2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)

2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID) | 979-8-3503-8440-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSID60093.2024.00101

Designing Hash and Encryption Engines using Quantum Computing

Suryansh Upadhyay
The Pennsylvania State University
University Park, PA, USA
sju5079@psu.edu

Rupshali Roy
The Pennsylvania State University
University Park, PA, USA
rzr5509@psu.edu

Swaroop Ghosh
The Pennsylvania State University
University Park, PA, USA
szg212@psu.edu

***Abstract*—Quantum computing (QC) holds the promise of revolutionizing problem-solving by exploiting quantum phenomena like superposition and entanglement. It offers exponential speed-ups across various domains, from machine learning and security to drug discovery and optimization. In parallel, quantum encryption and key distribution have garnered substantial interest, leveraging quantum engines to enhance cryptographic techniques. Classical cryptography faces imminent threats from quantum computing, exemplified by Shor's algorithm's capacity to breach established encryption schemes. However, quantum circuits and algorithms, capitalizing on superposition and entanglement, offer innovative avenues for enhancing security. In this paper we explore quantum-based hash functions and encryption to fortify data security. Quantum hash functions and encryption can have numerous potential application cases, such as password storage, digital signatures, cryptography, anti-tampering etc. The integration of quantum and classical methods demonstrates potential in securing data in the era of quantum computing.**

***Index Terms*—Quantum Computing, Hash, Encryption, Decryption**

I. Introduction

Quantum computing (QC) has become the focus of extensive research and attention, driven by its potential to revolutionize problem-solving across diverse domains. When tackling combinatorial problems, quantum computers can achieve exponential speedups by harnessing quantum-mechanical phenomena such as superposition and entanglement. The broad spectrum of applications for quantum computing spans areas such as machine learning [1], security [2], drug discovery [3] etc. However, the practical realization of quantum computing face significant obstacles, such as qubit decoherence, measurement inaccuracies, gate errors, and temporal fluctuations. Quantum error correction (QEC) codes present a promising solution for ensuring reliable quantum operations. Still, their current demand for substantial resources makes them impractical for widespread adoption in the immediate future. The emergence of Noisy Intermediate-Scale Quantum (NISQ) computers, which have a limited number of qubits and operate in the presence of noise, along with various hybrid algorithms offer a potential solution to important problems such as discrete optimization, quantum chemical simulations and drug discovery.

In the wake of recent advancements in quantum computing, there has been a notable surge of interest in the domains of quantum encryption and quantum key distribution. These developments have ignited intriguing possibilities such as , use of quantum algorithms for developing robust and efficient cryptographic primitives [4] [5] that form the backbone of secure data communication and storage.

Motivation: The advent and rapid progress in quantum computing have ushered in a new era that both threatens and inspires innovation in the realm of cryptographic codes and data security. **a) Threat to classical cryptography:** Quantum computing's computational power poses threat to classical cryptographic codes. Shor's algorithm, for example, has the potential to break widely-used encryption schemes like RSA by efficiently factoring large numbers [9]. This capability undermines the security foundations of much of today's digital communication and data protection. **b) The need for quantum-resistant solutions:** As quantum computers advance, many classical encryption methods will become obsolete hence there is an urgent need for cryptographic systems that are resilient to quantum attacks. Developing and deploying quantum-resistant cryptographic codes is a technical imperative in safeguarding sensitive information in a quantum-powered world. **c) Quantum algorithms as a solution:** However, in the face of this quantum challenge, there lies an opportunity for innovation and enhanced security. Quantum circuits and algorithms, with their unique properties like superposition and entanglement, can be used to design cryptographic solutions that might be more secure against attacks while also being more robust in terms of performance.

