cipher Designs*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Berlin/Heidelberg, Germany, 2008; Volume 4986, pp. 84–97. [CrossRef]
27. Wheeler, D.J.; Needham, R.M. TEA, a tiny encryption algorithm. In Proceedings of the Fast Software Encryption, Second International Workshop, Leuven, Belgium, 14–16 December 1994; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer: Berlin/Heidelberg, Germany, 1995; pp. 363–366. [CrossRef]
28. Adams, C. The CAST-128 Encryption Algorithm. RFC Editor. May 1997. Available online: https://www.rfc-editor.org/info/rfc2144 (accessed on 12 June 2021).

29. Ekdahl, P.; Johansson, T. A new version of the stream cipher SNOW. In Proceedings of the Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, NL, Canada, 15–16 August 2002; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer: Berlin/Heidelberg, Germany, 2003; Volume 2595, pp. 47–61. [CrossRef]
30. Knudsen, L.R.; Rijmen, V.; Rivest, R.L.; Robshaw, M.J.B. On the Design and Security of $RC_2$. In Proceedings of the Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, 23–25 March 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 206–221.
31. Crypto-1. Available online: https://en.wikipedia.org/wiki/Crypto-1 (accessed on 12 June 2021).
32. Rivest, R.L.; Robshaw, M.J.B.; Sidney, R.; Yin, Y.L. The RC6 Block Cipher. In Proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, USA, 20–22 August 1998; p. 16.
33. Burwick, C.; Coppersmith, D.; D'Avignon, E.; Gennaro, R.; Halevi, S.; Jutla, C.; Matyas, S.M., Jr.; O'Connor, L.; Peyravian, M.; Safford, D.; et al. MARS—A Candidate Cipher for AES. NIST AES Proposal. 1998, pp. 8–23. Available online: http://cryptosoft.de/docs/Mars.pdf (accessed on 12 June 2021).
34. Schneier, B.; Kelsey, J.; Whiting, D.; Wagner, D.; Hall, C.; Ferguson, N. Twofish: A 128-Bit Block Cipher. NIST AES Proposal. 1998, Volume 15, pp. 1–27. Available online: https://www.schneier.com/wp-content/uploads/2016/02/paper-twofish-paper.pdf (accessed on 7 July 2021).
35. Wu, H. Stream Cipher HC-256. Available online: https://eprint.iacr.org/2004/092.pdf (accessed on 9 May 2022).
36. Hell, M.; Johansson, T.; Meier, W. Grain: A stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.* **2007**, *2*, 86–93. [CrossRef]
37. Babbage, S.; Dodd, M. The Stream Cipher MICKEY 2.0. ECRYPT Stream Cipher Project, Report. 2006, pp. 1–12. Available online: https://www.cosic.esat.kuleuven.be/ecrypt/stream/p2ciphers/mickey/mickey_p2.pdf (accessed on 12 June 2021).
38. Japan's First 128-Bit Block Cipher 'Camellia' Approved as a New Standard Encryption Algorithm in the Internet. NTT News Release. Available online: https://www.ntt.co.jp/news/news05e/0507/050720.html (accessed on 12 June 2021).
39. Fontaine, C. SEAL. In *Encyclopedia of Cryptography and Security*; Springer: Boston, MA, USA, 2011; p. 543.
40. Ferguson, N.; Lucks, S.; Schneier, B.; Whiting, D.; Bellare, M.; Kohno, T.; Callas, J.; Walker, J. Threefish. Available online: https://www.schneier.com/academic/skein/threefish/ (accessed on 2 August 2021).
41. Cannière, C. GOST. In *Encyclopedia of Cryptography and Security*; Springer: Boston, MA, USA, 2011. [CrossRef]
42. Whiting, D.; Schneier, B.; Lucks, S.; Muller, M. Phelix. 2004. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.8097&rep=rep1&type=pdf (accessed on 7 April 2021).
43. NIST, Post-Quantum Cryptography. Available online: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions (accessed on 12 June 2022).
44. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
45. ElGamal, T. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Available online: https://link.springer.com/content/pdf/10.1007/3-540-39568-7_2.pdf (accessed on 9 May 2022).
46. *FIPS 186*; Digital Signature Standard (DSS). National Institute of Standards and Technology: Gaithersburg, MD, USA, 1994.
47. *SEC 1 Ver. 2.0*; Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography. Certicom Research: Mississauga, ON, Canada, 2009.
48. Josefsson, S.; Liusvaara, I. Edwards-Curve Digital Signature Algorithm (EdDSA). *J. Chem. Inf. Modeling* **2017**, *53*, 1689–1699. Available online: https://www.rfc-editor.org/info/rfc8032 (accessed on 12 December 2021).
49. Python Software Foundation, Secrets. 2022. Available online: https://docs.python.org/3/library/secrets.html (accessed on 9 May 2022).
50. Araneus Information Systems Oy, Araneus Alea II. 2022. Available online: https://www.araneus.fi/products/alea2/en/ (accessed on 9 May 2022).
51. Ubld.it, TrueRNG v3. 2022. Available online: https://ubld.it/truerng_v3 (accessed on 9 August 2021).
52. Crowd Supply, Infinite Noise TRNG. 2022. Available online: https://www.crowdsupply.com/leetronics/infinite-noise-trng (accessed on 9 May 2022).
53. Nannipieri, P.; Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Belli, J.; Fanucci, L.; Saponara, S. True random number generator based on fibonacci-galois ring oscillators for fpga. *Appl. Sci.* **2021**, *11*, 3330. [CrossRef]
54. Crocetti, L.; di Matteo, S.; Nannipieri, P.; Fanucci, L.; Saponara, S. Design and Test of an Integrated Random Number Generator with All-Digital Entropy Source. *Entropy* **2022**, *24*, 139. [CrossRef]
55. European Processor Initiative (EPI). Available online: https://www.european-processor-initiative.eu/ (accessed on 12 June 2022).
56. IDQ, Quantis-Random-Number-Generator. 2020. Available online: https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator (accessed on 7 July 2020).
57. QRANGE, Qrng. 2020. Available online: https://qrange.eu/project/qrng (accessed on 27 July 2020).
58. ANU QRNG. 2022. Available online: https://qrng.anu.edu.au/ (accessed on 5 September 2021).
59. Kashmar, A.H.; Ismail, E.S. Blostream: A high speed stream cipher. *J. Eng. Sci. Technol.* **2017**, *12*, 1111–1128.
60. Patnala, T.R. A Modernistic way for KEY Generation for Highly. In Proceedings of the 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 892–897.

61. Amro, A.; El-Alfy, E.S.M. Known-plaintext attack and improvement of PRNG-based text encryption. In Proceedings of the 7th International Conference on Information and Communication Systems, ICICS 2016, Irbid, Jordan, 5–7 April 2016; pp. 233–238. [CrossRef]
62. Kowsalya, T.; Babu, R.G.; Parameshachari, B.D.; Nayyar, A.; Mehmood, R.M. Low area PRESENT cryptography in FPGA using TRNG-PRNG key generation. *Comput. Mater. Contin.* **2021**, *68*, 1447–1465. [CrossRef]
63. Mishra, M.; Mankar, V.H. Text Encryption Algorithms based on Pseudo Random Number Generator. *Int. J. Comput. Appl.* **2015**, *111*, 1–6. [CrossRef]
64. Mogos, G. Use quantum random number generator in Diffie-Hellman key exchange protocol. In Proceedings of the 2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 19–21 May 2016; pp. 1–6. [CrossRef]
65. Arab, A.; Rostami, M.J.; Ghavami, B. An image encryption method based on chaos system and AES algorithm. *J. Supercomput.* **2019**, *75*, 6663–6682. [CrossRef]
66. Banthia, A.K.; Tiwari, N. Image Encryption using Pseudo Random Number Generators. *Int. J. Comput. Appl.* **2013**, *67*, 1–8. [CrossRef]
67. Teh, J.S.; Samsudin, A. A chaos-based authenticated cipher with associated data. *Secur. Commun. Netw.* **2017**, *2017*, 9040518. [CrossRef]
68. Çavuşoğlu, Ü.; Akgül, A.; Zengin, A.; Pehlivan, I. The design and implementation of hybrid RSA algorithm using a novel chaos based RNG. *Chaos Solitons Fractals* **2017**, *104*, 655–667. [CrossRef]
69. Hughes, R.; Nordholt, J. Strengthening the Security Foundation of Cryptography with Whitewood's Quantum-Powered Entropy Engine. 2016. Available online: http://www.whitewoodencryption.com (accessed on 3 August 2021).
70. Mads, H. Random.org. 1998. Available online: https://www.random.org/ (accessed on 12 June 2022).
71. Lee, J.; Seo, Y.; Heo, J. Analysis of random number generated by quantum noise source and software entropy source. In Proceedings of the 9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, Maison Glad Jeju, Jeju Island, Korea, 17–19 October 2018; pp. 729–732. [CrossRef]
72. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
73. Wei, W.; Guo, H. Quantum random number generator based on the photon number decision of weak laser pulses. In Proceedings of the Optics InfoBase Conference Papers, Shanghai, China, 30 August–3 September 2009. Available online: http://arxiv.org/abs/0811.0082 (accessed on 9 May 2022).
74. Miszczak, J.A. Employing online quantum random number generators for generating truly random quantum states in Mathematica. *Comput. Phys. Commun.* **2013**, *184*, 257–258. [CrossRef]
75. Pooser, R.C.; Evans, P.G.; Humble, T.S.; Grice, W.P.; Williams, B.P. Self correcting quantum random number generators using tapered amplifiers. In Proceedings of the Optics InfoBase Conference Papers, Kyoto, Japan, 30 June–4 July 2013. [CrossRef]
76. Soares, E.d.L.; Mendonca, F.A.; Ramos, R.V. Quantum random number generator using only one single-photon detector. *IEEE Photonics Technol. Lett.* **2014**, *26*, 851–853. [CrossRef]
77. Wang, J.M.; Xie, T.Y.; Zhang, H.F.; Yang, D.X.; Xie, C.; Wang, J. A bias-free quantum random number generation using photon arrival time selectively. *IEEE Photonics J.* **2015**, *7*. [CrossRef]
78. Ma, X.; Yuan, X.; Cao, Z.; Qi, B.; Zhang, Z. Quantum random number generation. *npj Quantum Inf.* **2016**, *2*, 16021. [CrossRef]
79. Siswanto, M.; Rudiyanto, B. Designing of quantum random number generator (QRNG) for security application. In Proceedings of the 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech, Bandung, Indonesia, 25–26 October 2017; pp. 273–277. [CrossRef]
80. Avesani, M.; Marangon, D.G.; Vallone, G.; Villoresi, P. Quantum Random Number Generator at 17 Gbps. *Nat. Commun.* **2018**, *9*, 5365. [CrossRef]
81. Xu, H.; Perenzoni, D.; Tomasi, A.; Massari, N. A 16 × 16 Pixel Post-Processing Free Quantum Random Number Generator Based on SPADs. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 627–631. [CrossRef]
82. Hasan, R.S.; Tawfeeq, S.K.; Mohammed, N.Q.; Khaleel, A.I. A true random number generator based on the photon arrival time registered in a coincidence window between two single-photon counting modules. *Chin. J. Phys.* **2018**, *56*, 385–391. [CrossRef]
83. Shaw, G.; Sivaram, S.R.; Prabhakar, A. Quantum Random Number Generator with One and Two Entropy Sources. In Proceedings of the 2019 National Conference on Communications (NCC), Bangalore, India, 20–23 February 2019; pp. 1–4. [CrossRef]
84. Septriani, B.; de Vries, O.; Gräfe, M. Quantum random number generation (QRNG) by phase diffusion process in a gain-switched semiconductor laser—New insights. In Proceedings of the Conference on Lasers and Electro-Optics, San Jose, CA, USA, 5–10 May 2019. [CrossRef]
85. Tontini, A.; Gasparini, L.; Massari, N.; Passerone, R. SPAD-Based Quantum Random Number Generator with an Nth-Order Rank Algorithm on FPGA. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *66*, 2067–2071. [CrossRef]
86. Nicola, M.; Gasparini, L.; Meneghetti, A.; Tomasi, A. A SPAD-based random number generator pixel based on the arrival time of photons. In Proceedings of the 2017 1st New Generation of CAS, NGCAS, Genova, Italy, 6–9 September 2017; pp. 213–216. [CrossRef]
87. Sarkar, A.; Chandrashekar, C.M. Multi-bit quantum random number generation from a single qubit quantum walk. *Sci. Rep.* **2019**, *9*, 12323. [CrossRef] [PubMed]

88. Li, Y.-H.; Han, X.; Cao, Y.; Yuan, X.; Li, Z.-P.; Guan, J.-Y.; Yin, J.; Zhang, Q.; Ma, X.; Peng, C.-Z.; et al. Quantum random number generation with uncharacterized laser and sunlight. *npj Quantum Inf.* **2019**, *5*, 97. [CrossRef]
89. McCabe, H.; Koziol, S.M.; Snider, G.L.; Blair, E.P. Tunable, Hardware-Based Quantum Random Number Generation Using Coupled Quantum Dots. *IEEE Trans. Nanotechnol.* **2020**, *19*, 292–296. [CrossRef]
90. Imran, M.; Sorianello, V.; Fresi, F.; Potì, L.; Romagnoli, M. Quantum random number generator based on phase diffusion in lasers using an on-chip tunable soi unbalanced Mach-Zehnder interferometer (uMZI). In Proceedings of the Optics InfoBase Conference Papers, Optical Fiber Communication Conference, San Diego, CA, USA, 8–12 March 2020. [CrossRef]
91. Sosa-Gómez, G.; Rojas, O.; Páez-Osuna, O. Using hadamard transform for cryptanalysis of pseudo-random generators in stream ciphers. *EAI Endorsed Trans. Energy Web* **2020**, *7*, e1. [CrossRef]
92. Nakov, S. Secure-Random-Generators. Available online: https://cryptobook.nakov.com/secure-random-generators (accessed on 3 September 2021).

# Appendix II : Published paper II - CryptoQNRG: a new framework for evaluation of cryptographic strength in quantum and pseudorandom number generation for key-scheduling algorithms.

The paper is available online at https://rdcu.be/c7b6O

**Title:** CryptoQNRG: A new Framework for Evaluation of Cryptographic Strength in Quantum and Pseudorandom Number Generation for Key-Scheduling Algorithms

**Author(s): Saini, A - Tsokanos, A - Kirner, R – University of Hertfordshire**

**Address:**
Department of computer science
School of Physics, Engineering and Computer Science
University of Hertfordshire, Hatfield, United Kingdom

**Abstract**

In a cryptosystem, a cipher's security is directly dependent on a key-schedule or *Key-Scheduling Algorithm* (KSA) or that is used for both encryption and decryption. The random-number-based KSA adds another layer of security and prevents hackers from performing cryptanalysis. Several previous studies have investigated the strength of a cipher's encryption process. The strength evaluation of the key scheduling process has received less attention, that can lead to weaknesses in the overall encryption process. This paper proposes a new framework consisting of cryptographic strength evaluation criteria for Random Number Generators (RNG) based KSAs. Our framework (CryptoQNRG) evaluates different key-schedules based on pseudorandom and quantum random number generators with a set of tests. There are test suites that compare the strength of KSAs for different block ciphers. To the best of our knowledge this is the first time that a framework is built to compare the strength of KSAs incorporating RNGs and various block ciphers. CryptoQNRG comprises of four tests: Frequency, Bit_Correlation, Bit_Interfold, and Bit_Entropy. The tests are used to explore cryptographic properties such as unpredictability, balance of bits, correlation, confusion, and diffusion in the subkeys generated by the RNG based KSA. We have evaluated the most common KSAs with different block ciphers and a significant outcome of the proposed framework is the distinction between strong and weak RNG-based KSAs.

**Keywords:** Pseudo Random Number Generator, Quantum Random Number Generator, Block Cipher, Key Schedule Algorithms

## 1. Introduction

In cryptography, encryption methods[1] and ciphers are used to ensure data security and prevent unauthorized access. An encryption algorithm combines plaintext with key information to generate a ciphertext, making the plaintext difficult for hackers to decipher. This process of encryption with a secret key can lead to a secure cipher by combining nonlinearity, propagation criteria, correlation, algebraic immunity and randomness [2][3].

As part of an encryption process, the strength of the *Key-Schedule Algorithm* (KSA) directly impacts the security of an encryption algorithm. The KSA expands the secret key and generates subkeys according to the number of rounds of encryption required for a particular cryptographic algorithm. This generation and expansion of the secret key into multiple subkeys increases the robustness and complexity of the KSA. One of the potential flaws in this approach

is the vulnerability in the key schedule expansion, which is characterised by the algorithm's resistance to cryptoanalysis attacks [4][5][6].

Researchers have demonstrated that logical gates such as AND, NAND, XOR, and OR, as well as Boolean functions with symmetrical properties, can be used to achieve security in the key expansion [7]. In addition to that, quantum data generation by generating quantum random number bits can increase security.

The best RNGs generate more random and unpredictable data to make keys resilient against a cryptanalysis attack. Randomness is generated using an entropy source, which can be either pseudo or quantum-based [8]. Pseudorandom number sequences are associated with an initial input seed, whereas the Quantum Random Number Generator (QRNG) [9] ensures high entropy from a quantum origin such as light or thermal noise for example.

There are various studies related to QRNGs focusing on different features. Lunghi et al. [10] proposed a protocol for self-testing quantum random number generation, in which the user can monitor the entropy in real time. Hesong Xu et al.[11] proposed a QRNG using single-photon avalanche diodes (SPADs) that produces a quantum random number without any post-processing. Biasing is one of the critical features that researchers consider when generating a quantum number. R.C.Pooser et al. [12] used a tapered amplifier that consists of optical semiconductor devices and an array of random number registration techniques to create quantum-based random numbers. A photon arrival time selectively based high-quality bias-free QRNG was introduced by Jian-min Wang et al.[13].

Another aspect of quantum processes is the speed at which random numbers can be generated. Yu-Huai Li et al. [14] proposed quantum random number generation with an uncharacterized laser and sunlight that generates random numbers at 1 Mbps. Abellan et al. [15] proposed an ultra-fast quantum random number generation accelerated phase diffusion in a pulsed laser diode observing 43 Gbps.

Quantis [16], a QRNG developed by IDQ, generates random numbers using light consists of elementary "particles" called photons [17][18]. The device allows live verification of its operation and provides a high level of entropy without requiring a post-processing function to increase its entropy rate. QRNG [19] is considered superior to traditional random number generators, as their source of randomness is invulnerable to environmental perturbations such as temperature, voltage, or current and considered highly secure random number generators [20] [21].

An improvement in terms of RNG, from pseudo randomness to quantum randomness, will benefit a KSA. The focus of this article is to create a framework to evaluate the strength of cryptography KSAs generated by RNGs. In particular the concrete contributions of this work are:

- Formulation of a test suite to evaluate the strength of cryptography KSAs generated by RNGs. This test suite consists of four parts: a bit-frequency test, a bit-correlation test, a bit-interfold test, and a bit-entropy test.
- Applying the proposed criteria in the framework’s test suite to a number of different block ciphers’ key-scheduling algorithms. P-value is a measure of the statistical significance of a test result calculated by statistical computation. The test is based on statistics and provides a P-value and the probability of achieving a pass or fail result. A pass does not

indicate that the cipher is secure against all attacks, however, a failure suggests that the algorithm is highly susceptible to attacks.

In contrast to existing randomness tests such as CryptRndTest, NIST Random Number Generator test, Dieharder battery test, our proposed test suite evaluates the strength of the KSA's subkeys [35], [36], [37], [38]. The tests of our framework are designed to evaluate the main properties of a Key-schedule such as unpredictability, balance, confusion, diffusion and correlation. Moreover, our proposed work compares two different RNG-based key schedules of the same block cipher simultaneously to distinguish their strength.

The organisation of this paper is as follows: section 2 provides information about related work; section 3 introduces the structure and evaluation criteria of our framework; section 4 presents the different RNG-based key-scheduling algorithms we use in our tests. The test results and analysis are presented in section 5, while in section 6 the conclusion can be found.

