Known IV attack on PANAMA Stream Cipher using Grover's Algorithm*

Rajeswari S, Kunwar Singh, and Sanakaranarayanan R†

National Institute of Technology, Trichy, Tamilnadu, India
srajes84@gmail.com, {kunwar, sankar}@nitt.edu

Abstract

Panama is a cryptographic module that functions both as a cryptographic hash and a stream cipher. It is specifically designed for high efficiency in software implementations on 32-bit architectures. In this paper, we analyzed the Panama stream cipher in relation to Grover's search algorithm. We constructed reversible quantum circuits for the Panama stream cipher and developed a simplified version of Quantum Panama using Qiskit programming. We then applied Grover's algorithm to this version to extract the secret key under a known IV attack model. In this model, the adversary knows some pairs of initialization vectors (IV) and keystreams to discover the secret key. Although not widely used, Panama's state-based design inspired modern sponge ciphers like Keccak; studying it aids in strengthening lightweight encryption for secure mobile and IoT systems in the quantum era.

1 Introduction

Quantum computing is an advanced model of computation that utilizes the principles of quantum mechanics to process data. Quantum computing provides a new paradigm in computation by exploiting superposition, which allows qubits to exist in multiple states simultaneously, and entanglement, which enables strong correlations between qubits, thereby enabling faster and more efficient processing of information.

Quantum cryptanalysis uses quantum algorithms to study and potentially compromise cryptographic systems that would remain secure against traditional, classical attacks. The two most significant quantum algorithms used in this field are Shor's algorithm[16] and Grover's algorithm[9].

In 1994, Peter Shor, in his seminal work, presented a quantum algorithm[16] that solves the prime factorization and discrete logarithm problems in polynomial time on a Quantum computer. In other words, once quantum computer becomes a reality all the existing public key cryptosystems will be broken in polynomial time.

In 1996, Lov Grover, proposed a quantum search algorithm known as Grover's Algorithm [9], which is designed to search an unsorted database of $N = 2^n$ elements in $O(\sqrt{N})$ time, providing a quadratic speedup over classical search algorithms. Grover's algorithm may not be applied in normal database searching because it requires an oracle (a black-box function) that can identify the marked element, which is not naturally available in classical databases. In cryptanalysis, an oracle can be constructed using a known-plaintext attack, where it is assumed that the adversary has access to one or more plaintext-ciphertext pairs to facilitate key recovery.

Recently, research has been focused on estimating the quantum resources required to apply

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 29, December 16-18, 2025, Sapporo, Japan.

†Corresponding Author

the Grover search algorithm to symmetric key cryptography. Grassl et al.[11] applied Grover's algorithm to the Advanced Encryption Standard (AES)[15]. They proposed quantum circuits for the three variants of AES, where $k = \{128, 192, 256\}$, and executed a Grover attack on AES-128, which was successful in $O(2^{64})$ time indicating that it may not be secure. So Grassl et al. suggested that AES-256 may be a quantum-safe variant of AES. Moreover, Grover's algorithm has recently gained attention for its potential impact on block ciphers ([4], [11], [8], [10]). Grover's algorithm has also been applied to stream ciphers such as RC4[2], ChaCha[3], and A5/1[1], as well as to authenticated encryption schemes[12].

Although designed before the quantum era, analysing Panama under quantum algorithms reveals its vulnerabilities and guides the development of quantum-safe lightweight ciphers for mobile and IoT security.

Our Contribution: In this work, we first design quantum circuits for the Panama stream cipher and estimate the quantum resources required for their implementation. Second, we present a known-IV attack on the Panama stream cipher using Grover's algorithm, which can recover the secret key in $O(2^{128})$ time. Finally, we implement a Grover-based attack on a simplified version of the Panama stream cipher in Qiskit[17] to demonstrate its feasibility.

2 Preliminaries

2.1 Quantum Basic concepts

A qubit is the basic unit of quantum information and is described as a linear combination (or superposition) of the computational basis states $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. Here, α and β are complex probability amplitudes. Upon measurement, the qubit collapses to either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively. This superpositional behavior is key to the inherent parallelism exploited by quantum algorithms.