In this work we explore the potential of quantum-based hash function and encryption algorithm in fortifying data security. Quantum hash functions can leverage quantum properties to enhance data integrity verification and collision resistance, addressing vulnerabilities exposed by quantum computing. It can serve as a robust tool for verifying the integrity of data and safeguarding against potential adversarial threats. For instance, a user can utilize a quantum hash function to create a distinctive hash value for their data. Once the data is secured using this quantum hash function, it is transmitted to the recipient. Alongside the data, a seed that specifies the specific hash function used is also shared. The recipient can then verify the incoming data by applying the same hash function. Furthermore, the quantum hash function's inherently chaotic dynamics may enable it's use in various other applications, including the generation of pseudo-random numbers

979-8-3503-8440-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VLSID60093.2024.00101

and the development of image encryption algorithms based on quantum hash functions.

State-of-the-art implementations of quantum AES offer improved quantum cryptanalysis estimations, i.e. they show resistance against quantum algorithms such as Shor's algorithm or Grover's algorithm, unlike existing classical encryption standards. However, our objective is to show that encryption can be done in a completely different way (and by following completely different steps/algorithms) using quantum computing. It can potentially be scaled to larger dimensions and evaluated for resistance to quantum cryptanalysis however, the present study is limited to conceptual level. As such, we have not compared the proposed design with existing large scale quantum AES or delved into the cryptanalysis estimations.

The quantum-enhanced encryption protocol utilizes a lookup table and a matrix operator (which is applied on qubits in the form of a quantum circuit) for data encryption. The sender encrypts the binary data, resulting in ciphertext (represented as measured qubits to the receiver). To successfully decrypt the data, the receiver also receives a seed that contains the lookup table and matrix operator used in encryption. Since the lookup table is the inverse of itself, the receiver applies the inverse of the matrix and lookup table in reverse order. It is important to ensure that equally efficient hardware/backends, having the same coupling map are used by both the sender and the receiver for the encryption/decryption process, so that the matrix operator and its inverse translate into the desired quantum circuit and its reverse without extra number of gates and circuit depth (which in turn will maintain similar level of noise in encryption and decryption process).

Paper organization: Section II provides background information on quantum computing, terms used and related work. Section III discusses the quantum based hash functions. In section IV we present a quantum based implementation of AES-128. Section V concludes the paper.

II. Background

A. Qubits and Quantum gates

Qubits due to their quantum nature, can exist in a superposition of both $|0\rangle$ and $|1\rangle$ unlike classical bits. As a result, a n-qubit system can represent all 2^n basis states at the same time. A qubit's state, denoted by φ = a $|0\rangle$ + b $|1\rangle$, can be expressed as a combination of complex probability amplitudes, a and b, corresponding to the states $|0\rangle$ and $|1\rangle$. A quantum program performs a series of gate operations on a group of correctly initialized qubits (using laser pulses in ion trap qubits and RF pulses in superconducting qubits). Mathematically, quantum gates are represented using unitary matrices (a matrix U is unitary if $UU^\dagger$ = I, where $U^\dagger$ is the adjoint of matrix U and I is the identity matrix)

B. Parameterized quantum circuits (PQC)

PQC is built from a collection of parameterized and controlled single qubit gates. A classical optimizer optimizes the parameters iteratively to achieve the desired input-output relationship. A PQC is used by a quantum processor to prepare a quantum state. The classical computer generates a new set of optimized parameters for the PQC based on the output distribution, which is then fed back to the quantum computer. The entire procedure continues in a closed loop until a traditional optimization target is reached.

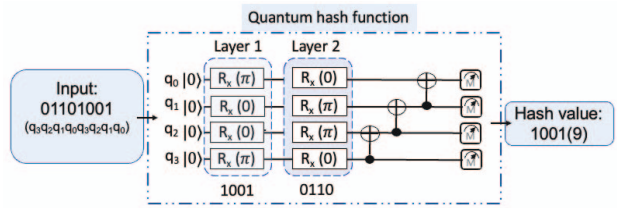

Fig. 1: Generating a 4-bit numerical hash value from an 8-bit input bit string. Encoding occurs in two layers within the circuit: the first layer encodes the first four bits using rotation angles, employing π radians for bit value 1 and 0 radians for bit value 0. The subsequent layer encodes the remaining four bits.