## 2. Related Work

The key generation is a process that creates a key and expands it based on logical operations to encrypt plain text. The strength of the KSA [7] can be evaluated on the different properties of the key. In 2019, Hakim and Nusrom [22] proposed a new algorithm for scheduling subkeys in the L-block cipher, as the Niaz correlation test concluded that the LBlock's KSA generates keys with a high correlation. Kareem and Rahma [23] proposed a novel method for modifying the Twofish algorithm by implementing multi-level keys in the KSA to control the dynamic block bit sizes and multi-state tables. Using these keys allows for a greater complexity in the algorithm while incurring a relatively small amount of additional computational time. Sulaiman et al. [24] proposed an enhancement of Rijndael's KSA [35]. The analysis conducted by Sulaiman addresses the algorithm's shortcomings and optimises the KSA in terms of frequency and Strict Avalanche Criteria. Huang et. al. [25] modified AES's KSA by transposing its subkey matrix. According to the authors, the new KSA is immune to SQUARE, meet-in-the-middle, and related-key differential type attacks. Shahzadi et al. [26] proposed that 2D Chaotic maps enhance the strength of the generated keys in the KSA of RC5 algorithm, making it difficult for hackers to decrypt the data. Their KSA work targets resource-constrained environments and analyses the security mechanism for specific applications of critical clinical images.

The security of the key generation process starts with the generation of bits based on a random number. Sahmoud et al. [27] proposed generating distinct subkeys from the AES real key using a Pseudo-Random-Number-Generator (PRNG) and encrypting the block with each subkey of KSA. Their research focused on the two techniques: first, preventing predictions of obtaining the sub-key from an available one, and second, presenting an initialisation method to speed up sub-key generation. Maram et al. [28] also used a PRNG and proposed a dynamic key-dependent S-box in KSA that achieved a better Avalanche effect with a cryptography algorithm.

Rahul Saha et al. [29] took a different approach, proposing a Symmetric Random Function Generator (SRFG) as a cryptographic function generating randomness in the KSA of AES. The results indicate that their proposed work has a threefold improvement in terms of confusion property and avalanche effect over the original AES. In another study [30], the FORTIS algorithm developed by Vuppala et al. [22] for generating sub-keys of KSA was implemented on an FPGA and the authors analysed the algorithm's resistance to a side power channel attack. Gaetan Leurent et. al. [31] presented a new representation for the AES's KSA

that efficiently combined the information from the first sub-key and the last sub-key to reconstruct the master key. Lauren et. al. [32] discussed the security properties of AES's KSA and its vulnerability to published attacks. Also, based on the information principle introduced by Claude Shannon, they proposed a faster and more secure KSA for generating subkeys in an AES cipher.

Afzal et al. [33] have recently published work that is closely related to our research. They proposed a Key-Schedule Evaluation Criterion (KSEC) to evaluate the cryptographic strength of subkeys of different KSA and establish a distinction between weak and strong keys using four statistical [34] tests.

They proposed a test suite consisting of a frequency test to analyse the balance of 0 and 1 bits, Bit Independence tests for confusion and diffusion property, Bitwise Uncorrelation tests for correlation among subkeys, and High/low-density key tests for testing randomness of the subkeys. Their test suite compares the strength of KSA of different block ciphers. To the best of our knowledge this is the first time that a framework is built to compare the strength of KSA based on RNGs and various block ciphers.

## 3. The CryptoQNRG framework

A KSA generates subkeys based on the input of the user-defined key. A RNG-based KSA adds another layer of security to the subkeys. CryptoQNRG has a series of test criteria in order to evaluate the strength of an RNG-based KSA.

Figure 1 illustrates an example of a basic scenario that we considered in our research. In this scenario personal data are stored on an internet server (cloud or a data center) so that they can be easily accessible from anywhere.

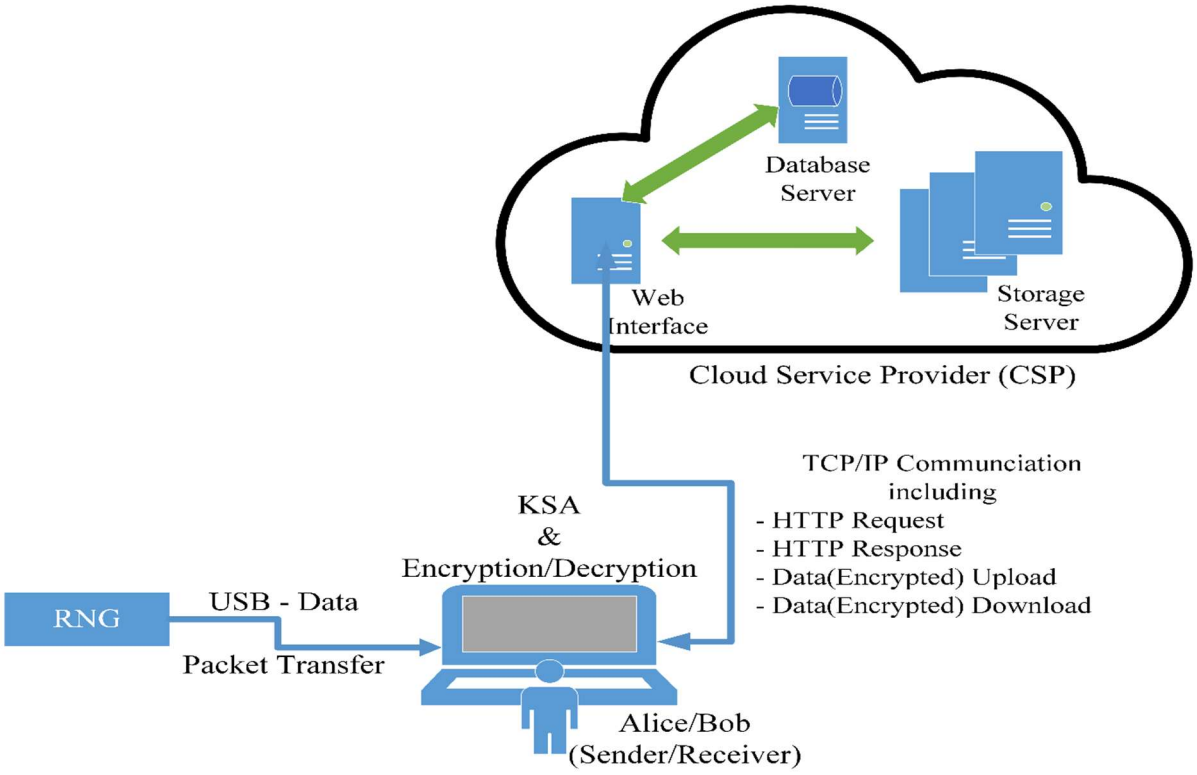



*Figure 1 Basic scenario of a secure communication*

The requirement is that the files must be encrypted before being sent over the network to make them secure, and when the user downloads them, they can be decrypted locally. Random Number Generator (RNG) plays a significant role as the KSA key comprises the bits of random numbers generated by an RNG to encrypt the contents.

Figure 2 shows the block diagram of CryptoQNRG for evaluating the strength of RNG-based KSA. The two keys, $K_1$ and $K_2$, are subkeys of individual KSAs generated using two different RNGs. The $K_1$ and $K_2$, are then passed to the proposed Key Schedule Evaluation Criterion – $KSEC_{RNG}(K_1, K_2)$ and tested for the cryptographic properties. The criterion includes four statistical tests. The first test, Frequency test $FT(K_1, K_2)$, calculates the frequency of 0's and 1's and checks how balanced their distribution is. The second test, Bit_Correlation $BCT(K_1, K_2)$ test, measures correlation, whereas, Bit_Interfold $BIT(K_1, K_2)$, the third test, checks for the confusion and diffusion properties. The last test is called Bit_Entropy $BET(K_1, K_2)$ test and examines the unpredictability of the bit stream generation. Based on the evaluation of each test, the strength of KSA is considered. The strength of KSA of $K_1$ and $K_2$ is strong if $K_1$ and $K_2$ pass all the tests of the $KSEC_{RNG}(K_1, K_2)$; otherwise, the KSA is weak. The proposed criterion, CryptoQNRG, which also compares $K_1$ and $K_2$ with the Bit_Entropy test.

The strength of KSA ($ST_{KSA}$) and difference of KSA ($DF_{KSA}$) are defined as follows:

$$ST_{KSA} = \begin{cases} \text{Strong}, & if\ (K_1, K_2)\ pass\ each\ test\ in\ EC_{RNG}(K_1, K_2) \\ \text{Weak}, & otherwise \end{cases}$$

$$DF_{KSA} = \begin{cases} K_1\ is\ better\ then\ K_2, & if\ \ K_1 > K_2\ in\ BET\ (K_1, K_2)\ tests \\ K_2, & otherwise \end{cases}$$

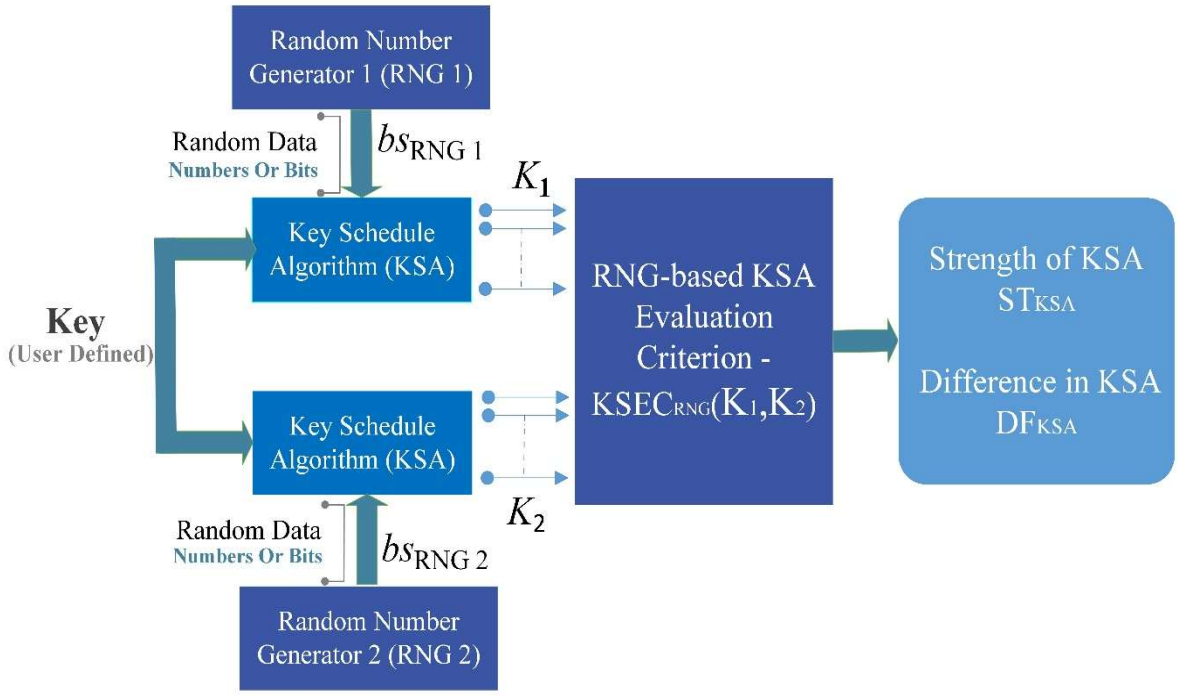



*Figure 2 Block diagram of CryptoQNRG*

We use the following notation in the equations below:

$Key$ ... user-defined key

$bs_{RNGi}$ ... the bitstring generated by RNG i, with $i \in \{1,2\}$

$K_i$ ... the subkey of length $L$, obtained with the KSA based on RNG i using multiple iterations ($rounds$), with $i \in \{1,2\}$:

$$K_i = KSA(Key, bs_{RNGi}, rounds, L)$$

$K_{i,j}$ ... the j-th bit of subkey $K_i$: $1 \leq j \leq len(K_i)$ for $i \in \{1,2\}$

$len(K_i)$ ... number of bits in $K_i$ , with $i \in \{1,2\}$

(note that for all keys $K_i$ the length $len(K_i)$ $is\ even$)

*a. Frequency Test – $FT(K_1, K_2)$*

The first test to be performed is the frequency test which checks that the number of ones and zeroes in a sequence are the same in order to avoid biasing in the $K_i$. The test measures the distribution of bits in the RNG-based key-schedule algorithms. The sequence is a set of secret subkeys of different block ciphers. A subkey that fails the frequency test is considered weak as it fails the fundamental requirement of randomness, and there is no need to investigate other weaknesses based on the remaining tests. The $K_1$ and $K_2$ are balanced if it satisfies the key bits as;

$$FT(K_1, K_2) = \forall K_i \in \{K_1, K_2\}. \left| \bigcup_{K_{i,j} \in K_i \wedge K_{i,j} = 0} K_{i,j} \right| == \left| \bigcup_{K_{i,j} \in K_i \wedge K_{i,j} = 1} K_{i,j} \right| == \frac{len(K_i)}{2} \quad (1)$$

where, $K_i$ is the key obtained with the KSA based on RNG ($KSA_{RNG\ i}$) with $i \in \{1,2\}$

*b. Bit_Correlation Test – $BCT(K_1, K_2)$*

The second test is the Bit- Correlation test which measures the correlation of bits in the $K_B$. This evaluation is divided into two parts, the *Rogers-Tanimoto* distance measure $R(K_1, K_2)$ and the *Pearson's Correlation* $r_{K_1,K_2}$.

The first part computes the dissimilarity index between the two different RNG key-schedules with the *Rogers-Tanimoto* distance measure $R(K_1, K_2)$ [35]:

$$R(K_1, K_2) = \frac{T}{C_{11} + C_{00} + T} \quad (2)$$

where, $C_{pq}$ is the number of corresponding pairs of elements in $K_1$ and $K_2$ respectively equal to p and q.
and $T = 2(C_{10} + C_{01})$

The second part calculates the *Pearson's Correlation* $r_{K_1,K_2}$ [36]. Based on that we test whether the key schedules $K_1$, $K_2$ are independent (null hypothesis $H_0$) or dependent (alternative hypothesis $H_A$).

The *Pearson's Correlation* $r_{K_1,K_2}$ will be calculated as follows:

$$r_{K_1,K_2} = \frac{\sum_{j=1}^{len(K_i)}(K_{1,j} - \overline{K_1})(K_{2,j} - \overline{K_2})}{\sqrt{\sum_{j=1}^{len(K_i)}(K_{1,j} - \overline{K_1})}\sqrt{\sum_{j=1}^{len(K_i)}(K_{2,j} - \overline{K_2})}} \quad 3$$

where $\overline{K_i}$ is the mean value of $K_{i,j}$ with j = 1 ... $len(K_i)$.

Performing the Hypothesis Test based on the P-value generated by $r_{K_1,K_2}$:

- Null hypothesis $H_0$: $K_1$ and $K_2$ are dependent on each other

$$H_0 = (r_{K_1,K_2} \leq 0.1) \quad 4$$

- Alternative hypothesis $H_A$: $K_1$ and $K_2$ are independent of each other

$$H_A = (r_{K_1,K_2} > 0.1) \quad 5$$

The Bit_Correlation Test $BCT(K_1, K_2)$ combines the two parts and is calculated as follows:

$$BCT(K_1, K_2) = \begin{cases} Pass, & H_0 \wedge (R(K_1, K_2) \leq 0.5) \\ Fail, & otherwise \end{cases} \quad 6$$

The advantage of using $R(K_1, K_2)$ to compute dissimilarity is that it calculates the value of symmetric binary attributes. Symmetric binary attributes mean when both attributes have the same significance. It means the input to this test; both the bits 0 and 1 are equally significant. If $K_1$ and $K_2$ passes the frequency test, then only $R(K_1, K_2)$ is computed. The next part $r_{K_1,K_2}$ determines the exact extent of the linear corelation between $K_1$ and $K_2$. Linear relationships occur when one variable change proportionally with another.

Therefore, $BCT(K_1, K_2)$ combines the linearity and dissimilarity test for $K_1$ and $K_2$. The $K_1$ and $K_2$ are considered to be acceptable if the dissimilarity index of the $R(K_1, K_2)$ is greater than or equal to threshold value of 0.5 and it is not linearly dependent on any each other subkey.

*c. Bit-Interfold Test – $BIT(K_1, K_2)$*

The third test is the Bit-Interfold test $BIT(K_1, K_2)$, which measures the confusion and diffusion in the $K_1$ and $K_2$, which is an important cryptographic property. This test is divided into two parts.

The first part is the calculation of the Hamming Distance $H(K_1, K_2)$ between the two subkeys $K_1$and $K_2$. The Hamming Distance is a dissimilarity distance [37].
$H(K_1, K_2)$ is calculated as the number of bit positions $K_{1,j}$ in $K_1$ that are different to those $K_{2,j}$ in $K_2$, divided by the subkey length:

$$H(K_1, K_2) = \frac{\left|\{K_{1,j} \mid K_{1,j} \neq K_{2,j} \wedge (0 \leq j < len(K_1))\}\right|}{len(K_1)} \quad 7$$

The Inverse Hamming Distance $\overline{H(K_1, K_2)}$, which is the inverse of $H(K_1, K_2)$, i.e., counting the number of equal bit positions, is calculated as follows:

$$\overline{H(K_1, K_2)} = len(K_1) - H(K_1, K_2) \quad (8)$$

$\overline{H(K_1, K_2)}$ refers to whether the bits' position will produce the same proportion of confusion and diffusion in $K_1$ and $K_2$. Confusion refers to the process of combining subkey bits with plain text make a cipher. Diffusion refers to the change in a plaintext resulting in changing the bit order in the subkeys $K_1$ and $K_2$. To analyse this proportion of similar bits leading to complex subkeys in $K_1$ and $K_2$ that are the basis of confusion and diffusion, the second part Z-Proportion [38] statistics hypothesis, is introduced.

In the second part, we use the calculated *Inverse Hamming Distance* $\overline{H(K_1, K_2)}$ to evaluate the Z-Proportion [38] statistics hypothesis. The analysis checks whether the confusion and diffusion will be similar or not by $K_1$ and $K_2$.

The Z-proportion test is calculated as follows:

$$Z(K_1, K_2) = \frac{\overline{H(K_1, K_2)} - k_0}{\sqrt{\frac{k_0(1-k_0)}{len(K_1)}}} \quad (9)$$

where $k_0$ is the hypothesized value of population proportion in the null hypothesis, i.e., it is the acceptance threshold of the Hamming Distance. n is the sample size.
The value of $k_0$ is 0.7 as at least 70 percent of bits differ in $K_1$ and $K_2$ to justify the bits responsible for confusion and diffusion, and KSA is considered strong.

The *Null Hypothesis* ($H_0$) and *Alternative Hypothesis* ($H_A$) based on a P-value computed by $Z(K_1, K_2)$ are as follows:

$H_0$ = Confusion and Diffusion is similar in $K_1$ and $K_2$.

$$H_0 = Z(K_1, K_2) \leq 0.7 \quad (10)$$

$H_A$ = Confusion and Diffusion is not similar in $K_1$ and $K_2$.

$$H_A = Z(K_1, K_2) > 0.7 \quad (11)$$

The Bit_Interfold Test $BIT(K_1, K_2)$ is calculated as:

$$BIT(K_1, K_2) = \begin{cases} Pass, & H_A \\ Fail, & H_0 \end{cases} \quad (12)$$

The advantage of using $H(K_1, K_2)$ to compute dissimilarity is that it calculates the value of the exact number of different bits in $K_1$ and $K_2$. The inversion of $H(K_1, K_2)$, i.e., $\overline{H(K_1, K_2)}$ is given to the $Z(K_1, K_2)$ to analyse the proportion of bits responsible for generating similar confusion

and diffusion by $K_1$ and $K_2$. $K_1$ and $K_2$ pass the Bit-Interfold Test if the confusion and diffusion with threshold value is not similar in both the KSAs, i.e., the $Z(K_1, K_2)$ results in *Alternative Hypothesis (HA).*

*d. Bit-Entropy Test – $BET(K_1, K_2)$*

The entropy is a measure of a random variable's uncertainty, and it plays a critical role in information theory. The higher the entropy, the greater is the uncertainty in predicting the value of an observation. There are various definitions available for entropy. In this work we use the Shannon entropy [39] (or *entr* for short) and the BiEntropy [40] (or *BiEn* for short).

The *entr* calculates the entropy as the amount of information conveyed when identifying a random outcome. The *BiEn* is a weighted average of the Shannon entropies of the string and the first n − 2 binary derivatives of the string.

The *entr* is advantageous for the larger value of binary strings, whereas the *BiEn* calculation is helpful for the smaller length of binary strings.