Quantum gates are unitary operators that perform on qubits to transform their state. Gates are an extension of classical logic gates and can be performed on a single or multiple qubits. The most commonly utilized gates in quantum algorithms are:

- **Hadamard Gate (H):** Transforms a qubit into an equal superposition of $|0\rangle$ and $|1\rangle$, i.e., prepares a qubit for quantum parallelism.
- **Pauli Gates (X, Y, Z):** Represent quantum analogues of bit flips and phase flips.
- **CNOT Gate:** A two-qubit controlled gate that flips the target qubit if the control qubit is in state $|1\rangle$.
- **Controlled-Phase (CPhase or CZ):** Applies a conditional phase shift. These are particularly useful for encoding weighted relationships or applying constructive/destructive interference.
- **OR gate:** This gate computes $c = a \vee b$ where a, b and c are qubits. In quantum computing, there is no standard OR gate. The OR operation can be constructed using a Toffoli gate, as shown in Figure 1.
  These gates form the building blocks of quantum circuits, including those used in Grover's search and the Quantum Panama Cipher. For more detailed information about quantum gates, please refer to this source [13].

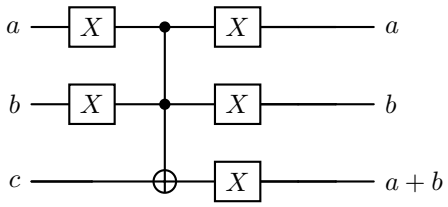


Figure 1: OR gate

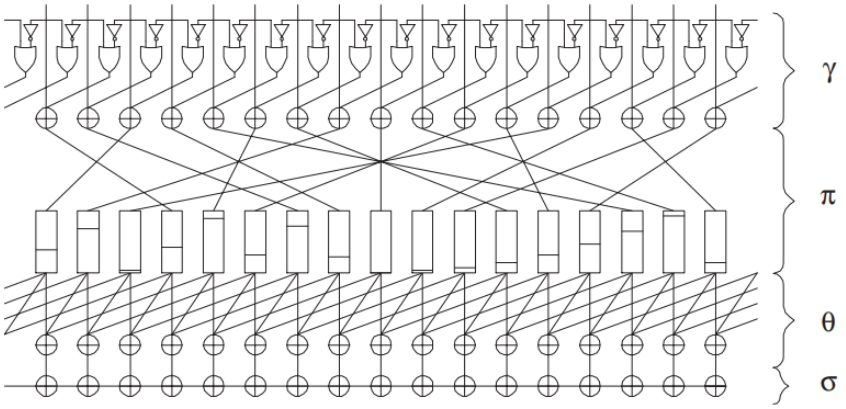


Figure 2: State Update Transformation

2.2 Panama Stream Cipher

The Panama stream cipher[5] is a cryptographic algorithm developed by Joan Daemen and Craig Clapp in 1998. The cipher is structured around a unique 544-bit state, represented by $a_0, a_1, ..., a_{16}$, where each a_i is a 32-bit word. Additionally, it features a buffer consisting of 32 stages, denoted as $b^0, b^1, ..., b^{31}$, with each b^i containing 8 words (256 bits). The words within each stage are represented as $b^i_0, b^i_1, b^i_2, ..., b^i_7$. The Panama cipher processes data through a combination of state transformations and buffer shifting techniques.

The state transition function of Panama operates on seventeen 32-bit words, named $a_0, a_1, ..., a_{16}$. Its steps are illustrated in Figure 2:

There are three key operations in Panama:

1. **Reset :** All the bits of State and Buffer initialised to Zero.

2. **Push :** This operation is performed twice and producing no output. In first time, input block $p = Key \in \{0,1\}^{256}$. In second time, input block $p = IV \in \{0,1\}^{256}$. It consists of five suboperations.