C. Hash function

A hash function is a mathematical function that converts variable-length data into fixed-length values, though some hash functions can also produce variable-length outputs. A hash function's output is commonly referred to as hash values, hash codes, digests, or simply hashes. These hash values are typically used to index a hash table, which is a fixed-size data structure. A hash function accepts a key as input to uniquely identify a datum or record within a data storage. These keys can be fixed-length, such as integers, or variable-length, such as names. In some cases, the key itself may represent the datum. This process yields a hash code, which is used to effectively index a hash table containing the data or records, or references to them.

D. Encryption

Advanced Encryption Standard (AES), is a popular symmetric encryption algorithm for data and communications security. This encryption method uses a (AES-n) n-bit key, which means it encrypts and decrypts data using n bits of data as the secret key. AES-128 is widely used in a variety of applications, including secure communication protocols, data encryption, and data protection mechanisms in modern computing systems, due to its efficiency and robust security features.

E. Related work

Several studies have focused on creating optimal quantum circuits for block ciphers. Kim et al. proposed optimal quantum circuits for SHA-2 based on its message expansion function in [12], while Song et al. presented a new quantum circuit implementation for SM3 in [13]. These studies demonstrate that optimal classical circuit designs can be adapted into optimal quantum circuits, although they face challenges such as high depth and error rates. In our work, we investigate well-established and error-tolerant PQCs as potential hash

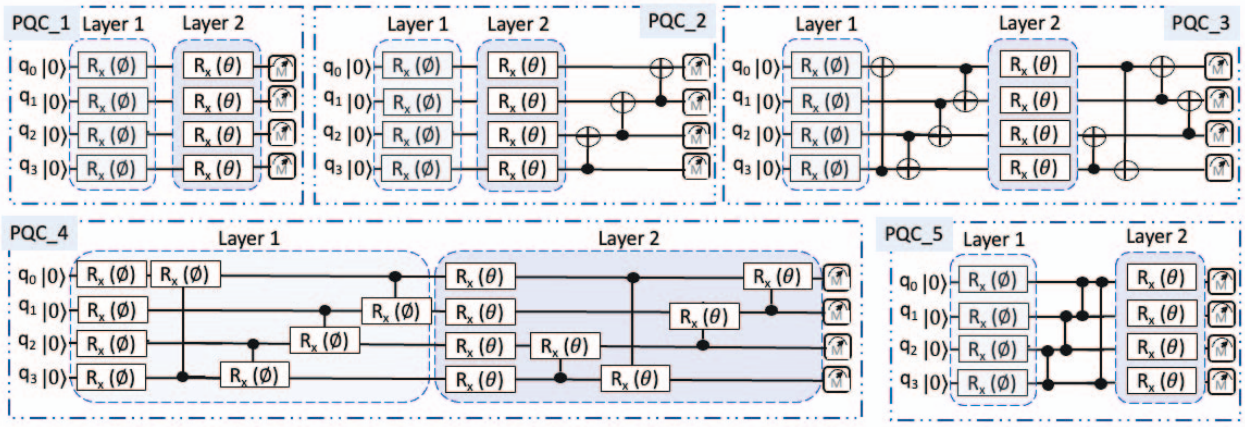

Fig. 2: Different PQC circuits explored as candidates for hash functions.

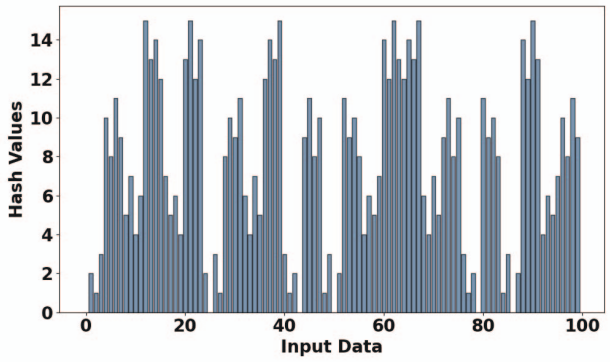

Fig. 3: Validation of PQC-3 as a quantum hash function on fake backend (fake_vigo[6 qubit]). A 4-qubit PQC circuit variant processes a limited 100-input batch of 8-bit bitstrings (0 to 99) with 1000 shots each.