The *entr* $E(K_i)$ of $K_1$ and $K_2$, that takes values from the set $A$=$\{K_{i,j}, K_{i,j+1}, \ldots, K_{i,n}\}$ with probability Pr (X=$K_i$) = $K_{i,j}$ for $i \in \{1,2\}$ (keys $K_1, K_2$) and $j \in \{1,2, \ldots, len(K_1)\}$ is defined as:

$$E(K_i) = -\sum_{j=1}^{len(K_i)} p(K_{i,j})\, log_2(K_{i,j}) \qquad 13$$

The *BiEn* $BiEn(K_i)$ is calculated as follows for $K_1$ and $K_2$ distinctly. The *BiEn* value ranges from 0 to 1. When the disorder is more significant in a binary string, the *BiEn* value will be higher.

$$BiEn(K_i) = (1/(2^{n-1} - 1))\left[\sum_{b=0}^{n-2}\left(-p(b).log_2 p(b) - \left(1 - p(b)\right).log_2\left(1 - p(b)\right)\right).2^b\right] \qquad 14$$

where, $p(b)$ is the proportion of 1's in $K_1$ and $K_2$.

The Bit-Entropy Test $BET(K_1, K_2)$ is calculated as follows:

$$BET(K_1, K_2) = \begin{cases} Pass, & E_{K_i} \geq 1.0 \wedge (BiEn(K_i) \geq 0.1) \\ Fail, & otherwise \end{cases} \qquad 15$$

The, $BET(K_1, K_2)$ combines $E_{K_i}$ and $BiEn(K_i)$ to test entropy along with relative and disorder bits of any length in $K_1$ and $K_2$. The $K_1$ and $K_2$ are considered to be pass if the $E(K_i)$ is greater than threshold value of 0.1 and $BiEn(K_i)$is greater than 1.0.

*e.* RNG-based Key Schedule Evaluation Criterion - $KSEC_{RNG}(K_1, K_2)$

The four tests: Frequency-$FT(K_1, K_2)$, Bit_Correlation-$BCT(K_1, K_2)$, Bit_Interfold-$BIT(K_1, K_2)$ and Bit_Entropy-$BET(K_1, K_2)$ together form a test suite – $KSEC_{RNG}(K_1, K_2)$ - to evaluate the strength of the KSA based on RNG.

An RNG-based key schedule evaluation criterion $KSEC_{RNG}(K_1, K_2)$ is illustrated in Table 1. The table shows the required data generation for each test and the corresponding value for the threshold. The table also summarizes the cryptographic properties and the random keys associated with each test. During the test, a key size column specifies the length of the key in bits. In the next session, we have described the RNG-based KSA and data generation to evaluate their strength.

*Table 1 Performed Tests with the RNG-based Key Schedule Evaluation Criterion – $KSEC_{RNG}(K_1, K_2)$ with Key Size $L = 2^N$ with $N \in \{6,7,8\}$*

| Test Type | Number of Keys | Cryptographic Property | Threshold level |
|---|---|---|---|
| **Frequency** | 500 | Balance of 0 and 1 | $\frac{len(K_i)}{2}$ |
| Frequency | | | |
| **Bit_Correlation** | | | |
| Rogers-Tanimoto | 500 | Correlation | 0.5 |
| Pearson Correlation | | | 0.1 |
| **Bit_Interfold** | | | |
| Hamming Distance | 400 | Confusion & Diffusion | 0.7 |
| Z Proportion | | | |
| **Bit_Entropy** | | | |
| BiEntropy: $BiEn(K_i)$ | 50 | Unpredictability | 0.1 |
| entr: $E(K_i)$ | 500 | | 1.0 |

In the next session, we have described the RNG-based KSA and data generation to evaluate their strength.

## 4. The RNG-based KSAs tested with CryptoQNRG

We analysed the KSA of the following five Block ciphers from a symmetric cryptosystem for an experimental purpose: AES, DES, CAST, Camellia, and GOST. The block cipher encrypts data in blocks of specified key size. The KSA key size taken is 128 bits for AES, Camellia, and CAST, 64 Bits for DES, and 256 bits for GOST. The sub keys are extended using the key expansion function of the KSA. These ciphers are proposed by different authors and with different KSA key size.

Rijndael [41] proposed an Advanced Encryption Standard (AES) with three variants AES-128, 192 and 256 based on KSA key length sizes of 128, 192, and 256 respectively, all of which were approved by National Institute of Standards and Technology(NIST). Endre Bangerter et al. [42] were able to recover AES-128 encryption keys in 2010. The second block cipher, DES [43] with KSA key size of 64 bits, is based on the Balanced Feistel structure and was proposed by IBM. Biham and Shamir [6] proposed a differential cryptanalysis attack on complete rounds of DES.

CAST [44], is based on the Feistel Network (FN) with a KSA key size of 40 to 128 bits. The fourth cipher, Camellia [45], was developed by Mitsubishi Electric and NTT of Japan, based on the FN algorithm with a KSA key size of 128, 192 or 256 bits. The final cipher, GOST [46], supports KSA key sizes of 256 bits. Nicolas Courtois et al. [47] proposed a Contradiction Immunity to attack the complete 32 - rounds of GOST cipher in 2011.

The RNG-based key-schedule also depends on different entropy based on its generator. In our set up we have taken CSPRNG and QRNG to evaluate the cryptographic strength. The entropy within the system is used to provide pseudo-random bits for the key-schedule ($KSA_{PK}$) that is required to create the keys. In QRNG, photons are used to generate quantum random bits for the key schedule ($KSA_{QK}$). The bits length of the generated random key depends on the key size of the KSA. The subkeys are generated using the cryptol [48] language. The results will demonstrate the strength of KSA in terms of cryptographic parameters for each block ciphers.

We use the same notation for result analyses:

$K_1$ ... the subkey obtained with the KSA based on QRNG

$K_2$ ... the subkey obtained with the KSA based on PRNG

To generate subkey $K_1$ of size $L$ we take the bitstream of the QRNG and combine it with the user-defined key (Key) with multiple KSA iterations (rounds):

$$K_1 = KSA\left(Key, bs_{QRNG}, rounds, L\right)$$

Analogously, to generate subkey $K_2$ of size $L$ we take the bitstream of the PRNG and combine it with the user-defined key (Key) with multiple KSA iterations (rounds):

$$K_2 = KSA(Key, bs_{PRNG}, rounds, L)$$

In our experiment we use KSA with 11 iterations ($rounds$) and subkey-size $L = 2^N$ with $N \in \{6,7,8\}$:

$$K_1 = KSA\left(Key, bs_{QRNG}, 11, 2^N\right) \mid N \in \{6,7,8\}$$

$$K_2 = KSA(Key, bs_{PRNG}, 11, 2^N) \mid N \in \{6,7,8\}$$

where the subkey-size $L = 2^N$ depends on the concrete block cipher length, so for DES we have $N = 6$ ($L = 2^6 = 64$), for AES, Camellia, and CAST we have $N = 7$ ($L = 2^7 = 128$), and for GOST we have $N = 8$ ($L = 2^8 = 256$).

In this study, we use two sets of data, one for the Frequency and Bit_Entropy test, where we used random subkeys, and another set for Bit_Correlation and Bit_Interforld, where we used the samples of subkeys to test the hypotheses.

Finally, the key schedule evaluation criterion $KSEC_{RNG}(K_1, K_2)$ is used to evaluate the cryptographic strength of $K_1$ and $K_2$.

## 5. Results and Analysis

The results and data are covered in this section, where the result is drawn on the strength of the $KSA_{RNG}$. We have used the following hardware and software to implement the proposed framework.

*Hardware:* Quantis [49]– A USB-based Quantum random number generators developed by IDQ Its general specifications include - Random bit rate 1 : 4 Mbit/s ± 10% (Quantis-USB-4M), Thermal noise contribution: < 1% (Fraction of random bits arising from thermal noise), Storage temperature : - 25 to + 85°C, USB specification 2.0 and Power Via USB port

Computer - Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM: 8.00 GB, System type: 64-bit operating system, x64-based processor

*Software:* Java – Netbeans with JDK 1.8.0

*a. Frequency Test*

Table 2 displays the results of the frequency test. The number of bits is balanced in both $K_1$ and $K_2$. The numbers of bits in the Quantum $K_1$ and Pseudo-based $K_2$ key-schedule integrate the equal number of 0's and 1's. The table shows that the $FT(K_1, K_2)$ of each block cipher passes the frequency test. However, the frequency test alone cannot predict the strength of the RNG-based key-schedule.

*Table 2 Frequency analysis $FT(K_1, K_2)$ of Bits in $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$ Schedule of five block ciphers.*

| | **AES** | **Camellia** | **CAST** | **DES** | **GOST** |
|---|---|---|---|---|---|
| Ratio of Percentage of 0:1 in $K_1$ | 50:50 | 50:50 | 50:50 | 50:50 | 50:50 |
| Ratio of Percentage of 0:1 in $K_2$ | 50:50 | 50:50 | 50:50 | 50:50 | 50:50 |

*The Number of 0's and 1's is compared in $K_1$ and $K_2$ schedules of five different block ciphers. The statistical analysis shows the bits are balanced in $K_1$ and $K_2$ for all the ciphers.*

*b. Bit – Correlation Test*

Bit correlation tests the strength in terms of the correlation while taking the Pearson's correlation hypothesis test in conjunction with the Rogers-Tanimoto distance measure. The graph in Figure 3 shows the changes in the Rogers-Tanimoto distance measure in $K_1$ and $K_2$. The results show that the dissimilarity index of CAST is the lowest of all, whereas the GOST key-schedule shows the highest. The index of DES is slightly less then Camellia and AES with 0.66589.

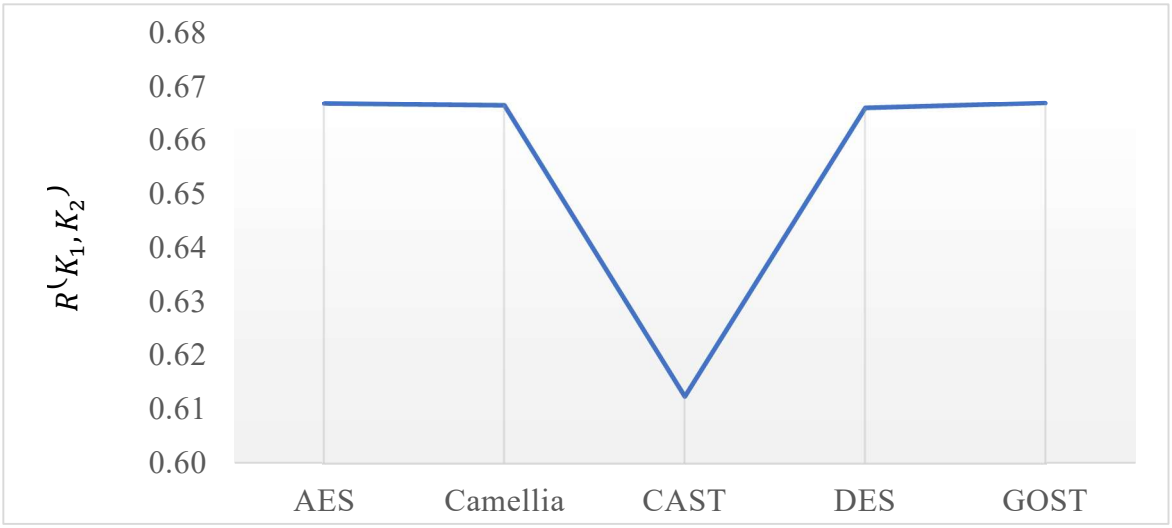



*Figure 3 Rogers-Tanimoto distance measure $R(K_1, K_2)$ of five block ciphers.*
*The dissimilarity index of $K_1$ and $K_2$ is compared for five different block ciphers. Statistical analysis shows the bits are nearly thirty percent similar in both the key-schedule for four ciphers and forty percent in CAST; among all the ciphers.*

.

Table 3 shows the Pearson's Correlation hypothesis testing result of each block cipher. The values of the $r_{K_1,K_2}$ correspond to P-value statistics. The P-value of AES, Camellia, DES and GOST subkeys passes the threshold value of 0.1; therefore, we reject the null hypothesis that the $K_1$ and $K_2$ key-schedules are dependent on each other for these ciphers. On the other hand, CAST subkeys are failed to pass the threshold value. This means the $K_1$ and $K_2$ are dependent of each other and do not pass the Bit Correlation – $BCT(K_1, K_2)$ test. This shows that CAST keys are weak and susceptible for key-dependent [5] and correlation [50] attacks with both, Quantum and Pseudo random number-based key-schedules.

*Table 3 Correlation of bits of $K_1$ and $K_2$ of five block ciphers based on Pearson's Correlation Hypothesis Test.*

| | **AES** | **Camellia** | **CAST** | **DES** | **GOST** |
|---|---|---|---|---|---|
| $r_{K_1,K_2}$ | 0.979 | 0.889 | 0.076 | 0.745 | 0.808 |
| $H_0$ *or* $H_A$ | Independent | Independent | Dependent | Independent | Independent |

*c. Bit – Interfold Test*

Table 4 shows the results of the Bit-Interfold test. The test first calculates the Hamming Distance of the $K_1$ and $K_2$ based on a 64, 128 and 256-bit key size with a sample of 400 keys. The result of the Hamming Distance measures dissimilarity between the $K_1$ and $K_2$, and the inverse of which is then passed to one Z-Proportions hypothesis testing and the corresponding P-values are calculated.

Camellia and CAST result in the alternative hypothesis-$H_A$ (taken from below result Table 3), which means they pass the Bit-interfold test. Any KSA that fails this test will create a weak cipher which is vulnerable to an easy cryptoanalysis. For example, AES, DES and GOST failed the $BIT(K_1, K_2)$ test and showed that they are weak and vulnerable to attacks such as related-key and side-channel [51] attacks.

*Table 4 Confusion and Diffusion of $K_1$ and $K_2$ of five block ciphers based on Hamming Distance and Z - Proportion Hypothesis Test*

| | **AES** | **Camellia** | **CAST** | **DES** | **GOST** |
|---|---|---|---|---|---|
| H($K_1, K_2$) | 27661 | 28502 | 31676 | 25623 | 27461 |
| $Z(K_1, K_2)$ | $H_0$ | $H_A$ | $H_A$ | $H_0$ | $H_0$ |

*d. Bit – Entropy Test*

The most critical parameter for a key in order to be secure is unpredictability. The $K_1$ and $K_2$ are tested with two different entropy tests. The $K_1$ shows better entropy than the $K_2$, as shown in Figure 4 and Figure 5. The variations in entropy resulted in different values with the *entr* and Bi_Entropy tests, one for each $K_1$ and $K_2$, and same were analysed against the threshold value. The distinct entropy values of each block cipher exceed the threshold value of 1.0 for *entr* and 0.1 for Bi_Entropy; hence, all the $K_1$ and $K_2$ pass the entropy test – $BEnT(K_1, K_2)$.

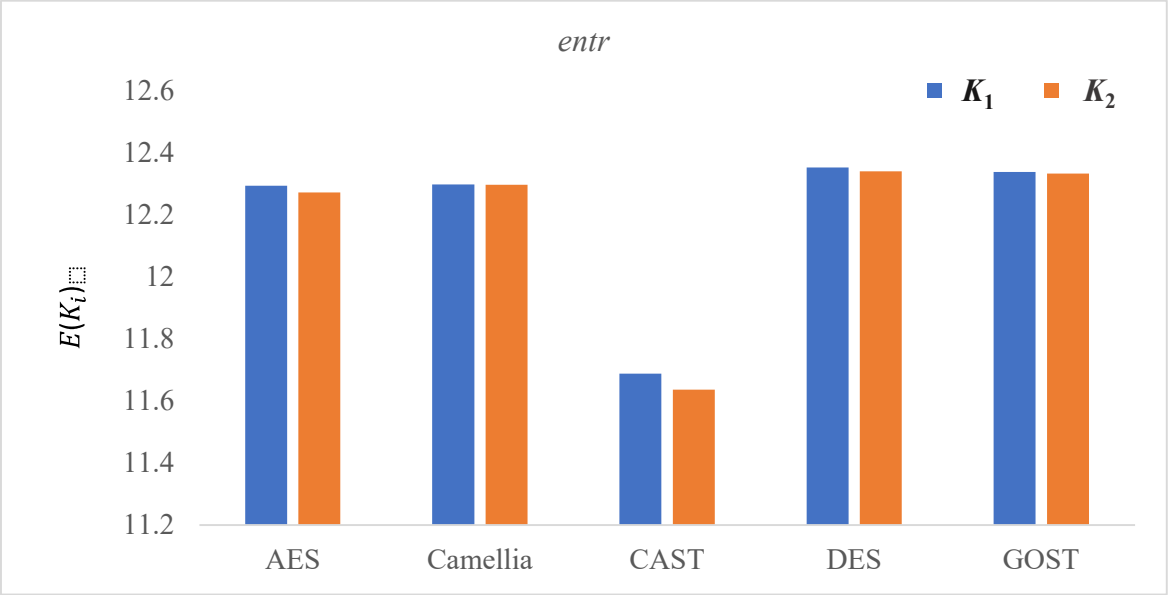



*Figure 4 Entropy analysis entr $E(K_i)$ for $K_1$ and $K_2$ using five block ciphers.*

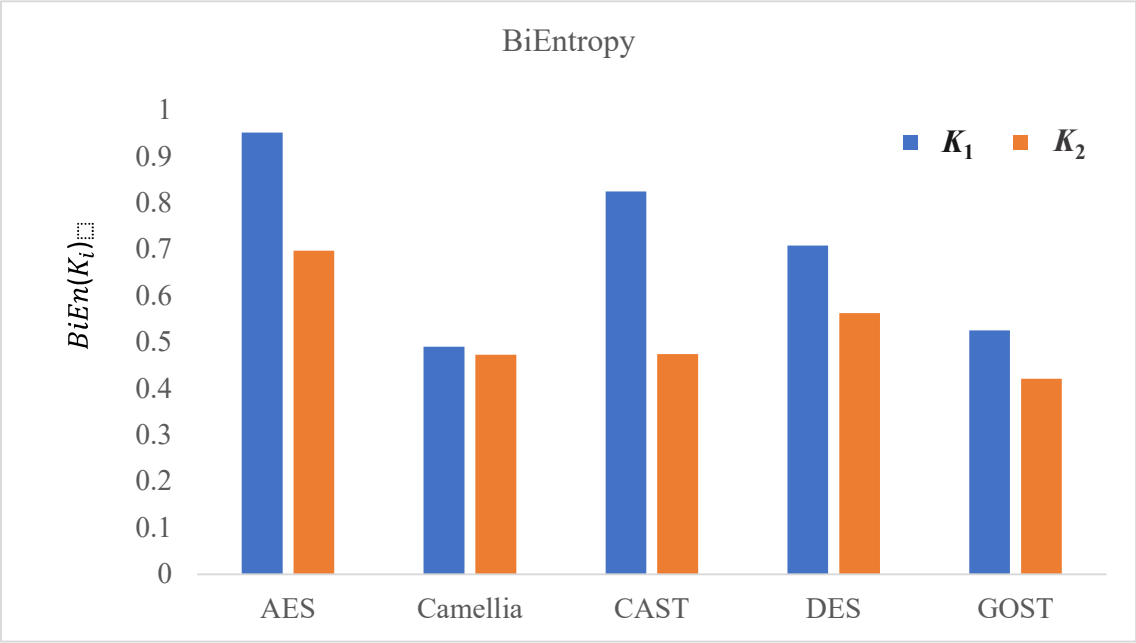



*Figure 5 Bi_ Entropy analysis $BiEn(K_i)$ for $K_1$ and $K_2$ using five block ciphers.*

*The $K_1$ and $K_2$ of five different block ciphers are compared with the 500 and 50 different subkeys using two entropy tests: entr and Bi_Entropy. Statistical analysis shows that the key-schedule generated by quantum random bits are more unpredictable than pseudo random for all the block ciphers.*

The analysis also proves that the quantum random number-based($K_1$) key-schedule is more unpredictable than the pseudo-random($K_2$) one. Unpredictability increases the $K_1$ schedule's strength, making it strong and hard to do cryptanalysis to partially access the key with Related-Key and Fault-Injection Attacks[52].

*e. Encryption Time*

The encryption time for all of the block ciphers was calculated to evaluate the impact of the $K_1$ and $K_2$. We used two different file sizes, 8 MB and 16 MB, to illustrate the time required to convert plain text to ciphertext. The encryption time for each file size can be seen in Figure 6. DES takes the longest time to encrypt, while GOST takes the second-longest time. The minimum time computation is by AES in both the $K_1$ and $K_2$ schedules. The analysis also shows that quantum and pseudo-based key-schedules are taking nearly the same time for encryption. All the ciphers showed nearly the same transformation time with $K_1$ and $K_2$, with AES taking the least time among all the ciphers for both the schedules.