functions. Quantum implementations of AES have emerged, notably by Grassl et al. [5], refined by Kim et al. [6], and improved by Langenberg et al. [7], reducing quantum gate requirements. The AES has proven its resistance to quantum attacks with the costs of doubling key sizes [4]. Wang et al. presented an efficient quantum AES-128 implementation [8], demanding fewer gates and qubits. Kuang and Bettenburg introduced the Quantum Permutation Pad (QPP) offering a versatile symmetric encryption solution for both quantum and classical systems [10]. While QPP's unconventional gate permutations pose encryption strength questions, this paper presents a proof-of-concept for implementing AES-128 steps (SubBytes, MixColumns and ShiftRows) on 4-qubit data chunks, showcasing encryption and decryption transformations on a reduced-size image.

III. Quantum Hash Function

A. Basic Idea

In the context of quantum hash functions, it is important to note that there can be an extensive number of approaches and circuits that can potentially serve as effective hash functions. In this work, we focus on PQCs. These PQCs are employed to encode input data using rotation angles, ultimately generating a hash string or hash value as the output. An essential consideration in this implementation is the quantum circuit's ability to effectively and uniformly address the Hilbert space as this influences the distribution and quality of hash values generated by the quantum circuit and, consequently, the overall performance of the quantum hash function. PQCs offer the advantage of high-dimensional Hilbert spaces, potentially improving accuracy in hash function generation. To find a dependable quantum hash function, we adhere to a set of critical properties that a hash function must have. These characteristics serve as benchmarks for assessing the quality and efficacy of any hash function:

a) **Deterministic:** it consistently produces the same hash value for a given input, ensuring predictability.

b) **Fixed Size Output:** hash values are of a set size, simplifying handling.

c) **Efficient:** swift hash value generation supports real-time applications.

d) **Pre-image Resistance:** prevents reverse-engineering of the input from its hash value, enhancing security.

e) **Collision Resistance:** makes finding two inputs with the same hash value highly challenging, ensuring data integrity.

f) **Avalanche Effect:** small input changes yield significantly different hash values, bolstering security.

g) **Uniform Distribution:** hash values are evenly distributed across the entire range for clustering prevention.

B. Implementation

To ensure the robustness and reliability of our approach, we leverage well-established PQCs that have been extensively studied in the field of quantum computing [11]. Our approach involves the encoding of input data into these PQCs by representing the input as rotation angles for the circuit's rotation gates. We present a generalized encoding approach in Algorithm 1. Any data can be transformed into bit strings, the bit value within the bit string determines whether it is

Algorithm 1: Quantum hash function

Input: Input (image/integer/bit string); PQC circuit
Output: Quantum hash function with encoded inputs

1) Convert input to binary bit string.
2) Loop through each bit in the binary bit string (*bit_string*).
   a) For the first $\frac{n}{2}$ bits:
      i) If bit $\equiv$ '1', set Rx gate on qubit n (*Layer 1*) to θ_1.
      ii) If bit $\equiv$ '0', set Rx gate on qubit n (*Layer 1*) to 0 (or ϕ_1 if needed).
   b) For the last $\frac{n}{2}$ bits:
      i) If bit $\equiv$ '1', set Rx gate on qubit n (*Layer 2*) to θ_2.
      ii) If bit $\equiv$ '0', set Rx gate on qubit n (*Layer 2*) to 0 (or ϕ_2 if needed).
3) **Output:** Hash values of the input.

encoded using an angle denoted as θ_1 (for a bit value of 1) or ϕ_1 (for a bit value of 0) within the rotation gate. For example: let's consider an input bitstring: 1001. In this case, the rotation values for the rotation gates applied to the qubits (q_3, q_2, q_1, q_0) will be as follows: θ_1, ϕ_1, ϕ_1, θ_1.