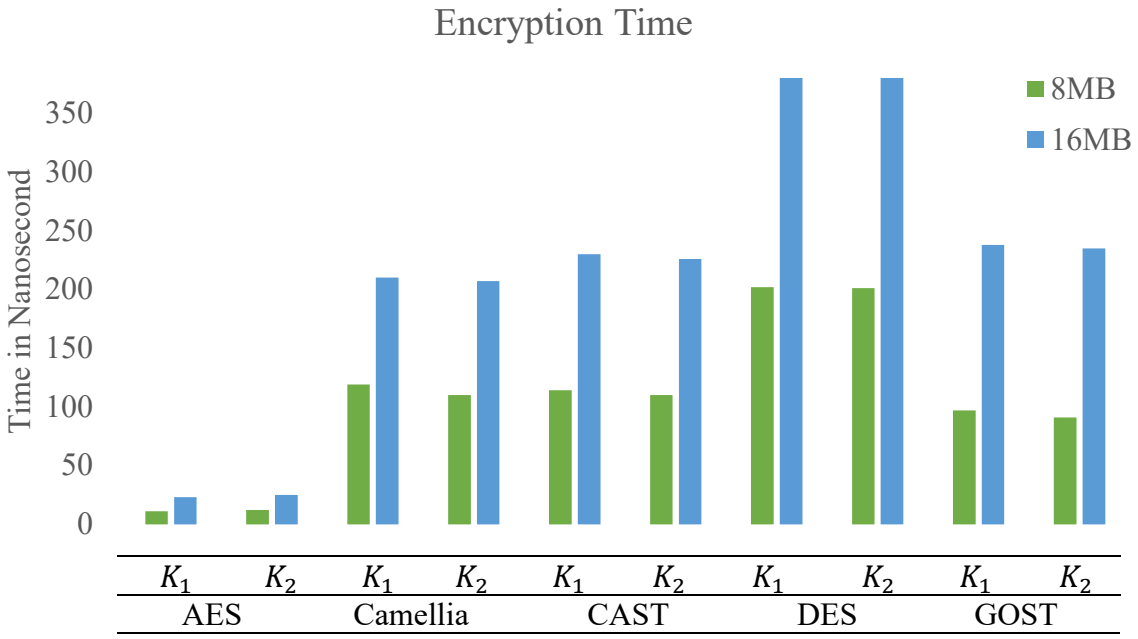



*Figure 6 Encryption Time of plain text with $K_1$ and $K_2$ with file size of 8 MB and 16 MB.*

*The time of converting plain text to cipher text with the help of Quantum and Pseudo based key-schedule is measured in nanoseconds.*

## 6. Conclusion

We proposed CryptoQNRG, a new framework in order to evaluate the strength of RNG-based Key Schedules using four tests: Frequency, Bit_Correlation, Bit-Interfold, and Entropy. The test suite evaluates the resilience of subkeys of KSA in terms of the balance of 0 and 1's, correlation of bits, confusion and diffusion, and an essential parameter of security, uncertainty.

The proposed CryptoQNRG evaluates and assesses the subkeys of the most common KSA with quantum and pseudo-random numbers. The main focus of the paper is to compare the strength of KSA based on RNGs, as compared to Afzal et al. [33], who evaluated the subkeys without considering them. The results indicate the strength of Quantum- and Pseudo-based key-schedules and their cryptographic properties. The results show that CAST did not pass the Bit_Correlation test, and keys are prone to cipher attacks. The analysis also indicates that the AES, DES, and GOST did not pass the Bit-Interfold test, whereas CAST and Camellia did. However, the computational time required to generate a cipher with a quantum random number-based($K_1$) key-schedule and pseudo-random($K_2$) of AES is much faster than the

others. The results also revealed that a quantum-based key is less predictable than a pseudo-random number-based key.
The future work of the study includes testing the KSA of lightweight cryptographic algorithms that play a major role in the field of the internet of things (IoT).

**Declarations**

**Ethical Approval:** Not Applicable

**Competing interests:** The authors declare that they have no conflict of interest.

**Authors' contributions:** A.S. and A.T. devised the idea presented here. A.S. developed the theory, performed the computations, and prepared figures. R.K and A.S. verified the analytical methods. All authors reviewed the manuscript.

**Funding:** This research work received funding by University of Hertfordshire, United Kingdom.

**Data Availability Statements:** The data used to support the findings of this study are included in this article.

## References

[1] W. Stallings, "Cryptography and Network Security: Principles and Practices," *Cryptography and Network Security*. Pearson, USA, 2005.

[2] K. Verma and D. K. Sharma, "Calculation of non-linearity and algebraic degree of constructed boolean function," in *2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, May 2017, pp. 501–505, doi: 10.1109/RTEICT.2017.8256647.

[3] F. L. Shi and H. Bin, "Propagation properties of symmetric Boolean functions," in *International Conference on Intelligent Computation Technology and Automation*, May 2010, pp. 947–950, doi: 10.1109/ICICTA.2010.614.

[4] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," in *Advances in Cryptology – ASIACRYPT Lecture Notes in Computer Science*, Springer, 2009, pp. 1–18.

[5] K. B. Jithendra and T. K. Shahana, "New Results in Related Key Impossible Differential Cryptanalysis on Reduced Round AES-192," in *2018 International Conference On Advances in Communication and Computing Technology, ICACCT 2018*, Feb. 2018, pp. 291–295, doi: 10.1109/ICACCT.2018.8529666.

[6] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, Jan. 1991, doi: 10.1007/BF00630563.

[7] N. P. Smart, V. Rijmen, B. Warinschi, and G. Watson, "Algorithms, Key Sizes and Parameters Report," *Report*. ENISA, Nov. 2014, Accessed: Sep. 09, 2021. [Online]. Available: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014.

[8] J. Lee, Y. Seo, and J. Heo, "Analysis of random number generated by quantum noise source and software entropy source," *In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Jeju, Korea (South), pp. 729–732, Oct. 17, 2018, doi: 10.1109/ICTC.2018.8539618.

[9] M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum random number generators," *Reviews of Modern Physics*, vol. 89, no. 1, p. 015004, Feb. 2017, doi: 10.1103/RevModPhys.89.015004.

[10] T. Lunghi *et al.*, "Self-Testing Quantum Random Number Generator," *Phys. Rev. Lett.*, vol. 114, no. 15, p. 150501, Apr. 2015, doi: 10.1103/PhysRevLett.114.150501.
[11] H. Xu, D. Perenzoni, A. Tomasi, and N. Massari, "A 16 × 16 Pixel Post-Processing Free Quantum Random Number Generator Based on SPADs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 627–631, 2018, doi: 10.1109/TCSII.2018.2821904.
[12] R. C. Pooser, P. G. Evans, and T. S. Humble, "Self correcting quantum random number generators using tapered amplifiers," *In Proceedings of the IEEE Photonics Society Summer Topical Meeting Series*. IEEE, Waikoloa, HI, USA, pp. 147–148, Jul. 08, 2013, doi: 10.1109/PHOSST.2013.6614471.
[13] J. M. Wang, T. Y. Xie, H. F. Zhang, D. X. Yang, C. Xie, and J. Wang, "A bias-free quantum random number generation using photon arrival time selectively," *IEEE Photonics Journal*, vol. 7, no. 2, 2015, doi: 10.1109/JPHOT.2015.2402127.
[14] Y.-H. Li *et al.*, "Quantum random number generation with uncharacterized laser and sunlight," *npj Quantum Information*, vol. 5, no. 1. p. 97, Dec. 14, 2019, doi: 10.1038/s41534-019-0208-1.
[15] C. Abellán *et al.*, "Ultra-fast quantum randomness generation by accelerated phase diffusion in a pulsed laser diode," *Optics Express*, vol. 22, no. 2, p. 1645, 2014, doi: 10.1364/oe.22.001645.
[16] ID Quantique, "What is the Q in QRNG ?" 2020, Accessed: Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/random-number-generation/overview/.
[17] G. Shaw, S. R. Sivaram, and A. Prabhakar, "Quantum Random Number Generator with One and Two Entropy Sources," *In Proceedings of the National Conference on Communications (NCC)*. IEEE, Bangalore, India, pp. 1–4, Feb. 20, 2019, doi: 10.1109/NCC.2019.8732222.
[18] G. Mogos, "Quantum Random Number Generator vs. Random Number Generator," in *IEEE International Conference on Communications*, Jun. 2016, pp. 423–426, doi: 10.1109/ICComm.2016.7528306.
[19] ID Quantique, "Understanding Quantum Cryptography." ID Quantique SA, 2020, Accessed: Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/quantum-safe-security/quantum-key-distribution/.
[20] IDQ, "Quantum versus Classical Random Number Generators." Switzerland, May 2020.
[21] ID Quantique, "gaming-and-lotteries." Accessed: Jul. 07, 2020. [Online]. Available: https://www.idquantique.com/random-number-generation/applications/gaming-and-lotteries/.
[22] A. R. Hakim and Z. Z. Nusron, "An improved Lblock-s key schedule algorithm," in *International Conference on Information and Communications Technology*, Jul. 2019, pp. 232–236, doi: 10.1109/ICOIACT46704.2019.8938569.
[23] S. M. Kareem and A. M. S. Rahma, "A novel approach for the development of the Twofish algorithm based on multi-level key space," *Journal of Information Security and Applications*, vol. 50, Feb. 2020, doi: 10.1016/j.jisa.2019.102410.
[24] S. Sulaiman, Z. Muda, J. Juremi, R. Mahmod, and S. M. Yasin, "A New ShiftColumn Transformation : An Enhancement of Rijndael Key Scheduling," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 3, pp. 160–166, 2013.
[25] J. Huang, H. Yan, and X. Lai, "Transposition of AES key schedule," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10143 LNCS, pp. 84–102, 2017, doi: 10.1007/978-3-319-54705-3_6.
[26] R. Shahzadi, S. M. Anwar, F. Qamar, M. Ali, and J. J. P. C. Rodrigues, "Chaos based enhanced RC5 algorithm for security and integrity of clinical images in remote health monitoring," *IEEE Access*, vol. 7, Apr. 2019, doi: 10.1109/ACCESS.2019.2909554.

[27] S. Sahmoud, W. Elmasry, and A. Shadi, “Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher,” *International Arab Journal of e-Technology*, vol. 3, no. 1, pp. 17–26, Jan. 2013.
[28] B. Maram and J. M. Gnanasekar, “A Block Cipher Algorithm to Enhance the Avalanche Effect Using Dynamic Key-Dependent S-Box and Genetic Operations,” *International Journal of Pure and Applied Mathematics*, vol. 119, no. 10, pp. 399–418, 2018.
[29] R. Saha, G. Geetha, G. Kumar, and T. H. Kim, “RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys,” *Security and Communication Networks*, vol. 2018, pp. 1–11, Nov. 2018, doi: 10.1155/2018/9802475.
[30] A. Vuppala, R. S. Roshan, S. Nawaz, and J. V. R. Ravindra, “An Efficient Optimization and Secured Triple Data Encryption Standard Using Enhanced Key Scheduling Algorithm,” in *Procedia Computer Science*, Jun. 2020, vol. 171, pp. 1054–1063, doi: 10.1016/j.procs.2020.04.113.
[31] G. Leurent and C. Pernot, “New Representations of the AES Key Schedule,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12696 LNCS. pp. 54–84, 2021, doi: 10.1007/978-3-030-77870-5_3.
[32] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, “Strengthening the key schedule of the AES,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2384, pp. 226–240, 2002, doi: 10.1007/3-540-45450-0_19.
[33] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, “Cryptographic Strength Evaluation of Key Schedule Algorithms,” *Security and Communication Networks*, pp. 1–9, May 2020, doi: 10.1155/2020/3189601.
[34] S. Afzal, U. Waqas, M. A. Mir, and M. Yousaf, “Statistical Analysis of Key Schedule Algorithms of Different Block Ciphers,” *Science International - Report*. Jun. 2015.
[35] M. M. Chatzimichailidou and I. M. Dokas, “RiskSOAP: On the Relationship between Systems Safety and the Risk SA Provision Capability,” *IEEE Systems Journal*, vol. 12, no. 2, pp. 1148–1157, 2018, doi: 10.1109/JSYST.2016.2614953.
[36] P. Socha, V. Miskovsky, H. Kubatova, and M. Novotny, “Optimization of Pearson correlation coefficient calculation for DPA and comparison of different approaches,” in *International Symposium on Design and Diagnostics of Electronic Circuit and Systems*, Apr. 2017, pp. 184–189, doi: 10.1109/DDECS.2017.7934563.
[37] T. S. Community, “hamming.” https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html (accessed Jul. 09, 2020).
[38] E. Volchok, “Clear-Sighted Statistics: Module 14: One-Sample Hypothesis Tests (slides).” City University of New York (CUNY)., 2020.
[39] S. Vajapeyam, “Understanding Shannon’s Entropy metric for Information,” no. March, pp. 1–6, Mar. 2014, doi: https://doi.org/10.48550/arXiv.1405.2061.
[40] G. J. Croll, “Bientropy, TriEntropy and primality,” *Entropy*, vol. 22, no. 3, Mar. 2020, doi: 10.3390/e22030311.
[41] J. Daemen and V. Rijmen, *The Design of Rijndael*. Springer Berlin Heidelberg, 2002.
[42] D. Gullasch, E. Bangerter, and S. Krenn, “Cache games - Bringing access-based cache attacks on AES to practice,” in *IEEE Symposium on Security and Privacy*, May 2011, pp. 490–505, doi: 10.1109/SP.2011.22.
[43] A. Biryukov and C. Cannière, “Data encryption standard (DES),” *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, USA, Oct. 1999, doi: 10.1007/0-387-23483-7_94.
[44] C. Adams, “The CAST-128 Encryption Algorithm.” 1997, Accessed: Jun. 12, 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc2144.

[45] “Japan’s First 128-bit Block Cipher ‘Camellia’ Approved as a New Standard Encryption Algorithm in the Internet.” NTT News Release, Accessed: Jul. 17, 2021. [Online]. Available: https://www.ntt.co.jp/news/news05e/0507/050720.html.
[46] C. Cannière, “GOST,” *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, 2011, doi: 10.1007/978-1-4419-5906-5_579.
[47] N. T. Courtois, J. A. Gawinecki, and G. Song, “Contradiction Immunity and Guess-Then-Determine Attacks on Gost,” *Tatra Mountains Mathematical Publications*, vol. 53, no. 1, pp. 65–79, Dec. 2013, doi: 10.2478/v10127-012-0039-3.
[48] “Cryptol.” Galois, Inc., [Online]. Available: https://cryptol.net/.
[49] IDQ, “quantis-random-number-generator,” 2020. https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator (accessed Jul. 07, 2020).
[50] N. N. Anandakumar and S. Dillibabu, “Correlation power analysis attack of AES on FPGA using customized communication protocol,” in *International Conference on Computational Science, Engineering and Information Technology*, Oct. 2012, pp. 683–688, doi: 10.1145/2393216.2393330.
[51] Y. Niu, J. Zhang, A. Wang, and C. Chen, “An Efficient Collision Power Attack on AES Encryption in Edge Computing,” *IEEE Access*, vol. 7, pp. 18734–18748, 2019, doi: 10.1109/ACCESS.2019.2896256.
[52] Y. Li, M. Chen, Z. Liu, and J. Wang, “Reduction in the number of fault injections for blind fault attack on SPN block ciphers,” *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, pp. 1–20, Apr. 2016, doi: 10.1145/3014583.

# Appendix III : Published paper III - QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption

Sci

CITESCORE
5.2

Article

# QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption

Anish Saini, Athanasios Tsokanos and Raimund Kirner

MDPI

https://doi.org/10.3390/sci6040086

*Sci*

MDPI

*Article*

# QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption

**Anish Saini *, Athanasios Tsokanos and Raimund Kirner**

School of Physics, Engineering and Computer Science (SPECS), University of Hertfordshire, Hertfordshire AL10 9AB, UK; a.tsokanos@herts.ac.uk (A.T.); r.kirner@herts.ac.uk (R.K.)
* Correspondence: a.saini@herts.ac.uk

**Abstract:** Cloud computing has revolutionized the digital era by providing a more efficient, scalable, and cost-effective infrastructure. Secure systems that encrypt and protect data before it is transmitted over a network and stored in the cloud benefit the entire transmission process. Transmission data can be encrypted and protected with a secure dynamic substitution box (S-box). In this paper, we propose the QuantumGS-box, which is a dynamic S-box for high-speed cloud-based storage encryption generated by bit shuffling with a genetic algorithm and a quantum random number generator (QRNG). The proposed work generates the S-box optimized values in a dynamic way, and an experimental evaluation of the proposed S-box method has been conducted using several cryptographic criteria, including bit independence criteria, speed, non-linearity, differential and linear approximation probabilities, strict avalanche criteria and balanced output. The results demonstrate that the QuantumGS-box can enhance robustness, is resilient to differential and provide improved linear cryptoanalysis compared to other research works while assuring non-linearity. The characteristics of the proposed S-box are compared with other state of the art S-boxes to validate its performance. These characteristics indicate that the QuantumGS-box is a promising candidate for cloud-based storage encryption applications.

**Keywords:** dynamic S-box; QRNG; cloud-based storage; encryption

check for updates

**Citation:** Saini, A.; Tsokanos, A.; Kirner, R. QuantumGS-Box—A Key-Dependent GA and QRNG-Based S-Box for High-Speed Cloud-Based Storage Encryption. *Sci* **2024**, *6*, 86. https://doi.org/10.3390/sci6040086

Academic Editor: Luis Javier Garcia Villalba

Received: 13 October 2024
Revised: 8 December 2024
Accepted: 12 December 2024
Published: 23 December 2024

**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Cloud computing has become a significant part of the IT landscape in the last few years due to the migration from local drive storage to cloud computing storage. This leads to data being stored remotely rather than locally and with interconnected networks, being accessed over the Internet [1]. Transmission of data over a network requires the consideration of a number of factors, including the processing speed and security. Data security is a significant concern as information flows through the internet. Incorporating cryptography [2] that meets secure transmission requirements ensures that the data is transmitted securely. Secure data transmission over a network transmission can be achieved by cryptography that uses *Key Schedule Algorithms* (KSA) [3] and *encryption* [4].

KSA includes methods of generating a random key typically based on substitution and permutation. The substitution replaces the specific bits with other defined bits, and a *substitution-box* (S-box) is used to implement substitution. Permutation includes the shuffling of bits with specific operations. The S-box performs substitution in both KSA and encryption.

Our approach is shown in Figure 1, where Alice is both the sender and receiver. Alice encrypts the data before sending it to the network receiving the same encrypted data. Encrypting the data before sending it to the network is the most effective way of protecting cloud-based (remote) storage from unauthorised access or attacks.

Figure 2 illustrates a cloud-based storage network that uses a static S-box to implement KSA and encryption.