Fig. 1 illustrates the process of generating a 4-bit numerical hash value from an 8-bit input bit string. In this particular scenario, we begin by setting all qubits to the 0 state. Within the circuit, the encoding process takes place in two distinct layers. We encode the first four bits of the input bit string as rotation angles in the first layer. If the bit has a value of 1, we use π radians as the corresponding angle, and if the bit has a value of 0, we use 0 radians. The remaining four bits of the input bit string are then encoded in the following layer of the circuit. The values obtained from measurements form the resulting hash value.

C. Result and analysis

1) Experimental setup: We leverage the Qiskit open-source quantum software development kit from IBM, employing a Python wrapper for simulations. For **benchmarks**, we make use of different PQCs [11] Fig.2. For **benchmark execution**, we utilize Qiskit's fake provider module (fake_vigo[6 qubit], fake_singapore[20 qubit]), which comprises noisy simulators mimicking real IBM Quantum systems through system snapshots. These snapshots contain crucial information about the quantum system, such as the coupling map, basis gates, and qubit parameters. For **performance metric** we use:

1) **Collision rate (CR):** serves as a metric for quantifying collisions in a quantum hash function when applied to a particular input dataset. It is defined as-

$$\text{CR} = \frac{f_{avg} + stdev}{2^{\text{qubits}}} \tag{1}$$

Here, f_{avg} and $stdev$ denotes the average frequency of hash values and standard deviation generated by a specific function

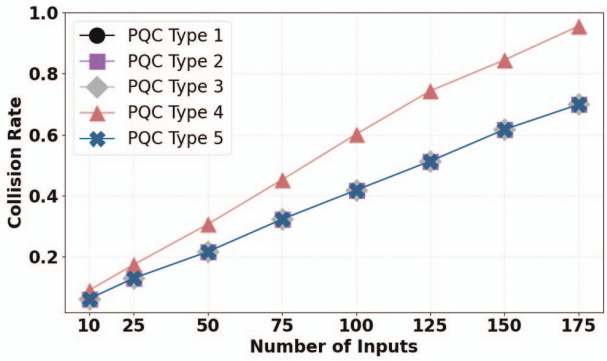

Fig. 4: The 4-qubit variants of various PQC circuits and their collision response to different batch sizes. Run on fake_vigo for 1000 shots each.

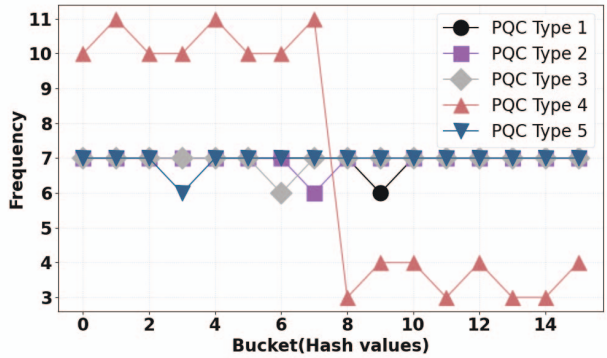

Fig. 5: Bucket histogram for the proposed PQC hash functions. A bucket represents an output hash value. We use 4-qubit PQCs to process a batch of 100 inputs represented as 8-bit bitstrings, run on fake_vigo for 1000 shots each.

for a given input data-set, and 2^{qubits} represents the total count of distinct possible hash values attainable by an n-qubit quantum hash circuit. A *lower collision rate indicates superior performance* of the quantum hash function when applied to the given input values.

2) **Buckets histograms:** provide insights into the distribution of hash values among various buckets and reveal the frequency of hash value recurrence within a given input set. These histograms serve as a valuable tool for identifying anomalous patterns. *In an ideal scenario, hash values should exhibit uniform distribution, indicating that they appear with equal frequency across the entire range* to mitigate clustering.

3) **Statistical goodness test:** The chi-squared test is used to ascertain whether an observed probability distribution aligns with a known and expected distribution. In our specific context, our objective is to assess whether the distribution of output hash values across all possible outputs follows a uniform and random pattern. The p-value derived from this test quantifies the likelihood that the behavior of the hash function resembles that of a uniformly distributed random variable or not. *A p-value close to 1 indicates strong hash*

function performance, signifying that the observed distribution is not statistically significantly different from the expected distribution.

2) Concept validation: We validate PQC's (PQC_3) application as a hash function using a fake backend (fake_vigo) Fig.3. A 4-qubit variant of the PQC circuit is used, processing 100 input batch (due to limited hardware availability) of 8-bit bitstrings (ranging from 0 to 99) run for 1000 shots each. For simplicity the values of θ, ϕ are either 0 or π, depending on the input bit being encoded (π if bit is 1, else 0). The figure demonstrates the generation of well-distributed hash values. This empirical validation highlights the practicality of the PQC circuit as a hash function.

3) Performance evaluation: We assess the performance of the proposed PQC hash functions using a bucket histogram, as depicted in Fig. 5. We use 4-qubit PQCs, processing a batch of 100 inputs (8-bit bitstrings). Notably, PQCs 1, 2, 3, and 5 exhibit robust performance, as their hash values are evenly distributed across the entire range. In contrast, PQC_4 displays a non-uniform frequency distribution. Furthermore, we have computed the p-values for each circuit: *PQC_1: 1, PQC_2: 1, PQC_3: 1, PQC_4: 0.02, PQC_5: 1*. A p-value close to 1 indicates a strong hash function performance, suggesting that the observed distribution closely aligns with the expected distribution, with the exception of PQC_4, which shows statistically significant deviation.

4) Batch size impact on collisions: The effect of batch size on collision rates is a critical consideration in evaluating the performance of hash function. We compare the 4 qubit variant of each PQC circuit and how they handle different batch sizes Fig. 4. For simplicity θ and ϕ take on either 0 or π. As the batch size increases, the collision rate tends to rise proportionally. When batch sizes are expanded, more data inputs are processed simultaneously. This increased input volume introduces a higher probability of two or more inputs coincidentally generating identical hash values, thus elevating the collision rate. We notice consistent collision performance across PQC 1, 2, 3, and 5, with PQC_4 exhibiting the highest collision rate among them.

D. Discussion

In PQC, the choice of encoding angles is crucial as it affects how points are distributed in hilbert space. In this work, we use angles of 0 and π to represent input bits 0 and 1, but there can be various ways to encode data. An interesting extension can be assigning different angles to input bits based on a weighting scheme, offering more precise control over how the output hash values are distributed. For instance, we could weigh the angles based on the probability distribution of input bit values, potentially improving the hash function's performance (collision resistance, cluster prevention), especially for data with non-uniform bit distributions. Noise can substantially affect the reliability and precision of quantum operations. Moreover, it can perturb the deterministic nature of quantum hash functions. Judicious selection of quantum circuit and encoding methodologies are instrumental in mitigating the effects of noise.

IV. Quantum AES implementation

A. Methodology and Results

We use the Qiskit for simulations and fake provider module (fake_valencia[5 qubit]) for **benchmark execution**. Our implementation adapts the *SubBytes*, *MixColumns*, *ShiftRows* steps of the classical AES-128 to a mix of classical and quantum operations to encrypt and decrypt data (Fig. 6). We illustrate the process using a 10X10 image of the alphabet A as data input (Fig. 6).