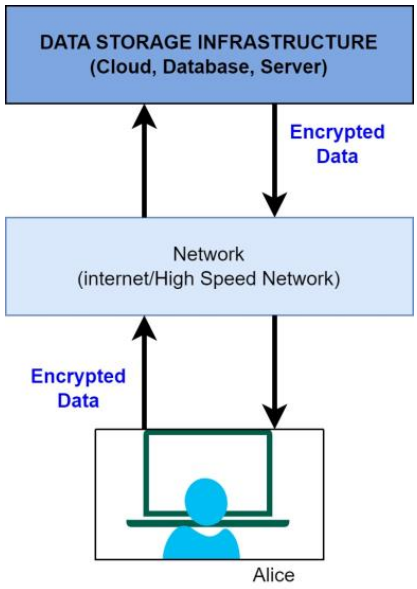
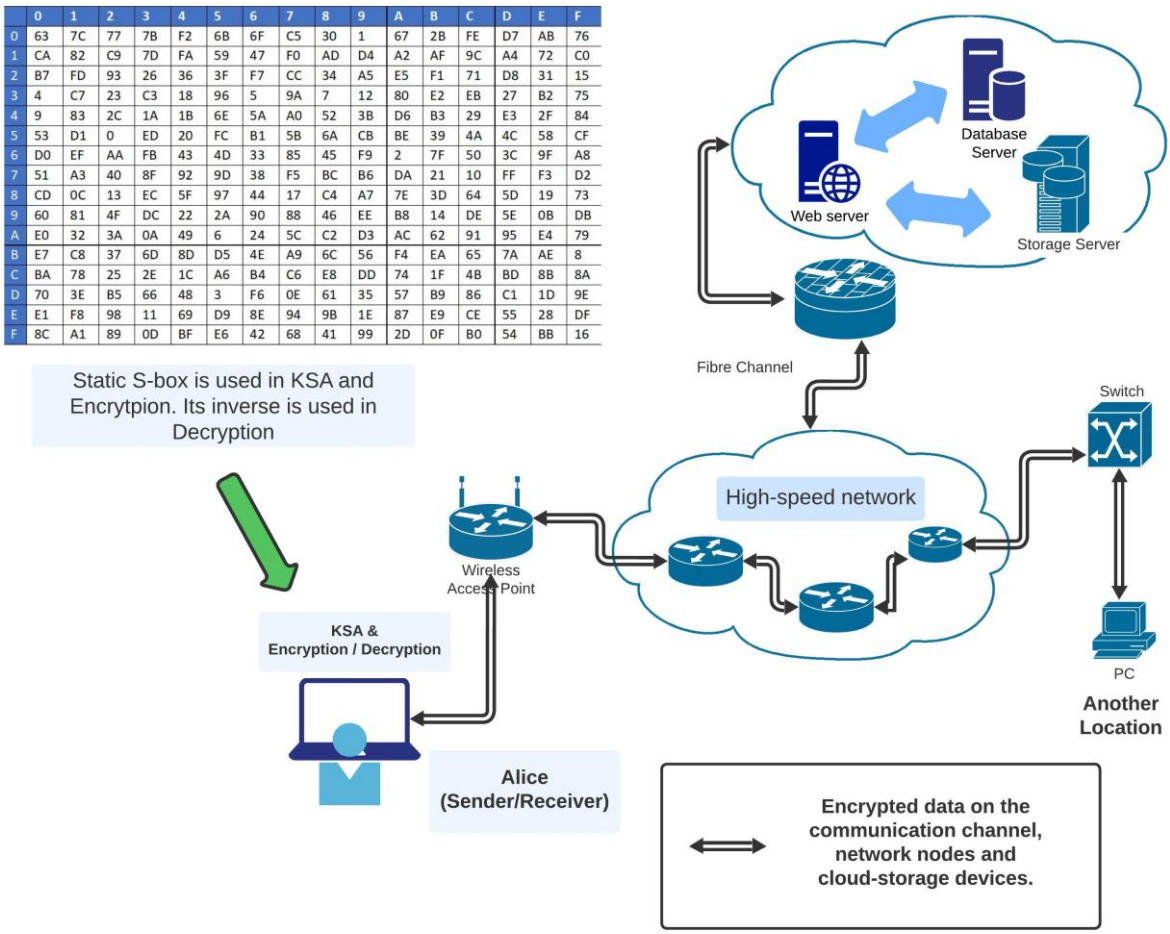



**Figure 1.** Alice is sender and receiver.



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 0 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | 3A | 0A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 3 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |



**Figure 2.** Usage Motivation: AES static S-box in cloud-based storage encryption through high-speed network.

The AES S-box shown is a 16 × 16 matrix of static bits. Using the KSA and S-box, Alice generates the key and encrypts the data. Alice encrypts the data before sending it

to cloud-based storage (remote) through HSN, and then decrypts it after receiving it. The inverse of an S-box is used to decrypt data.

The static S-box is used to provide a substitution of values. The security strength of the system depends on how vulnerable is the key generated by the KSA and its components [5] for the cryptanalysis attack, which is a key point of our research. In the last two decades the concept of a dynamic S-box has become an important research area for enhancing security. The static bits of an S-box make it vulnerable to attack and many researchers have focused on making a dynamic S-box with different scientific approaches.

There are various approaches of creating dynamic S-box with

- *Algebraic theory-based*: A mixture of classical and modern mathematics, Galois Theory processed the solutions of polynomial equations (mathematical equations made of numbers and variables) to form different values of dynamic S-box [6–9].
- *Chaos theory-based:* The theory uses different patterns and mathematical formulations to generate randomness in the system and corresponds to different dynamic values of the S-box [10–12].
- *Random number generator-based*—The random number generated by sources like electrical and quantum noise leads to pseudo, true, and quantum generators, processing the values of dynamic S-box [13,14].
- *Advanced data analysis techniques-based:* Advanced data analysis techniques like machine learning, particle swarm optimization, genetic algorithms, and heuristic techniques analyze the different sources of data and generate the values of the S-box dynamically [15–17].

Conventional AES keys are based on Pseudo-Random Number Generators [18,19] to generate random data to enhance AES's security. RK-AES [20] proposed a Symmetric Random Function Generator to achieve randomness in AES's key. However, the most unpredictable [21] and truly random data [22] can be achieved using a *quantum random number generator* (QRNG) [23].

Miguel Herrero-Collantes et al. explained different types of Quatum Number Generator based on different sources like optical and non-optical QRNG [24]. Xiongfeng Ma et al. categorized the QRNGs into three groups: the practical QRNG (generate randomness at a high speed and built on fully trusted and calibrated devices), self-testing QRNG (randomness without the trusting the actual implementation) and semi-self-testing QRNG (a tradeoff between the trustworthiness on the device and the random number generation speed) [25]. Jinlu Liu et al. proposed a 117 Gbits/s random bit generation rate QRNG using min-entropy estimation and Toeplitz-hashing randomness extraction, based on the quantum phase fluctuation of a distributed feedback (DFB) laser [26]. Meilana Siswanto, et al. designed a photonic-based RNG to enhance the security of an Internet of Things (IoT) system comprised of optical components, analog-digital electronic systems, and an asynchronous transmitter [27]. It also utilizes Verilog firmware to integrate the IoT system. Zhu Cao et al. proposed a source-independent QRNG in which output randomness can be certified even when the source is uncharacterized and untrusted. Figure 3 shows the random number generation process using optical device and generating quantum random numbers.

IDQ created a hardware QRNG for generating true (non-pseudo) randomness. Quantis [22] is a state-of-the-art QRNG, exploiting an optical quantum process as the source of randomness. A QRNG is superior to traditional random number generators as their source of randomness is associated with environmental perturbations [28]. The device has also a monitoring function. If a failure is detected, the random bit stream created is immediately disabled. The device provides true randomness from the first bit, separates Quantum noise from classical noise, auto calibrated and certified [29].

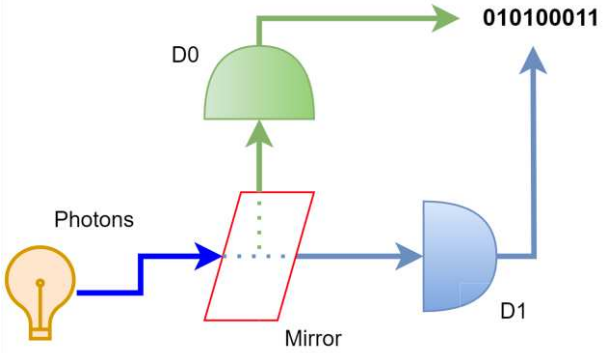



**Figure 3.** Random number generation using optical process as QRNG.

On the other hand, a genetic algorithm [30] is a search heuristic inspired by the natural evolution of Charles Darwin's theory. The algorithm corresponds to the natural selection of selecting the fittest individuals based on a fitness function in order for reproduction to produce offsprings of the next generation. A genetic algorithm requires the following two factors of a domain solution: a genetic representation and a fitness function to evaluate suitability. Initial population, fitness function, selection, crossover, mutation, and termination are phases of a genetic algorithm aiming to an optimal solution.

Sourabh Katoch et al. reviewed various advantages, disadvantages of different GAs like Classical GA, Binary coded GAs, Real-coded GAs, Multiobjective GAs, Parallel GAs, Chaotic GAs and Hybrid GAs [31]. They also discussed various operators used for different GAs. Yong Wang et al. proposed a novel method for constructing S-box by transforming it to a Traveling Salesman Problem and designed S-box based on chaos and genetic algorithm [32]. Another research by Yong Wang et al. proposed a novel genetic algorithm to construct bijective S-boxes with high nonlinearity, taking the nonlinearity of the S-box as the optimization objective [33].

Our research focuses on improving security by creating an S-box that is dynamic using a quantum random number generator and evolutionary computation. The KSA depends on a proposed dynamic S-box based on quantum random bits and a genetic algorithm in order to generate the key.

Our proposed work uses a 128-bits secret key and 128-bits from the QRNG as an initial population and performs crossover and mutation functions to generate different values in the S-box.

The focus of this article is to create a key-dependent dynamic S-box based on quantum randomness and a bit shuffling method. The contributions of this work are:

- Designing an algorithm that takes a user-defined key and quantum random bits from a QRNG and generates a dynamic S-box using bit shuffling with genetic evolution.
- Analysing the proposed algorithm corresponding to a range of cryptographic properties, including nonlinearity, randomness, linear and differential probabilities, bit independence criteria, and balance.

The organisation of this paper is as follows: Section 2 covers the background of AES static S-boxes; Section 3 provides information about related work; Section 4 introduces the design principles and proposed methodology of the proposed dynamic S-box (QuantumGS-box); The test results and analysis are presented in Section 5, while Section 6 concludes the paper.

## 2. Background

An AES static S-box [34,35] having the order $(p \times q)$ is a mapping function $L = S(c)$, where $S = \{0,1\}^p \rightarrow \{0,1\}^q$ which is used to map p-bits input string $c$ to q-bits output string L. The input is mapped with its multiplicative inverse in $GF(2)^8$.

I. The first step is a nonlinear function $f(c)$ defined as:

$$f(c) = \begin{cases} 0, if\ c = 0 \\ c^{-1}, if\ c \neq 0 \end{cases} \quad (1)$$

The function $f(c)$ maps zero to zero, and for a non-zero field element c, it maps the element to its multiplicative inverse $c^{-1}$ in $GF(2)^8$

II. The multiplicative inverse is then transformed using the following affine transformation

$$\mathrm{L} = \mathrm{Affine}_8(\mathrm{C}) + \mathrm{b} \quad (2)$$

where $\mathrm{Affine}_8$ is $8 \times 8$ matrix and b is a constant. Namely, for a field element C = (C7, C6, C5, C4, C3, C2, C1, C0), L = $\mathrm{Affine}_8$(C) + b with

$$\begin{bmatrix} L_7 \\ L_6 \\ L_5 \\ L_4 \\ L_3 \\ L_2 \\ L_1 \\ L_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Table 1 shows an AES static S-box. It is a $16 \times 16$ matrix of hexadecimal values that are used in KSA and encryption. There are several cryptographic properties that make an AES S-box unique, such as high nonlinearity, correlation immunity, algebraic immunity, and no fixed points or opposite fixed points. However, because of its static nature, it is susceptible to enhanced cryptoanalysis techniques [36,37].

**Table 1.** Value matrix of an AES Static S-box.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 0 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | B | DB |
| A | E0 | 32 | 3A | A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 3 | F6 | E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | D | BF | E6 | 42 | 68 | 41 | 99 | 2D | F | B0 | 54 | BB | 16 |

The other part of this research study is the evolution computation with a *genetic algorithm*. A *genetic algorithm* (GA) is a well-known algorithm for search-oriented optimization inspired by biological evolution. A GA derives from the Darwinian theory of survival of the fittest. A GA dynamically changes the search process to reach an optimal solution based on crossover and mutation probabilities as a part of different phases of its cycle. Binary GA, as a search-oriented optimization, dynamically searches the solution within the generations

of the population encoded in the binary format. The fitness function is used to evaluate the population that undergoes a crossover or mutation to find the best value.

Our proposed work uses the transformation operations of a binary GA to provide efficient shuffling for the determination of the values of a dynamic S-box.

The initial population phase sets the foundation for the genetic algorithm by creating a diverse set of candidate solutions. The fitness function phase evaluates the quality of each solution, allowing the algorithm to prioritize the fittest individuals. The selection phase determines which individuals will be chosen as parents for reproduction, promoting the propagation of desirable traits. The crossover phase combines the genetic information of selected parents to create offspring with potentially improved characteristics. The mutation phase introduces small random changes to the offspring, ensuring the exploration of new solutions. Finally, the termination phase stops the algorithm once a satisfactory solution is found, or a predefined number of generations is reached.

## 3. Related Work

During the past two decades, cryptosystem designers have become increasingly concerned with designing key-dependent S-box methods with random properties. Various cryptoanalysis techniques, such as differential, linear, meet-in-the-middle, key recovery, and shortcut attacks, require dynamic methods and randomly generated substitution cryptosystems. As such, a strong emphasis has been placed on developing and implementing key-dependent S-box methods with random properties resistant to various cryptoanalysis techniques. Julia Juremi et al. surveyed many dynamic S-boxes focusing on framing the technology for generating them and creating them dynamically to increase cryptographic strengths and resist different attacks significantly [38].

Various methodologies for constructing dynamic S-boxes have been proposed, including pseudo random number generators (PRNGs) and true random number generators (TRNGs) with specific considerations, Chaos-based, algebraic, and optimization techniques.

Mangal Deep Gupta et al. proposed an architecture for reconfigurable PRNG designed using Verilog HDL, synthesized on the Xilinx tool using the Virtex-5 (XC5VLX50T) and Zynq (XC7Z045) FPGA [39]. Utilizing these PRNGs, they design two $16 \times 16$ substitution boxes (S boxes). The proposed S-boxes fulfil the cryptographic criteria such as Bijective, Balanced, Nonlinearity, Dynamic Distance, Strict Avalanche Criterion (SAC), and BIC nonlinearity criterion. They enhanced the security for image encryption by using these two S-boxes in terms of various parameters of image encryption like key space, information entropy, number of pixels change rate (NPCR), unified average changing intensity (UACI), and image encryption quantitatively in terms of (mean-squared error (MSE), structural similarity index (SSIM) and peak signal-to-noise ratio (PSNR)). Mengdi Zhao et al. a non-degenerate 2D enhanced quadratic map (2D-EQM) that exhibits ergodicity and randomness. In addition, they used it to generate affine transformation matrices and constants for seeding S-boxes based on affine transformation matrices [40]. They generated a number of keyed strong S-boxes with high nonlinearity.

Researchers have also used irreducible polynomials, additive constants, and static lookup tables [41] to make a dynamic S-box. Alamsyah et al. proposed a combination of an irreducible polynomial (irreducible polynomials) and an affine matrix (a Boolean operation XOR for each affine matrix element of the AES S-box) [42]. Several high-quality S-boxes with corresponding cryptographic properties are built from the combination of the results of these modifications. Using dynamic S-boxes and shift rows to simplify the encryption process also increased the encryption strength of the AES cipher [43]. Manjula G. et al. proposed that the modified S-boxes are dynamic, random, and key-dependent, adding to the algorithm's complexity and making cracking them more challenging [44]. Praveen et al. proposed a dynamic key-dependent S-box that relies on affine transformations with irreducible polynomials and affine constants [45]. Grasha Jacob et al. developed an S-box that can be generated dynamically based on the sub-byte transformation used in cryptography [46].

Chaos-based dynamic S-boxes are also of interest to many researchers. Alaa F. Kadhim et al. used shifting, chaotic theory (1D, 2D logistic maps), and particle swarm algorithms to generate a dynamic S-box based on the input key [17]. The system provides enhanced cryptographic properties. The authors of another study proposed an uncorrelated S-box element for generating an S-box that meets cryptographic criteria like bijection, nonlinearity, strict avalanche, output bit independence, equiprobable input/output XOR distribution, and maximum expected linear probability computed on typical chaos-based schemes without taking into account the lag time of a chaotic series [47]. Muhammad et al. report a new chaos-based affine transformation generation method that uses rotational matrices to generate key-based S-boxes [48].

Researchers also focus on genetic algorithms for creating dynamic S-boxes that lead to secure symmetric cryptosystems. Aguirre et al. proposed high nonlinearity by using the multi-objective evolutionary approach to evolve Boolean functions [49]. Stjepan Picek et al. experimented with a genetic algorithm and programming by modifying the mutation operator and initialization process to search Boolean functions with cryptographic properties [50]. Anand Kumar et al. proposed a robust S-box design with a hybrid approach of GA and Particle Swarm Optimization (PSO) [51]. Their performance evaluation is better regarding nonlinearity, strict avalanche effects, bits distribution, and bijectivity. By employing a GA in SP boxes and implementing a nonlinear neural network in the SP network, Kalaisel et al. proposed a redesigned, enhanced AES cryptosystem that increases security against timing attacks and reduces computation time. In order to reduce the probability of the algorithm being impervious to future dialects, Alaa F. Haitham et al. used the chaotic function to generate an initial random sequence of bits and the quantum crossover to provide a new and improved substitution box with increased nonlinearity [52].

Researchers focused on creating dynamic S-boxes; however, generating dynamicity using quantum randomness and a user key and creating a dynamic S-box while retaining the box's cryptographic properties is still an unexplored of research.

Our proposed substitution method contributes to existing knowledge by enhancing security of symmetric cryptosystems in HSN by generating the values of S-box at the time of execution from the secret key and (QRNG).

## 4. QuantumGS-Box—High-Speed Cloud-Based Storage Encryption with GA-Based Key Optimisation

The proposed method of generating a dynamic S-box includes a QRNG and bit shuffling based on the operations of a genetic algorithm to make the mathematical calculation simple while at the same time ensuring that the values generated by this method are highly secure. The proposed QuantumGS-box values are generated dynamically with the user defined key and quantum random number generated bits. The design principle is discussed first, followed by the proposed methodology.

### *4.1. The Design Principle*

The design principle of the proposed work uses algebraic techniques, quantum randomness, and optimization techniques to generate a dynamic, highly secure S-box. There are a variety of cryptographic properties associated with a highly secure S-box. These properties include nonlinearity, balance, correlation immunity, algebraic degree, and differential and linear approximation probabilities. A diagram illustrating the design principles is shown in Figure 4.

The *random number generators* (RNGs)-based S-box effectively affects the randomness of bits in the S-box. Pseudo-random number generator (PRNG) [39] and true random number generator (TRNG) [13] based S-boxes are proposed by some researchers in order to enhance the cryptographic properties. However, QRNG [53] ensures high entropy using quantum states of light. Quantis [22], a QRNG developed by IDQ, generates random numbers using a quantum process [54,55]. The device allows live verification of its operation and provides the highest level of entropy without requiring a post-processing function

to increase its entropy rate. QRNGs [23] are considered superior to traditional random number generators, as their source of randomness is invulnerable to environmental perturbations such as temperature, voltage, or current and are provably secure random number generators [28,56].

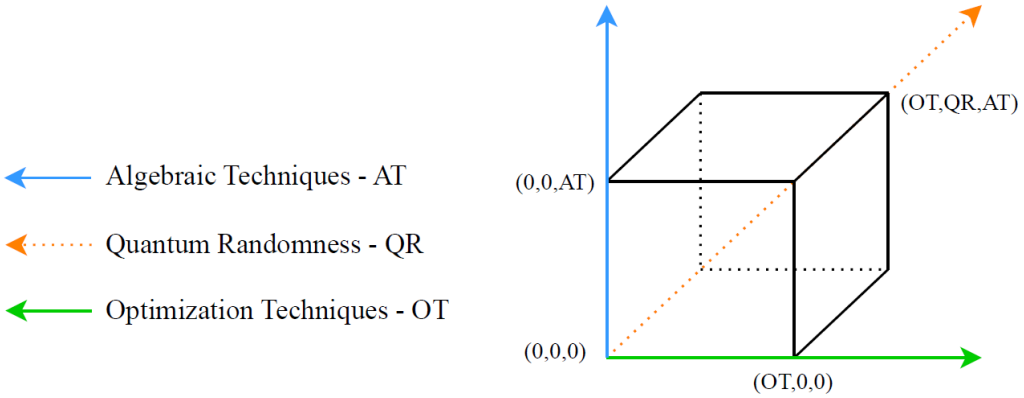



**Figure 4.** Design principle of the proposed work.

Algebraic methods are used to achieve various cryptographic properties in generating the S-box. However, with the advancement of cryptanalysis [57,58] over the last few years, these methods are becoming more vulnerable to exploitation.

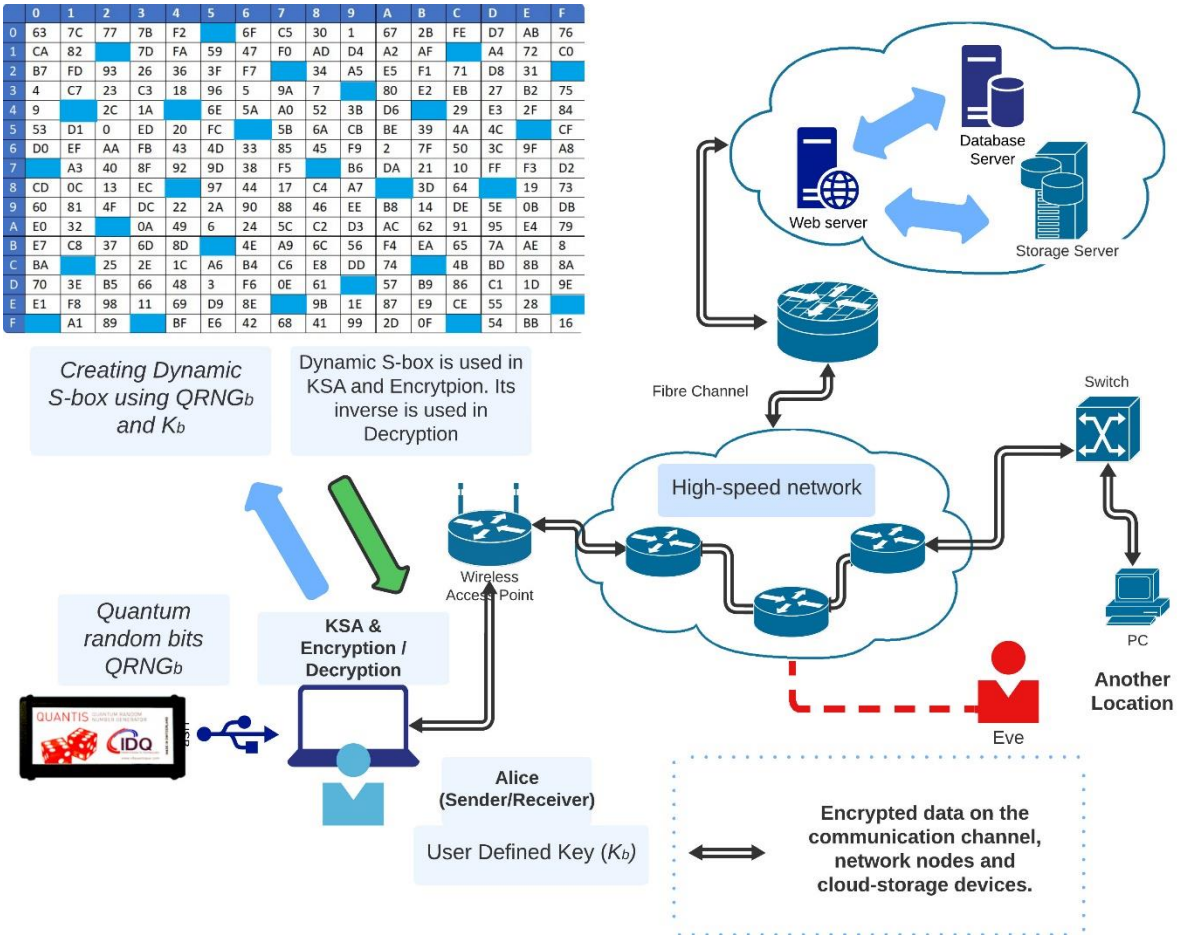
Another important criterion in the design of S-box dynamic bits is the speed at which the bits are generated. There needs to be an increase in the speed at which the S-box is constructed to make it more difficult to access the bits in the S-box. Dynamic S-boxes comprise the basis for generating the key and encrypting the data using that key. The higher speed of generating dynamic S-box values also contributes to the high-speed encryption of the data. Increasing the generation speed of dynamic S-box values reduces encryption time and data travel time on communication channels and data storage facilities.

The last criterion is the optimal values based on all three above-mentioned criteria, which can be achieved through evolutionary computation (EC). There are various studies [59–61] by researchers who created dynamic S-boxes based on EC and optimized the most effective solution for creating S-box values.

### *4.2. Proposed Methodology*

Figure 5 shows the dynamic proposed S-box, called QuantumGS-box, for KSA and for a cloud-based storage encryption. The figure shows the encryption with the new QuantumGS-box. The QuantumGS-box is a $16 \times 16$ matrix consisting of dynamic bits. The user will generate the key with KSA along with S-box and encrypt the information. Alice is both sender and receiver in this scenario, who encrypts the data before sending it to the high-speed network and decrypt it once it receives back.

The QuantumGS-box provides a matrix of substitution values based on a QRNG and a user defined key for KSA and encrypting the data. Alice will send the encrypted information over a communication channel to the cloud. The encrypted format makes the data secure before traveling through various HSN nodes. This ensures that the data is protected from Eve, a potential malicious attacker, who could attempt to intercept the data while in transit. The encrypted data is stored on multiple servers depending on their cloud application. The same encrypted data travels back over the network when Alice wants to access the data. Overall, this process ensures that Alice's data remains secure and confidential during data transmission to the cloud.




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | | 34 | A5 | E5 | F1 | 71 | D8 | 31 | |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | | 2C | 1A | | 6E | 5A | A0 | 52 | 3B | D6 | | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 0 | ED | 20 | FC | | 5B | 6A | CB | BE | 39 | 4A | 4C | | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| 7 | | A3 | 40 | 8F | 92 | 9D | 38 | F5 | | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | | 97 | 44 | 17 | C4 | A7 | | 3D | 64 | | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | | 0A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 3 | F6 | 0E | 61 | | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | | 9B | 1E | 87 | E9 | CE | 55 | 28 | |
| F | | A1 | 89 | | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | | 54 | BB | 16 |

**Figure 5.** Proposed QuantumGS-box in cloud-based storage encryption through high-speed network.

#### 4.2.1. Step by Step Procedure and Algorithm to Generate QuantumGS-Box

This section illustrates the mathematical calculation and the bit shuffling based on the operations of a *genetic algorithm* (GA) of the proposed method to generate dynamic S-box values. The step-by-step operation of the proposed method is shown with a control-flow diagram in Figure 6.

The flowchart starts with taking a user key from the user and then random bits from QRNG. The genetic algorithm will be processed with initial population, the generation and generating the chromosomes. A fitness function is used to select the best chromosomes in order to find the index to generate the values of QuantumGS-box.

***Step1 (S1):*** Initially, the user inputs a 16-character key, denoted as $UKey$
***Step2 (S2):*** The QRNG generates the 128 quantum random bits, denoted as $QRNG(128)$
***Step3 (S3):*** The user key UKey is converted into bits and denoted as $Key_b$
***Step4 (S4):*** Initial Population generation ($P_{init}$)

- $QRNG(128)$—Quantum random bits from QRNG (128 bits)
- $Key_b$ —User defined key (128 bits)

$$P = P \cup (QRNG(128) \oplus Key_b) \tag{3}$$

***Step5 (S5):*** This step calls the genetic algorithm with the function name optimize_GA and a parameter initial population ($P_{init}$), calculated at S4. Algorithm 1 shows the pseudo-code of the proposed algorithm for optimize_GA. Section 4.2.2 focuses on the optimize_GA.

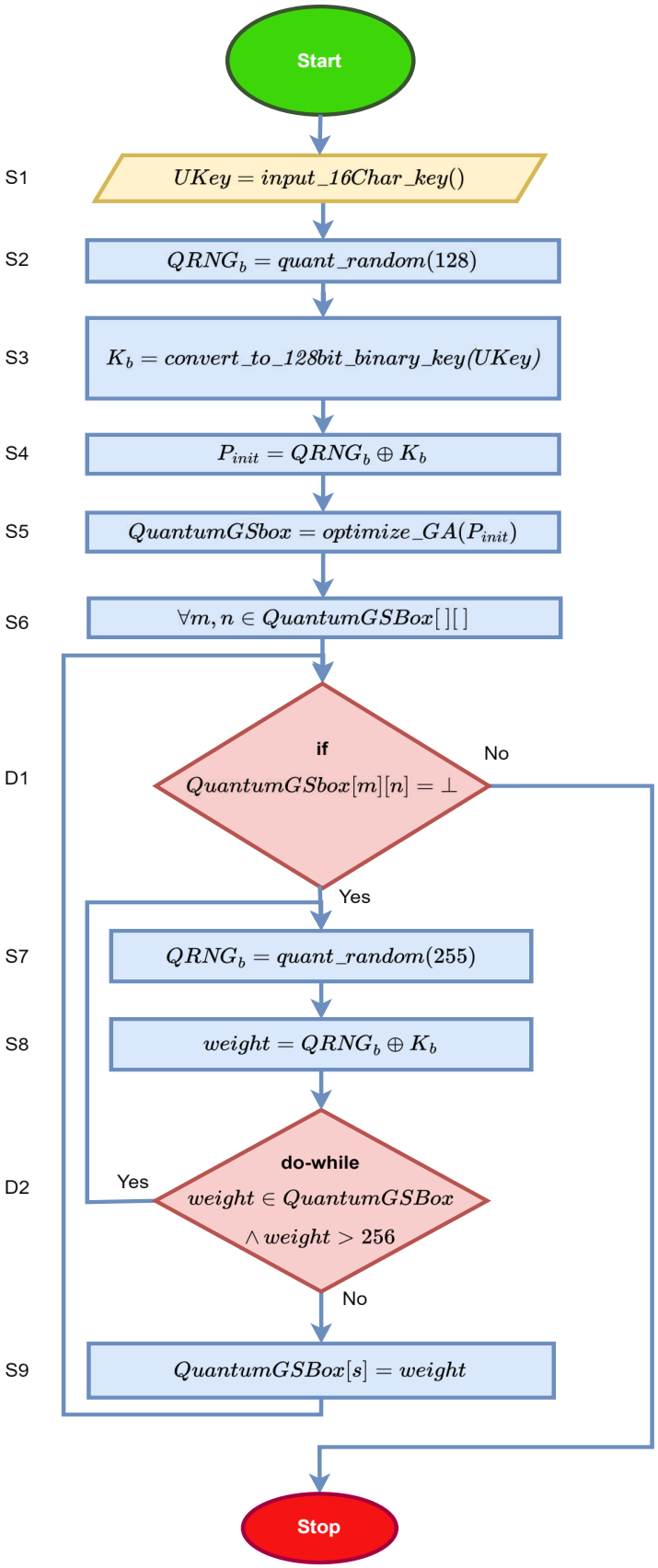



**Figure 6.** Flowchart of the proposed QuantumGS-box.

***Step 6 (S6):*** This step checks each element in the $QGSbox$ with row m and column n. The decision depends on the positions that have NaN ($\perp$) as their value.

The following steps will calculate the weight of that positions, where there is a NaN (Not a Number) in the $QGSbox$. If there is no NaN, then this will proceed to stop.

***Step 7 (S7) to Step 9 (S9):*** These steps will continue till $QGSbox$ has NaN ($\perp$).

The quantum random number bits will be generated from QRNG and as $QRNG(255)$

The value of temporary weight denoted as *weight* is calculated by using a XOR function on the quantum random bits and user-defined key bits.

$$weight = QRNG(255) \oplus Key_b \tag{4}$$

The next block will check if the value for *weight* has already been stored in $QGSbox$, and greater than 256. If it is, the value will be regenerated; if it does not, the value will be stored in the $QGSbox$.

When the algorithm reaches the stop, then all $QGSbox$ values have been calculated and the algorithm terminates. Algorithm 2 shows the pseudo-code of the proposed algorithm.

**Algorithm 1:** Algorithm $optimize\_GA(P_{init})$: Calculate positions for weights of QuantumGS-box

**INPUT:** $P_{init}$ *//initial population (with PSIZEchromosomes)*

$SSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$ *//static S-Box*

$RMAX$*//max number of S-Box weights to replace:* $10 \leq RMAX \leq 250$

**OUTPUT:** $QGSbox_{tmp}$*//interim QuantumGS-box*

***begin:***
  $P_{tmp} = P_{init}$ *//initialize current population with initial population*
  $QGSbox_{tmp} = SSbox$ *//initialize output QuantumGS-box*
  ***for*** *gen_cnt* ***is*** 1 ***to*** $RMAX$ *//iterate RMAXtimes*
    $C_{fittest} = \underset{C \in P_{tmp}}{\operatorname{argmax}} fitness\_GA(C)$*//chromosome with max fitness value*
    $i = C_{fittest}[60], \quad j = C_{fittest}[61], \quad k = C_{fittest}[62], \quad l = C_{fittest}[63]$
    $p = C_{fittest}[124], \quad q = C_{fittest}[125], \quad r = C_{fittest}[126], \quad s = C_{fittest}[127]$
    $x = bin\_to\_hex(i, j, k, l)$ // $i \cdot 2^3 + j \cdot 2^2 + k \cdot 2^1 + l \cdot 2^0$
    $y = bin\_to\_hex(p, q, r, s)$ // $p \cdot 2^3 + q \cdot 2^2 + r \cdot 2^1 + s \cdot 2^0$
    $QGSbox_{tmp}[x][y] = \perp$
    $QGSbox_{tmp}[y][x] = \perp$
    $P_{tmp} = mutate(crossover(P_{tmp}))$*//population of next generation*
  *return* $QuantumGSbox_{tmp}$
***end***

**Algorithm 2:** Algorithm to generate QuantumGS-box

**INPUT:** $UKey$ *//16-character textual key provided by user*

**OUTPUT:** $QGSbox = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0F} \\ a_{01} & a_{11} & \cdots & a_{1F} \\ \vdots & \vdots & \ddots & \vdots \\ a_{F0} & a_{F1} & \cdots & a_{FF} \end{pmatrix}$ *//QuantumGS-box*

***begin:***
  $K_b$ = *convert_to_128bit_binary_key(UKey)*
  $P_{init} = \varnothing$ *//initial population*
  $\forall i \in \{1 \ldots PSIZE\}.\ P = P \cup (QRNG(128) \oplus Key_b)$ *//fill initial population*

```
5       QGSbox = optimize_GA ( P_init)
6       ∀m, n ∈ QGSBox[ ][ ]
7       If QGSbox[m][n] = ⊥
8           do
9               weight = QRNG(255) ⊕ Key_b
10          while weight ∈ QGSBox ∧ weight > 256
11          QGSbox[s] = weight
12  end
```

Note: ⊥—Not a Number (NaN).

#### 4.2.2. Algorithm $optimize\_GA(P_{init})$

Algorithm 1 shows the steps to calculate the positions for weights of $QGSbox$.

This step will continue to execute from generation 1 to $RMAX$. RMAX is used to define the upper limit of the generation of the genetic algorithm.

In a population of chromosomes, $fitness\_GA(C)$ computes the fitness of each chromosome. The fitness value that is highest across all chromosomes will be considered the most appropriate and its chromosome is denoted as $C_{fittest}$. Algorithms 3 and 4 focuses on $fitness\_GA(C)$ and how to calculate each fitness value.

This highest value chromosome $C_{fittest}$ tells the positions for the weights of $QGSbox$. We have chosen four middle bits and four most significant bits to calculate the row and column index. The index positions $a_1$ and $a_2$ represent row and column positions. For example, the 60, 61, 62 and 63 bits of $C_{fittest}$ are 0110 and 124, 125, 127 and 128 bits are 0100, so the $a_1 = 0110$ in binary and $a_1 = 6$ in hexadecimal. Similarly, $a_2 = 0100$ and $a_2 = 4$.

These positions in $QGSbox$ as $QGSbox$ [4][6] and $QGSbox$ [6][4] will now have NaN.

The next step will calculate the next population $\left(P_{tmp}\right)$ using crossover and mutation for the next iteration of the generation.

Crossover: crossover$\left(\mathrm{P_{tmp}}\right)$

We are using the crossover operation of a GA [62]. The chromosomes of each generation are combined with via the crossover function to produce the chromosomes of the next generation. A one-point crossover mechanism has been used. In order to create a new offspring, a crossover point is chosen at random and the tails of $(C_1\ and\ C_2)$ are swapped to create a new offspring. The crossover chromosome is calculated as follows:

$$crossover\left(P_{tmp}\right) = f\left(P_{tmp}\right) \tag{5}$$

where:

$crossover\left(P_{tmp}\right)$—Population chromosome after crossover

$n$—a random number crossover point derived from the QRNG

Algorithm 3 shows the $crossover\left(P_{tmp}\right)$ function to calculate the crossover for the current population. Algorithm 4 crossover$_C(C_1, C_2)$ shows the 1-point crossover with a quantum random number.

**Algorithm 3:** Algorithm $crossover\left(P_{tmp}\right)$ : Calculate crossover for the current population

**INPUT:** $P_{tmp}$ *//initial population (with PSIZEchromosomes)*

**OUTPUT:** $P_{new}$ *//interim population for crossover*

```
1  begin:
2      P_new = ∅    //initialize new population
3      while(P_tmp ≠ ∅)
4          C1, C2 = remove(P_tmp)    //Two chromosomes from current population
5          P_new = P_new ∪ crossover_C(C1, C2)
6      return P_new
7  end
```

**Algorithm 4:** Algorithm $crossover_C(C_1, C_2)$ : Calculate 1-point crossover.

**INPUT:** $C_1$, $C_2$ //*two chromosomes from the current population*

**OUTPUT:** $C_1'$ , $C_2'$ //*new crossover chromosomes*

```
1 begin:
2     p = QRNG(0...127)//a quantum random number from QRNG within the range
3     C1' = C1[0...(p − 1)] + C2[(p)...127]
4     C2' = C2[0...(p − 1)] + C1[(p)...127]
5     return (C1' , C2')
6 end
```

Mutation: mutate$\left(P_{tmp}\right)$

We are using the mutation operation of a GA [62]. In the GA, mutations can be used to maintain diversity among the chromosomes in a population. A mutation has not been incorporated into the population as a whole, it is measured by a mutation probability that is assigned to each individual according to their fitness value. A random number is generated between 0 and 1 and the mutation probability ($k$) of our proposed work is 0.95 (see (5)). If the random number is greater than the ($k$) probability, the values will be generated by QRNG.

The mutation chromosome is calculated as follows:

$$mutate\_c(C) = \begin{cases} QRNG(128), Q_{rn}(0,1) > k \mid \ k = 0.95 \\ C, otherwise \end{cases} \tag{6}$$

where:

$mutate\_c(C)$—Population chromosome after mutation

C—Chromosome of initial population $(P_{init})$

$Q_{rn}(n_1, n_2)$—a random number from QRNG within the range $n_1 \leq Q_{rn}(n_1, n_2) \leq n_2$

The process will proceed to RMAX generation and the population for the next iteration will be as follows:

$$P_{tmp} = mutate\left(crossover\left(P_{tmp}\right)\right) \tag{7}$$

Algorithm 5 shows the $mutate\_c(C)$ function to calculate the mutation for the current population.

**Algorithm 5:** Algorithm $mutate\_c\left(P_{tmp}\right)$ : Calculate mutation for the current population.

**INPUT:** $P_{tmp}$ //*initial population (with PSIZEchromosomes)*

**OUTPUT:** $P_{new}$//*interim population for mutation*

```
1 begin:
2     P_new = ∅    //initialize new population
3     while(P_tmp ≠ ∅)
4         C = remove(P_tmp)    //chromosome from current population
5         P_new = P_new ∪ mutate_C(C)
6     return P_new
7 end
```

4.2.3. Algorithm $fitness\_GA(C)$

Algorithm 6 shows pseudo-code of the proposed algorithm for $fitness\_GA(C)$ and the calculation of each of the chromosomes fitness value.

**Algorithm 6:** Algorithm ***fitness_GA*(*C*)**: Calculate fitness value for a given chromosome (128-bit sequence).

**INPUT:** $C$ *//chromosome (128-bit sequence)*

**OUTPUT:** $Fitness$ *//fitness value*

```
begin:
    R = C[0...63]        //least significant 64 bits of C
    L = C[64...127]      //most significant 64 bits of C
    H_dist = Hamming_distance(R , L)
    JW_dist = Jaro_Winkler_distance (R , L)
    L_dist = Levenshtein_distance (R , L)
    Fitness = H_dist + JW_dist + L_dist
    return Fitness
end
```

The current chromosome splits into two parts.
*Chromosome:* (Split the $C$ into two halves $R$ and $L$)

$$R = C[0 \ldots 63] \tag{8}$$

$$L = C[64 \ldots 127] \tag{9}$$

Different distance values will be calculated using these two parts, leading to the fitness value, $Fitness$. The highest fitness value is the optimum to optimize chromosomes.