Encryption: We first convert the image into a binary bit string and segment it into 4-bit units. For *SubBytes* substitution, a classical 16-entry lookup table is employed, replacing each 4-bit segment with the corresponding entry. Next, these 4-bit units are processed through *MixColumns*. To apply a matrix to a circuit with n (4 in our case) qubits, we need a matrix of dimensions $2^n \times 2^n$, i.e. 16×16. Since this is a preliminary attempt at transforming the *MixColumns* step in the quantum domain, we have used a real matrix $[D]$ to the qubits. When used as an operator in the circuit, the matrix translates into a series of classical gates like controlled-NOT, controlled-CNOT, X and SWAP gates. Following this, *ShiftRows* is executed using SWAP gates, where specific bit segments undergo left circular shifts. Every $(4n+1)^{st}$ chunk undergoes a left circular shift by one position, while every $(4n+2)^{nd}$ chunk undergoes the shift by 2 positions, every $(4n+3)^{rd}$ chunk undergoes the shift by 3 positions. Every $(4n)^{th}$ chunk is left unchanged. The resulting processed bit segments are concatenated to generate a binary ciphertext. This ciphertext effectively conceals the original image data, rendering it indecipherable as an image file. To evaluate the encryption's impact, the binary string is examined using a raw pixel viewer.

Decryption: Each decryption step is the inverse of its corresponding encryption step, applied in reverse order. We start by applying the inverse of *ShiftRows* which uses a right circular shift on the bits by using the SWAP gates in the reverse order compared to encryption. Subsequently, we execute the inverse of *MixColumns* by applying the matrix operator $[D]^{-1}$ to the qubits. Finally, to undo the *SubBytes* step, each chunk of 4 bits is substituted by its corresponding values from the lookup table we used during the *SubBytes* step in encryption. This lookup table is designed to be self-inverse, enabling its utilization for decryption. The resulting values are concatenated into a binary bitstring, which is then transformed back into an image file, effectively restoring the original input. This successful decryption process demonstrates the reversibility of the encryption (Fig. 6).

B. Drawbacks and Scope for Improvement

The matrix employed for the *MixColumns* step is expressed in the form of CCX, CX, X, and SWAP gates within a corresponding quantum circuit. This representation implies that the qubits remain free from superposition or phase alterations.

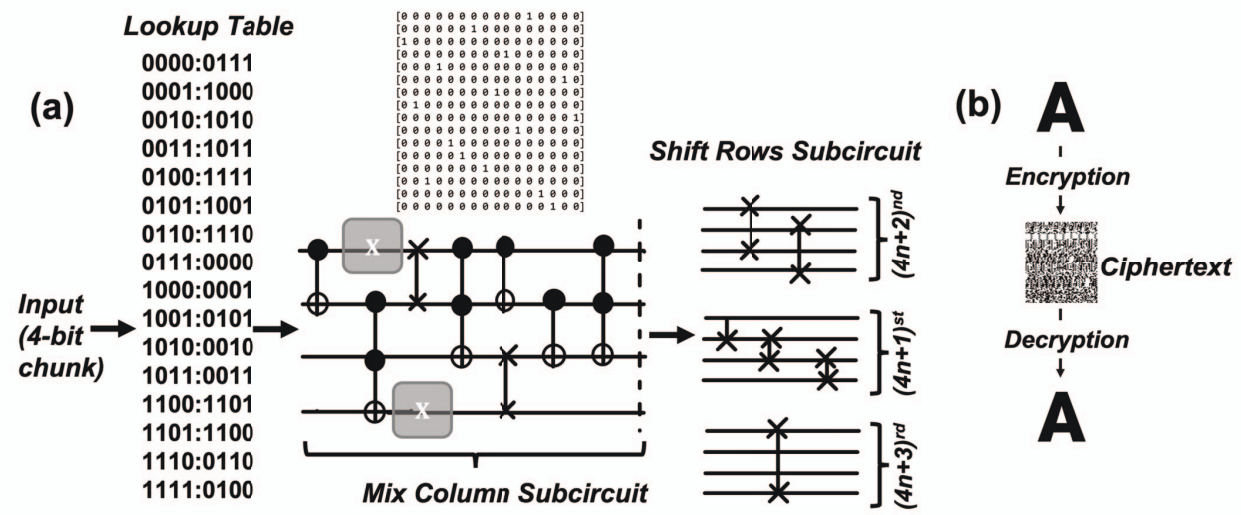

Fig. 6: a) Schematic of the circuitry used to encrypt the binary chunks from the image. b) Input image of the alphabet 'A', (reduced to 10×10 pixel), encrypted into a cipher image, then decrypted back into the original image.