#### 4.2.4. Calculation of Distance Values: $H_{dist}$, $\mathrm{JW_{dist}}$, $\mathrm{L_{dist}}$

$Fitness$ calculates the fitness function for the QuantumGS-box as follows:

$$Fitness = (H_{dist} + JW_{dist} + L_{dist}) \tag{10}$$

where:
$H_{dist}$—Hamming Distance between $R$ and $L$: $H_{dist}(R, L)$
$JW_{dist}$—Jaro-Winkler Distance between $R$ and $L$: $JW_{dist}(R, L)$
$L_{dist}$ —Levenshtein Distance between $R$ and $L$: $L_{dist}(R, L)$

Hamming Distance ($\mathrm{H_{dist}}$)

$\mathrm{H_{dist}}(BS_1\ ,\ BS_2)$, the hamming distance [63] between two bit sequences $BS_1$ and $BS_2$ is calculated as the number of bit positions $BS_{1,j}$ in $BS_1$ that are different to those $BS_{2,j}$ in $BS_2$, divided by the chromosome length:

$$H_{dist}(BS_1, BS_2) = \frac{\left|\left\{BS_{1,j} \;\middle|\; BS_{1,j} \neq BS_{2,j} \wedge (0 \leq j < len(BS_1))\right\}\right|}{len(BS_1)} \tag{11}$$

Jaro-Winkler Distance ($\mathrm{JW_{dist}}$)

The first step in calculating $\mathrm{JW_{dist}}(BS_1\ ,\ BS_2)$, the Jaro-Winkler Distance between $BS_1$ and $BS_2$, is to obtain the Jaro similarity $\mathrm{JS}(BS_1\ ,\ BS_2)$, score between $BS_1$ and $BS_2$.

The $JS(BS_1, BS_2)$ score is 0 if the strings do not match at all, and 1 if they are an exact match. $JS$ is calculated as follows:

$$JS(BS_1, BS_2) = \begin{cases} 0, if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|BS_1|} + \frac{m}{|BS_2|} + \frac{m-t}{m}\right), otherwise \end{cases} \tag{12}$$

where:
$|BS_i|$—is the length of the bitstring $BS_i$

$m$—the number of matching bits

$t$—the number of transpositions

The Jaro–Winkler similarity $(JWS)$ uses a con for a more specific similarity with a defined length $len$ and is calculated as follows:

$$JWS(BS_1, BS_2) = JS(BS_1, BS_2) + len \times con(1 - JS(BS_1, BS_2)) \quad (13)$$

where:

$JS$ is the Jaro similarity between bitstring; $BS_1$ and $BS_2$

$len$—the length of common prefix at the start of the string with up to a maximum of 4 characters

The final $JW_{dist}$ is

$$JW_{dist}(BS_1, BS_2) = 1 - JWS(BS_1, BS_2) \quad (14)$$

Levenshtein Distance $(L_{dist})$

The Levenshtein distance $L_{dist}$ $(BS_1, BS_2)$ between two bitstring $BS_1$ and $BS_2$ is calculated as follows:

$$L_{dist}(BS_1, BS_2) = \begin{cases} |BS_1| & if\ |BS_2| = 0, \\ |BS_2| & if\ |BS_1| = 0, \\ L_{dist}(tail(BS_1),\ tail(BS_2)) & if\ BS_1[0] = BS_2[0], \\ 1 + min \begin{cases} L_{dist}(tail(BS_1), BS_2) \\ L_{dist}(BS_1,\ tail(BS_2) \\ L_{dist}(tail(BS_1),\ tail(BS_2)) \end{cases} & Otherwise \end{cases} \quad (15)$$

The proposed algorithm generates different QuantumGS-boxes based on different generations and keys. Table 2 shows the QuantumGS-box using the 75th generation of the GA and the user defined key—"The QuantumGS-box".

**Table 2.** QuantumGS-box generated using the 75-th generation of the GA.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 25 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | F6 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | D9 | 8D | 31 | 15 |
| 3 | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | 72 | 29 | E3 | 2F | 8A |
| 5 | 53 | D1 | 1D | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | B3 | 9F | 42 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | B | DB |
| A | E0 | 32 | 3A | A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 84 | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| C | BA | 78 | F8 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 3C |
| D | 70 | 3E | B5 | 66 | 48 | 3 | D8 | E | 61 | 35 | 57 | B9 | 86 | C1 | B0 | 9E |
| E | E1 | BF | 98 | 11 | 69 | 76 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 8C | 28 | DF |
| F | A8 | A1 | 89 | D | 71 | E6 | 55 | 68 | 41 | 99 | 2D | F | 0 | 54 | BB | 16 |

## 5. Result Analysis

Results are drawn from the strength of S-box parameters such as nonlinearity, bit-independence, avalanche properties including differential and linear approximation analysis. There are several tools to evaluate the performance analysis of cryptographic properties of S-box. The most common tools are MATLAB, Sage [64], SET [65] and the S-BOX performance analysis [66] tool. We evaluated our results using SET and the S-BOX performance tool.

*SET-UP experiments and Tools*

Hardware: Quantis [29]—A USB-based Quantum random number generators developed by IDQ Its general specifications include—Random bit rate 1:4 Mbit/s ± 10% (Quantis-USB-4M), Thermal noise contribution: <1% (Fraction of random bits arising from thermal noise), Storage temperature: −25 to + 85 °C, USB specification 2.0 and Power Via USB port

Computer—Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM: 8.00 GB, System type: 64-bit operating system, x64-based processor

Software: Java—Netbeans with JDK 1.8.0

S-box Testing Tools: SET [65] and S-BOX performance analysis [66]

### 5.1. Bijectivity Property

An S-box of n × n size is said to be bijective if it has all possible output values from interval [0,$2^{n-1}$]. If $f_i(1 \leq i \leq n)$ is a Boolean function of an S-box [67], it satisfies

$$\mathrm{wt}\left(\sum_{i=1}^{n} a_i f_i\right) = 2^{n-1} \tag{16}$$

where $a_i \in \{0,1\}, (a_1 a_2 \ldots a_n) \neq (0,0 \ldots 0)$ and wt(x) is the hamming weight, which indicates the number of 1's in a given vector. $f_i$ is to be balanced between 0 and 1. The proposed QuantumGS-box generates the n × n matrix of unique values, as shown in Table 2.

### 5.2. Non-Linearity

An S-box is strong if it uses a Boolean function with a high non-linearity value. The non-linearity [68] $N_f$ of a Boolean function f(x) is

$$N_f = 2^{n-1}\left(1 - 2^{-n}\max\left|WS_f(\omega)\right|\right) \tag{17}$$

where $WS_f(\omega)$, the Walsh spectrum of function $f$ is defined as

$$WS_f(\omega) = \sum_{\omega \in GF(2^n)}^{n} (-1)^{f(x)+x\cdot\omega} \tag{18}$$

where, $x \cdot \omega$ represents the dot-product of x and $\omega$

Table 3 shows the nonlinearity of different S-boxes generated by different generations. The speed illustrates the time taken to generate the dynamic S-box bits based on the related generations.

**Table 3.** Nonlinearity of the QuantumGS-box with three different generation and their generation time.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Generation 50, Time: 3.238 s** | | | | | | | | |
| Nonlinearity | 108 | 110 | 112 | 112 | 112 | 110 | 110 | 110 |
| **Generation 75, Time: 4.560 s** | | | | | | | | |
| Nonlinearity | 106 | 108 | 108 | 110 | 108 | 110 | 110 | 110 |
| **Generation 100, Time: 4.874 s** | | | | | | | | |
| Nonlinearity | 112 | 108 | 108 | 108 | 108 | 108 | 108 | 110 |

Table 4 presents a comprehensive comparison of our proposed S-box security analysis of nonlinearity. The sample of QuantumGS-box (75th generation), generated by the proposed method is compared with relevant dynamic S-boxes published in the last few years in different category. In the proposed work, we achieve a 110 as the maximum value of nonlinearity with an average of 108.75. Figure 7 shows the graphical representation of nonlinearity comparisons.

**Table 4.** Comparison of QuantumGS-box with different research studies in terms of nonlinearity.

| Design Methodology | Ref. | Non-Linearity | | |
|---|---|---|---|---|
| | | Min | Max | Avg |
| Algebraic | [69] | 106 | 108 | 107 |
| | [70] | 106 | 108 | 107.25 |
| | [71] | 104 | 108 | 106.8 |
| | [72] | 102 | 111 | 106.5 |
| | [8] | 106 | 110 | 107.75 |
| | [73] | 106 | 108 | 106.5 |
| Chaos Based Design | [74] | 108 | 110 | 109.5 |
| | [75] | 104 | 110 | 106.3 |
| | [76] | 108 | 112 | 109.25 |
| | [77] | 100 | 106 | 104 |
| | [78] | 100 | 110 | 103.8 |
| | [79] | 100 | 108 | 105 |
| Others | [13] | 99 | 108 | 103.5 |
| | [80] | 102 | 110 | 106.5 |
| | [59] | 102 | 110 | 107 |
| QuantumGS-box 75 Gen | this work | 106 | 110 | 108.75 |

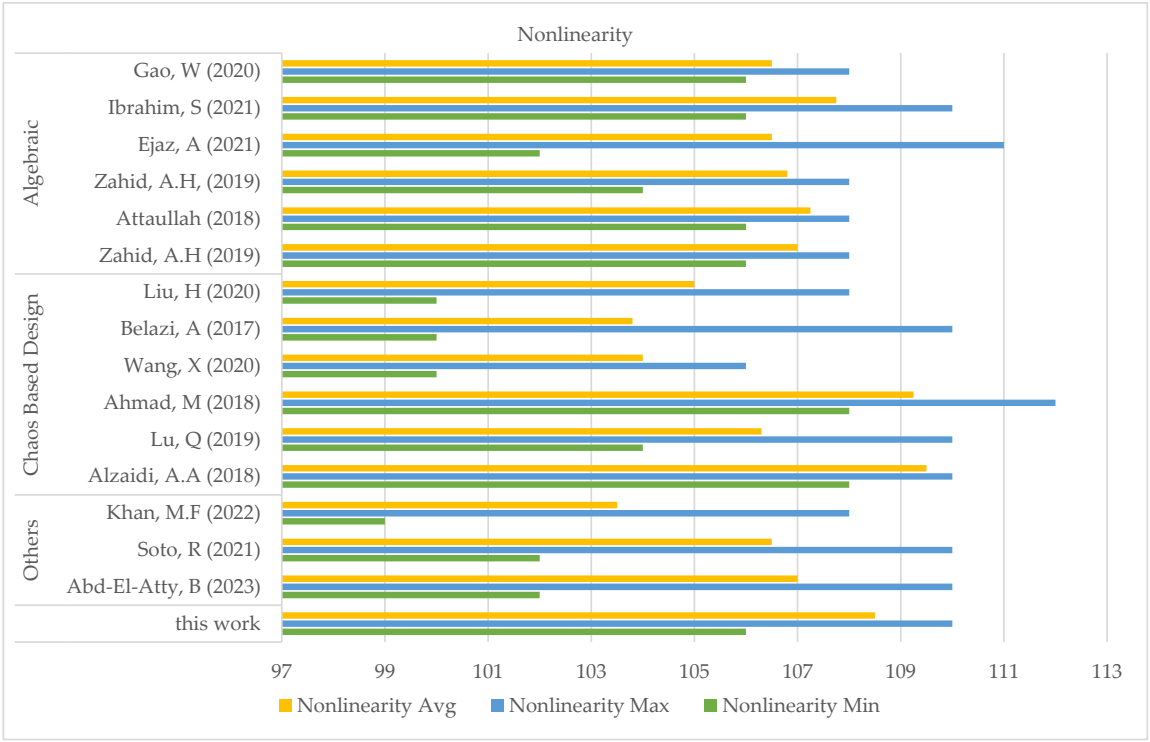



**Figure 7.** Comparison of QuantumGS-box(this work) with different research studies in terms of nonlinearity. Zahid, A.H (2019)—[69]; Attaullah (2018)—[70]; Zahid, A.H, (2019)—[71]; Ejaz, A (2021)—[72]; Ibrahim, S (2021)—[8]; Gao, W (2020)—[73]; Al-zaidi, A.A (2018)—[74]; Lu, Q (2019)—[75]; Ahmad, M (2018)—[76]; Wang, X (2020)—[77]; Belazi, A (2017)—[78]; Liu, H (2020)—[79]; Khan, M.F (2022)—[13]; Soto, R (2021)—[80]; Abd-El-Atty, B (2023)—[59].

### 5.3. Bit-Independence Criteria—Strict Avalanche Criteria (BIC-SAC)

*Bit-Independence Criteria* (BIC) tests measure the correlation between each pair of output bits when one input bit changes in response to the change in the output bit. In this test, the diagonal values are excluded from the output matrix of n × n dimensions.

If a Boolean function satisfies the *Strict Avalanche Criteria* (SAC) [46], half of the output bits should change when there is a change in a bit. Any change in the input vector will significantly change the output vector with a probability of ½. Table 5 shows the BIC-SAC values of the proposed QuantumGS-box. The maximum, minimum and average SAC values for the QuantumGS-box are equal to 0.525391, 0.46875, and 0.497070 correspondingly. Based on the results, the proposed QuantumGS-box fulfils the SAC property as the average value for the QuantumGS-box is equal to the desired value of SAC (0.5).

**Table 5.** BIC-SAC of QuantumGS-box.

| **0** | **0.511719** | **0.503906** | **0.525391** | **0.513672** | **0.486328** | **0.46875** | **0.509766** |
|---|---|---|---|---|---|---|---|
| 0.511719 | 0 | 0.511719 | 0.496094 | 0.498047 | 0.507813 | 0.507813 | 0.515625 |
| 0.503906 | 0.511719 | 0 | 0.527344 | 0.501953 | 0.507813 | 0.486328 | 0.505859 |
| 0.525391 | 0.496094 | 0.527344 | 0 | 0.519531 | 0.517578 | 0.503906 | 0.503906 |
| 0.513672 | 0.498047 | 0.501953 | 0.519531 | 0 | 0.519531 | 0.515625 | 0.5 |
| 0.486328 | 0.507813 | 0.507813 | 0.517578 | 0.519531 | 0 | 0.509766 | 0.488281 |
| 0.46875 | 0.507813 | 0.486328 | 0.503906 | 0.515625 | 0.509766 | 0 | 0.517578 |
| 0.509766 | 0.515625 | 0.505859 | 0.503906 | 0.5 | 0.488281 | 0.517578 | 0 |

### 5.4. Linear Approximation Probability (LAP)

The LAP assesses the security of S-boxes against linear cryptanalysis. An S-box provides diffusion and confusion of bits through linear mappings between inputs and outputs. The LAP of an event determines a maximum imbalance [48,65].

$$LAP = \max_{p_x, q_x \neq 0} \left| \frac{|\{ x \mid (x \in R) \wedge (x \cdot p_x = S(x) \cdot q_x) \}|}{2^n} - \frac{1}{2} \right| \tag{19}$$

where:

$p_x$ and $q_x$ are the input and output values respectively and $R = \{1, 2, 3, 4, \ldots, 255\}$

A small linear probability in an S-box makes the box very resistant to linear cryptanalysis. In our proposed work, we achieve a LAP value of 0.1015. Table 6 compares the maximum LP of different researchers with the proposed work. Figure 8 shows the graphical representation of the comparison data.

**Table 6.** Comparison of QuantumGS-box with different research studies in terms of LAP.

| **Design Methodology** | **Ref.** | **Linear Approximation Probability (LAP)** |
|---|---|---|
| Algebraic | [69] | 0.1560 |
| | [70] | 0.1094 |
| | [71] | 0.1400 |
| | [72] | 0.1090 |
| | [8] | 0.1172 |
| | [73] | 0.1250 |

**Table 6.** *Cont.*

| Design Methodology | Ref. | Linear Approximation Probability (LAP) |
|---|---|---|
| Chaos Based Design | [74] | 0.1328 |
| | [75] | 0.1250 |
| | [76] | 0.1250 |
| | [77] | 0.1328 |
| | [78] | 0.1250 |
| | [79] | 0.1250 |
| Others | [13] | 0.1406 |
| | [80] | 0.1484 |
| | [59] | 0.1172 |
| QuantumGS-box 75 Gen | this work | 0.1015 |

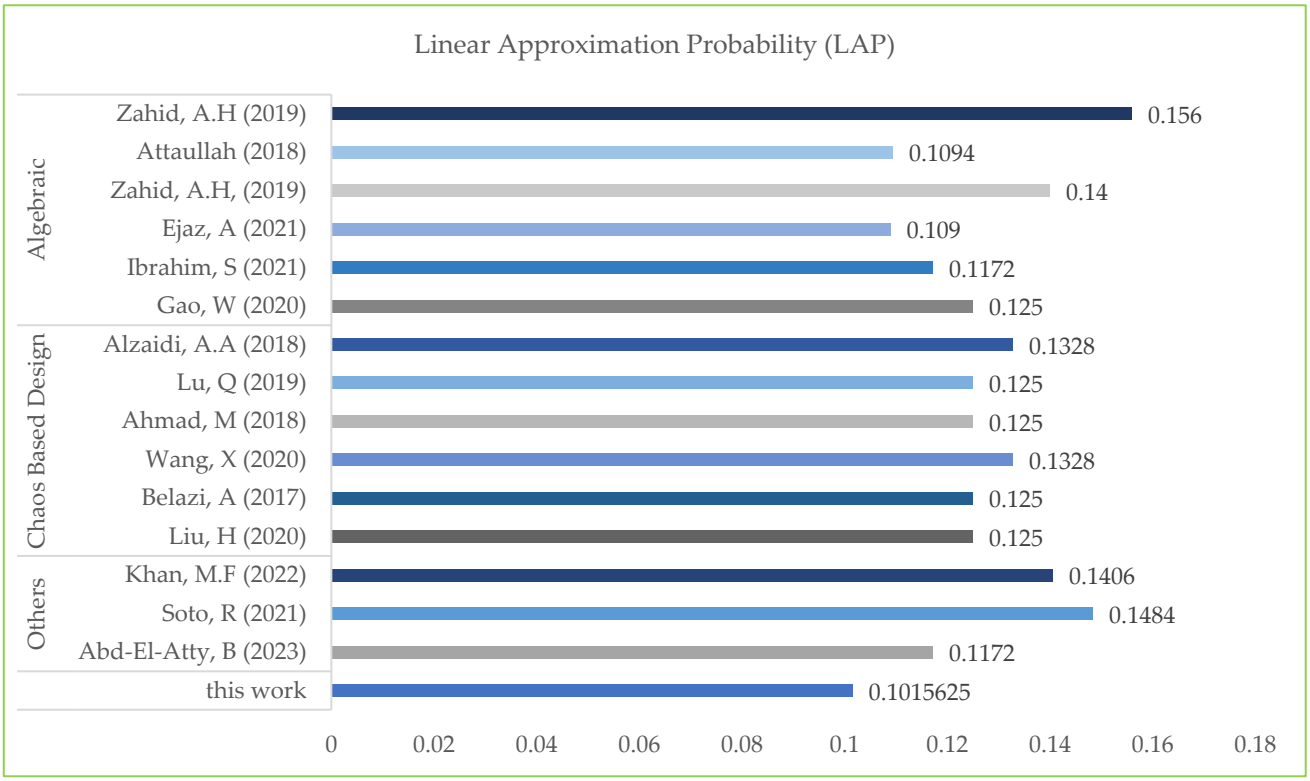



**Figure 8.** Comparison of the QuantumGS-box with different research studies in terms of LAP. Zahid, A.H (2019)—[69]; Attaullah (2018)—[70]; Zahid, A.H, (2019)—[71]; Ejaz, A (2021)—[72]; Ibrahim, S (2021)—[8]; Gao, W (2020)—[73]; Alzaidi, A.A (2018)—[74]; Lu, Q (2019)—[75]; Ahmad, M (2018)—[76]; Wang, X (2020)—[77]; Belazi, A (2017)—[78]; Liu, H (2020)—[79]; Khan, M.F (2022)—[13]; Soto, R (2021)—[80]; Abd-El-Atty, B (2023)—[59].

### *5.5. Differential Approximation Probability (DP)*

An effective way to assess S-box resistance to differential attacks is to use the differential approximation probability (DP) [48,81]. DP indicates the probability that a particular change in output bits will occur due to a change in input bits.

The *DP* is calculated as follows

$$DP(\Delta x \rightarrow \Delta y) = \left( \frac{|\{\, x \mid \ (x \in X) \ \wedge \ ((S(x) \oplus S(x \oplus \Delta x)) = \Delta y)\}|}{2^n} \right) \tag{20}$$

where

$x$ represents the set of all possible input values, 2n is a total number of all the elements in the S-box with a small differential value is strongly resistant to differential cryptoanalysis. In our proposed work, we achieve a DAP value of 0.03125.

### *5.6. Balanced Output*

An S-box with n input bits and m output bits, $m \leq n$, is balanced if each output occurs $2n-m$ times. For the S-box to be balanced [46], it should have the same number of 0's and 1's. The result from the SET [65] tool assessed that the QuantumGS-box is balanced.

## 6. Conclusions

Our research concludes that the quantum random number-based genetic algorithm can generate dynamic QuantamGS-boxes with different random S-box values at different arbitrary positions for cloud-based storage. The dynamic feature of the algorithm based on bit-shuffling with the operations of a genetic algorithm also makes it more resilient to modern cryptographic attacks. In our proposed work, we achieve the lowest LAP value when compared to other researches. QuantumGS-boxes generated by the proposed algorithm assures nonlinear. The QuantumGS-box also has low differential and linear approximation properties in addition to satisfying bijectivity cryptography properties, making it resistant to differential and linear attacks. Low differential and linear approximation properties makes guessing the algorithm's output from its inputs difficult and resistant to attacks that seek to exploit patterns in the input or output of the cryptographic system. Our proposed QuantumGS-box ensures that cloud-based storage over will be more resilient to various cryptographic attacks.

The future work of this research focuses on the proposed QuantumGSbox for the LoRaWAN IoT security symmetric algorithm and can be used to enhance the IoT security. LoRaWAN is a low-power wide-area network that connects devices with long-range communication over a low bit rate. This protocol is widely used in IoT communication. The encryption process of the LoRaWAN using the QuantumGS-box encrypts the user's data. The user sends the encrypted data over the high-speed network on IoT cloud storage and receives the same encrypted data.

**Author Contributions:** Methodology, A.S., A.T. and R.K.; Software, A.S.; Validation, R.K.; Data curation, A.S.; Writing—original draft, A.S., A.T. and R.K.; Writing—review & editing, A.S., A.T. and R.K.; Visualization, A.S. and R.K.; Supervision, A.T. and R.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research work received funding by University of Hertfordshire, United Kingdom.

**Data Availability Statement:** Data is contained within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zhang, X.; Li, W.; Hu, H.; Dutta, N.K. High-speed all-optical encryption and decryption based on two-photon absorption in semiconductor optical amplifiers. *J. Opt. Commun. Netw.* **2015**, *7*, 276–285. [CrossRef]
2. Stallings, W. *Cryptography and Network Security: Principles and Practices*, 6th ed.; Prentice Hall Press: Upper Saddle River, NJ, USA, 2013.
3. May, L.; Henricksen, M.; Millan, W.; Carter, G.; Dawson, E. Strengthening the key schedule of the AES. *Lect. Notes Comput. Sci.* **2002**, *2384*, 226–240. [CrossRef]
4. Hughes, R.; Nordholt, J. *Strengthening the Security Foundation of Cryptography with Whitewood's Quantum-Powered Entropy Engine*; Whitewood Encryption Systems, Inc.: Boston, MA, USA, 2016; Available online: http://www.whitewoodencryption.com (accessed on 3 August 2021).
5. Smart, N.P.; Rijmen, V.; Warinschi, B.; Watson, G. *Algorithms, Key Sizes and Parameters Report*; ENISA: Athens, Greece, 2014. Available online: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014 (accessed on 9 September 2021).
6. Siddiqui, N.; Khalid, H.; Murtaza, F.; Ehatisham-Ul-Haq, M.; Azam, M.A. A novel algebraic technique for design of computational substitution-boxes using action of matrices on galois field. *IEEE Access* **2020**, *8*, 197630–197643. [CrossRef]

7. Alharbi, A.R.; Jamal, S.S.; Khan, M.F.; Gondal, M.A.; Abbasi, A.A. Construction and Optimization of Dynamic S-Boxes Based on Gaussian Distribution. *IEEE Access* **2023**, *11*, 35818–35829. [CrossRef]
8. Ibrahim, S.; Abbas, A.M. Efficient key-dependent dynamic S-boxes based on permutated elliptic curves. *Inf. Sci.* **2021**, *558*, 246–264. [CrossRef]
9. Zahid, A.H.; Tawalbeh, L.; Ahmad, M.; Alkhayyat, A.; Hassan, M.T.; Manzoor, A.; Farhan, A.K. Efficient Dynamic S-Box Generation Using Linear Trigonometric Transformation for Security Applications. *IEEE Access* **2021**, *9*, 98460–98475. [CrossRef]
10. Lu, Q.; Zhu, C.; Deng, X. An Efficient Image Encryption Scheme Based on the LSS Chaotic Map and Single S-Box. *IEEE Access* **2020**, *8*, 25664–25678. [CrossRef]
11. Ibrahim, S.; Abbas, A.M.; Alharbi, A.A.; Albahar, M.A. A New 12-Bit Chaotic Image Encryption Scheme Using a 12 × 12 Dynamic S-Box. *IEEE Access* **2024**, *12*, 37631–37642. [CrossRef]
12. Özkaynak, F. On the effect of chaotic system in performance characteristics of chaos based s-box designs. *Phys. A Stat. Mech. Its Appl.* **2020**, *550*, 124072. [CrossRef]
13. Khan, M.F.; Saleem, K.; Alotaibi, M.; Hazzazi, M.M.; Rehman, E.; Abbasi, A.A.; Gondal, M.A. Construction and Optimization of TRNG Based Substitution Boxes for Block Encryption Algorithms. *Comput. Mater. Contin.* **2022**, *73*, 2679–2696. [CrossRef]
14. Ibrahim, H.; Ozkaynak, F. A Substitution-Box Structure Based on Crowd Supply Infinite Noise TRNG. In Proceedings of the 9th International Symposium on Digital Forensics and Security, ISDFS, Elazig, Turkey, 28–29 June 2021; Volume 2021. [CrossRef]
15. Kim, G.; Kim, H.; Heo, Y.; Jeon, Y.; Kim, J. Generating Cryptographic S-Boxes Using the Reinforcement Learning. *IEEE Access* **2021**, *9*, 83092–83104. [CrossRef]
16. Yang, M.; Wang, Z.; Meng, Q.; Han, L. Evolutionary design of S-box with cryptographic properties. In Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops, ISPAW 2011, ICASE 2011, SGH 2011, GSDP 2011, Busan, Republic of Korea, 26–28 May 2011; p. 12. [CrossRef]
17. Kadhim, A.F.; Kamal, Z.A. Generating dynamic S-BOX based on Particle Swarm Optimization and Chaos Theory for AES. *Iraqi J. Sci.* **2018**, *59*, 1733–1745. [CrossRef]
18. Sahmoud, S.; Elmasry, W.; Shadi, A. Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher. *Int. Arab J. E-Technol.* **2013**, *3*, 17–26.
19. Oracle. SecureRandom. 2020. Available online: https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#SecureRandom (accessed on 23 June 2020).
20. Saha, R.; Geetha, G.; Kumar, G.; Kim, T.H. RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys. *Secur. Commun. Netw.* **2018**, *2018*, 9802475. [CrossRef]
21. Jane, M.; Bruno, H.; Moulds, R.; Nino, W.; Anthony, F. *Quantum-Safe Security Working Group Quantum Random Number Generators*; Cloud Security Alliance: Bellingham, WA, USA, 2016.
22. Quantique, I.D. What Is the Q in QRNG? 2020. Available online: https://www.idquantique.com/random-number-generation/overview/ (accessed on 7 July 2020).
23. Quantique, I.D. Understanding Quantum Cryptography. 2020. Available online: https://www.idquantique.com/quantum-safe-security/quantum-key-distribution/ (accessed on 7 July 2020).
24. Herrero-Collantes, M.; Garcia-Escartin, J.C. Quantum random number generators. *Rev. Mod. Phys.* **2017**, *89*, 015004. [CrossRef]
25. Cao, Z.; Zhou, H.; Yuan, X.; Ma, X. Source-independent quantum random number generation. *Phys. Rev. X* **2016**, *6*, 011020. [CrossRef]
26. Liu, J.; Yang, J.; Li, Z.; Su, Q.; Huang, W.; Xu, B.; Guo, H. 117 Gbits/s Quantum Random Number Generation with Simple Structure. *IEEE Photonics Technol. Lett.* **2017**, *29*, 283–286. [CrossRef]
27. Siswanto, M.; Rudiyanto, B. Designing of quantum random number generator (QRNG) for security application. In Proceedings of the 2017 3rd International Conference on Science in Information Technology (ICSITech), Bandung, Indonesia, 25–26 October 2017; pp. 273–277. [CrossRef]
28. Quantique, I.D. Gaming-and-Lotteries. Available online: https://www.idquantique.com/random-number-generation/applications/gaming-and-lotteries/ (accessed on 7 July 2020).
29. IDQ. Quantis-Random-Number-Generator. Available online: https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator (accessed on 7 July 2020).
30. Picek, S.; Marchiori, E.; Batina, L.; Jakobovic, D. Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions. *Lect. Notes Comput. Sci.* **2014**, *8672*, 822–831. [CrossRef]
31. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
32. Wang, Y.; Wong, K.W.; Li, C.; Li, Y. A novel method to design S-box based on chaotic map and genetic algorithm. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* **2012**, *376*, 827–833. [CrossRef]
33. Wang, Y.; Zhang, Z.; Zhang, L.Y.; Feng, J.; Gao, J.; Lei, P. A genetic algorithm for constructing bijective substitution boxes with high nonlinearity. *Inf. Sci.* **2020**, *523*, 152–166. [CrossRef]
34. Nitaj, A.; Susilo, W.; Tonien, J. A New Improved AES S-box with Enhanced Properties. *Lect. Notes Comput. Sci.* **2020**, *12248*, 125–141. [CrossRef]
35. Daemen, J.; Rijmen, V. *The Design of Rijndael*; Springer: Berlin/Heidelberg, Germany, 2002; ISBN 3540425802.

36. Gullasch, D.; Bangerter, E.; Krenn, S. Cache games Bringing access-based cache attacks on AES to practice. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 2–25 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 490–505.
37. Biryukov, A.; Khovratovich, D. Related-key cryptanalysis of the full AES-192 and AES-256. In *Advances in Cryptology ASIACRYPT Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–18. ISBN 3642103650.
38. Juremi, J.; Sulaiman, S.; Saad, N.H.M. A survey on various dynamic S-box implementation in block cipher encryption algorithm. *J. Appl.* **2019**, *3*, 2–5.
39. Gupta, M.D.; Chauhan, R.K. Secure image encryption scheme using 4D-Hyperchaotic systems based reconfigurable pseudo-random number generator and S-Box. *Integration* **2021**, *81*, 137–159. [CrossRef]
40. Zhao, M.; Liu, H.; Niu, Y. Batch generating keyed strong S-Boxes with high nonlinearity using 2D hyper chaotic map. *Integration* **2023**, *92*, 91–98. [CrossRef]
41. Arrag, S.; Hamdoun, A.; Tragha, A.; Khamlich Salah, E. Implementation of stronger AES by using dynamic S-box dependent of master key. *J. Theor. Appl. Inf. Technol.* **2013**, *53*, 196–204.
42. Alamsyah Improving the Quality of AES S-box by Modifications Irreducible Polynomial and Affine Matrix. In Proceedings of the 2020 5th International Conference on Informatics and Computing, ICIC 2020, Gorontalo, Indonesia, 3–4 November 2020. [CrossRef]
43. Alamsyah; Prasetiyo, B.; Ardian, M.N. Enhancement security AES algorithm using a modification of transformation ShiftRows and dynamic S-box. *J. Phys. Conf. Ser.* **2020**, *1567*, 032025. [CrossRef]
44. Manjula, G.; Mohan, H.S. Constructing key dependent dynamic S-Box for AES block cipher system. In Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016, Bengaluru, India, 21–23 July 2016; pp. 613–617. [CrossRef]
45. Agarwal, P.; Singh, A.; Kilicman, A. Development of key-dependent dynamic S-Boxes with dynamic irreducible polynomial and affine constant. *Adv. Mech. Eng.* **2018**, *10*, 1–18. [CrossRef]
46. Jacob, G.; Murugan, A.; Viola, I. Towards the Generation of a Dynamic Key-Dependent S-Box to Enhance Security. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 92.
47. Cassal-Quiroga, B.B.; Campos-Cantón, E. Generation of Dynamical S-Boxes for Block Ciphers via Extended Logistic Map. *Math. Probl. Eng.* **2020**, *2020*, 2702653. [CrossRef]
48. Mahmood Malik, M.S.; Ali, M.A.; Khan, M.A.; Ehatisham-Ul-Haq, M.; Shah, S.N.M.; Rehman, M.; Ahmad, W. Generation of Highly Nonlinear and Dynamic AES Substitution-Boxes (S-Boxes) Using Chaos-Based Rotational Matrices. *IEEE Access* **2020**, *8*, 35682–35695. [CrossRef]
49. Aguirre, H.; Okazaki, H.; Fuwa, Y. An evolutionary multiobjective approach to design highly non-linear Boolean functions. In Proceedings of the GECCO 2007: Genetic and Evolutionary Computation Conference, New York, NY, USA, 9–13 July 2007; pp. 749–756. [CrossRef]
50. Picek, S.; Jakobovic, D.; Golub, M. Evolving cryptographically sound boolean functions. In Proceedings of the GECCO 2013 Genetic and Evolutionary Computation Conference Companion, Amsterdam, The Netherlands, 6–10 July 2013; pp. 191–192. [CrossRef]
51. Kalaiselvi, K.; Kumar, A. A Novel Method to Design S-box Based on Genetic Algorithm and Particle Swarm Optimizationin AES-128 Cryptosystem. 2018, 118, 1443–1457. *Int. J. Pure Appl. Math.* **2018**, *118*, 1443–1457.
52. Alsaif, H.; Guesmi, R.; Kalghoum, A.; Alshammari, B.M.; Guesmi, T. A Novel Strong S-Box Design Using Quantum Crossover and Chaotic Boolean Functions for Symmetric Cryptosystems. *Symmetry* **2023**, *15*, 833. [CrossRef]
53. Iavich, M.; Kuchukhidze, T.; Okhrimenko, T.; Dorozhynskyi, S. Novel Quantum Random Number Generator for Cryptographical Applications. In Proceedings of the 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 6–9 October 2020; pp. 727–732. [CrossRef]
54. Shaw, G.; Sivaram, S.R.; Prabhakar, A. Quantum Random Number Generator with One and Two Entropy Sources. In Proceedings of the National Conference on Communications (NCC), Bangalore, India, 20–23 February 2019; IEEE: Piscataway, NJ, USA, 2019; p. 1. [CrossRef]
55. Mogos, G. Quantum Random Number Generator vs. In Random Number Generator. In Proceedings of the IEEE International Conference on Communications, Kuala Lumpur, Malaysia, 22–27 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 423–426.
56. IDQ. *Quantum versus Classical Random Number Generators*; ID QUANTIQUE SA: Geneva, Switzerland, 2020.
57. Albrecht, M.R. Algebraic Techniques in Cryptanalysis. *Ecrypt2[Ppt]* **2011**, *3*, 120–141.
58. Bardeh, N.G.; Rønjom, S. Practical attacks on reduced-round AES. *Lect. Notes Comput. Sci.* **2019**, *11627*, 297–310. [CrossRef]
59. Abd-El-Atty, B. Efficient S-box construction based on quantum-inspired quantum walks with PSO algorithm and its application to image cryptosystem. *Complex Intell. Syst.* **2023**, *9*, 4817–4835. [CrossRef]
60. Kalaiselvi, K.; Kumar, A. Enhanced AES cryptosystem by using genetic algorithm and neural network in S-box. In Proceedings of the 2016 IEEE International Conference on Current Trends in Advanced Computing ICCTAC 2016, Bangalore, India, 10–11 March 2016. [CrossRef]
61. Zhu, D.; Zhang, M.; Tong, X.; Wang, Z. A Novel S-box Optimization Method Based on Immune Genetic Algorithm. In Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, Beijing, China, 21–23 October 2022; p. 32. [CrossRef]

62. Zhu, D.; Tong, X.; Zhang, M.; Wang, Z. A new s-box generation method and advanced design based on combined chaotic system. *Symmetry* **2020**, *12*, 2087. [CrossRef]
63. Community, T.S. Hamming. Available online: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.hamming.html (accessed on 9 July 2020).
64. Team, T.S.D. Sage. Available online: https://doc.sagemath.org/html/en/reference/cryptography/index.html (accessed on 14 November 2022).
65. Picek, S.; Batina, L.; Jakobović, D.; Ege, B.; Golub, M. S-box, SET, Match: A Toolbox for S-box Analysis. In Proceedings of the 8th IFIP International Workshop on Information Security Theory and Practice (WISTP), Heraklion, Crete, Greece, 30 June–2 July 2014; pp. 140–149. [CrossRef]
66. YongWang, D. S-Box Performance Analysis. Available online: http://42.192.236.76/Login (accessed on 17 March 2023).
67. Ahmad, M.; Haleem, H.; Khan, P.M. A new chaotic substitution box design for block ciphers. In Proceedings of the 2014 International Conference on Signal Processing and Integrated Networks, SPIN 2014, Noida, India, 20–21 February 2014; pp. 255–258. [CrossRef]
68. Ahmad, M.; Mittal, N.; Garg, P.; Maftab Khan, M. Efficient cryptographic substitution box design using travelling salesman problem and chaos. *Perspect. Sci.* **2016**, *8*, 465–468. [CrossRef]
69. Zahid, A.H.; Arshad, M.J.; Ahmad, M. A novel construction of efficient substitution-boxes using cubic fractional transformation. *Entropy* **2019**, *21*, 245. [CrossRef]
70. Attaullah; Jamal, S.S.; Shah, T. A Novel Algebraic Technique for the Construction of Strong Substitution Box. *Wirel. Pers. Commun.* **2018**, *99*, 213–226. [CrossRef]
71. Zahid, A.H.; Arshad, M.J. An innovative design of substitution-boxes using cubic polynomial mapping. *Symmetry* **2019**, *11*, 437. [CrossRef]
72. Ejaz, A.; Shoukat, I.A.; Iqbal, U.; Rauf, A.; Kanwal, A. A secure key dependent dynamic substitution method for symmetric cryptosystems. *PeerJ Comput. Sci.* **2021**, *7*, 587. [CrossRef]
73. Gao, W.; Idrees, B.; Zafar, S.; Rashid, T. Construction of Nonlinear Component of Block Cipher by Action of Modular Group PSL(2, Z) on Projective Line PL(GF(2 8)). *IEEE Access* **2020**, *8*, 136736–136749. [CrossRef]
74. Alzaidi, A.A.; Ahmad, M.; Ahmed, H.S.; Solami, E. Al Sine-Cosine Optimization-Based Bijective Substitution-Boxes Construction Using Enhanced Dynamics of Chaotic Map. *Complexity* **2018**, *2018*, 9389065. [CrossRef]
75. Lu, Q.; Zhu, C.; Wang, G. A Novel S-Box Design Algorithm Based on a New Compound Chaotic System. *Entropy* **2019**, *21*, 1004. [CrossRef]
76. Ahmad, M.; Doja, M.N.; Beg, M.M.S. ABC Optimization Based Construction of Strong Substitution-Boxes. *Wirel. Pers. Commun.* **2018**, *101*, 1715–1729. [CrossRef]
77. Wang, X.; Yang, J. A novel image encryption scheme of dynamic S-boxes and random blocks based on spatiotemporal chaotic system. *Optik* **2020**, *217*, 164884. [CrossRef]
78. Belazi, A.; El-Latif, A.A.A. A simple yet efficient S-box method based on chaotic sine map. *Optik* **2017**, *130*, 1438–1444. [CrossRef]
79. Liu, H.; Kadir, A.; Xu, C. Cryptanalysis and constructing S-Box based on chaotic map and backtracking. *Appl. Math. Comput.* **2020**, *376*, 125153. [CrossRef]
80. Soto, R.; Crawford, B.; Molina, F.G.; Olivares, R. Human Behaviour Based Optimization Supported with Self-Organizing Maps for Solving the S-Box Design Problem. *IEEE Access* **2021**, *9*, 84605–84618. [CrossRef]
81. Khan, M.F.; Saleem, K.; Shah, T.; Hazzazi, M.M.; Bahkali, I.; Shukla, P.K. Block Cipher's Substitution Box Generation Based on Natural Randomness in Underwater Acoustics and Knight's Tour Chain. *Comput. Intell. Neurosci.* **2022**, *2022*, 8338508. [CrossRef] [PubMed]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.