As a result, when we measure both the ciphertext qubits and the final decrypted qubits, we consistently observe a single basis state with the highest frequency, thereby achieving a pure state. This characteristic translates to zero entropy, as there are no intrinsic uncertainties involved in the measurements. The allowable matrices could be expanded to include complex matrices, which would correspond to gates involving superposition and/or phase shifts. In such a scenario, measuring the ciphertext qubits would yield multiple basis states with the highest frequency for several 4-bit chunks, resulting in higher entropy. We could replace the lookup table used in the *SubBytes* step with a quantum read-only memory (QROM), to bring this step to the quantum domain from the classical domain. Additionally, our model does not incorporate the *AddRoundKey* step from AES, indicating that we do not use a key. However, this could be a plausible addition to enhance the model in the future.

V. Conclusion

Quantum computers pose a significant threat to classical encryption methods, demanding the development of quantum-resistant cryptography. Yet, quantum properties like superposition and entanglement offer a chance to innovate in cryptography. In this work we've explored the potential of quantum-based hash functions and AES algorithms for data security. Data integrity and collision resistance may be enhanced by quantum hash functions. Cryptography with quantum enhancements may guarantee stronger encryption and decryption, protecting sensitive data.

VI. Acknowledgment

This work is supported in parts by NSF (CNS-1722557, CNS-2129675, CCF-2210963, CCF-1718474, OIA-2040667, DGE-1723687, DGE-1821766, and DGE-2113839) and Intel's gifts.

References

[1] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," Nature Physics, 2019. [Online]. Available: https://doi.org/10.1038/s41567- 019- 0648- 8.

[2] Upadhyay, Suryansh, and Swaroop Ghosh. "Robust and Secure Hybrid Quantum-Classical Computation on Untrusted Cloud-Based Quantum Hardware." arXiv preprint arXiv:2209.11872 (2022).

[3] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," IBM Journal of Research and Development, vol. 62, no. 6, pp. 6–1, 2018.

[4] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, pp. 710-722, 2003.

[5] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying grover's algorithm to AES: Quantum resource estimates," in Post-Quantum Cryptography, 2016, pp. 2943.

[6] P. Kim, D. Han, and K. C. Jeong, "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: Applying to AES and SHA-2," Quantum Information Processing, vol. 17, pp. 1–39, 2018.

[7] B. Langenberg, H. Pham, and R. Steinwandt, "Reducing the cost of implementing the advanced encryption standard as a quantum circuit," IEEE Transactions on Quantum Engineering, vol. 1, pp. 1–12, 2020.

[8] Wang, ZG., Wei, SJ. & Long, GL. A quantum circuit design of AES requiring fewer quantum qubits and gate operations. Front. Phys. 17, 41501 (2022). https://doi.org/10.1007/s11467-021-1141-2

[9] V. Bhatia and K. R. Ramkumar, "An Efficient Quantum Computing technique for cracking RSA using Shor's Algorithm," 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2020, pp. 89-94, doi: 10.1109/ICCCA49541.2020.9250806.

[10] R. Kuang and N. Bettenburg, "Shannon perfect secrecy in a discrete hilbert space," in Proc. IEEE International Conference on Quantum Computing and Engineering (QCE), 2020, pp. 249-255.

[11] Hubregtsen, T., Pichlmeier, J., Stecher, P. et al. Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. Quantum Mach. Intell. 3, 9 (2021). https://doi.org/10.1007/s42484-021-00038-w

[12] Kim, P., Han, D., Jeong, K.C.: Time-space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. 17(12), 339 (2018)

[13] Song, G., Jang, K., Kim, H., Lee, W.-K., Zhi, H., Seo, H.: Grover on SM3. IACR Cryptol. ePrint Arch. 2021, 668 (2021)