   (a) **Buffer Update :** During this operation, each stage is shifted right except stage 0 and stage 25. Stage 0 is computed by XORing stage 31 and input, Stage 25 is

computed by XORing stage 24 and stage 31 as follows:

$$\begin{aligned} b^j &= b^{j-1} \quad \text{if } j \notin \{0, 25\}, \\ b^0 &= b^{31} \oplus q, \\ b_i^{25} &= b_i^{24} \oplus b_{i+2 \bmod 8}^{31} \quad \text{for } 0 \leq i < 8. \end{aligned}$$

where q is the input block p.
$p = Key \in \{0,1\}^{256}$ in first Push operation.
$p = IV \in \{0,1\}^{256}$ in second Push operation.

(b) γ **function:** Let $c = \gamma(a)$ be a nonlinear word-level transformation applied to a sequence of 17 words $a = (a_0, a_1, \ldots, a_{16})$ where each a_i is a 32-bit word. The output $c = (c_0, c_1, \ldots, c_{16})$ is defined word-wise by the following equation:

$$c_i = a_i \oplus (a_{i+1} \vee \neg a_{i+2}), \quad \text{for } 0 \leq i < 17.$$

If circular indexing is used, then $a_{i+1} = a_{(i+1) \bmod 17}$, $a_{i+2} = a_{(i+2) \bmod 17}$. Each output word c_i is computed as the bitwise XOR of a_i with the bitwise OR of a_{i+1} and the bitwise negation of a_{i+2}.

(c) π **function:** Let $c = \pi(a)$ be a nonlinear permutation applied to a sequence of 17 words $a = (a_0, a_1, \ldots, a_{16})$, where each a_i is a 32-bit word. The output $c = (c_0, c_1, \ldots, c_{16})$ is defined word-wise by: $c_i = T_k(a_j)$, where: $j = 7i \bmod 17$, $k = \frac{i(i+1)}{2} \bmod 32$. Here, T_k denotes a bitwise rotation parameterized by k, and the index j determines a permutation of the input words and i denotes index of the word. The modular arithmetic ensures circular indexing across the 17-word input and 32-bit word size.

(d) θ **function:** We define the transformation $\theta : \{0,1\}^{17} \to \{0,1\}^{17}$ by:

$$c = \theta(a) \quad \text{where} \quad c_i = a_i \oplus a_{i+1} \oplus a_{i+4}, \quad \text{for } 0 \leq i < 17$$

All indices are taken modulo 17. This means $a_{i+k} := a_{(i+k) \bmod 17}$.

(e) σ **function:** This transformation defines how the output words $c_0, c_1 \ldots . c_{16}$ are derived from the input words $a_0, a_1 \ldots . . a_{16}$, depending on the operational mode (push or pull). Each word a_i is assumed to be a 32-bit word.

- **Initialization:** The first output word is computed by XORing the first input word with the constant 00000001_H:

$$c_0 = a_0 \oplus 00000001_H$$

- **For indices 1 to 8:** For $i = 0$ to 7, the words c_{i+1} are computed using a mode-dependent value l_i, as follows:
$c_{i+1} = a_{i+1} \oplus l_i$, for $0 \leq i < 8$
where l_i is the input block p.
$p = Key \in \{0,1\}^{256}$ in first Push operation.
$p = IV \in \{0,1\}^{256}$ in second Push operation.
- **For indices 9 to 16:** For $i = 0$ to 7, the words c_{i+9} are computed using the 16th internal buffer stage $b^{16} = (b_0^{16}, \ldots, b_7^{16})$:
$c_{i+9} = a_{i+9} \oplus b_i^{16}$, for $0 \leq i < 8$

3. **Pull mode:** It also contains five operations such as Buffer Update, γ function, π function, θ function, σ function. It takes no input and produces 256 bits Keystream as output by performing state transformation and Buffer Update operations.

   (a) **Buffer Update :** Same as in Push mode. Only difference is input block q is part of state a here.

   $$\begin{aligned} b^j &= b^{j-1} \quad \text{if } j \notin \{0, 25\}, \\ b^0 &= b^{31} \oplus q, \\ b_i^{25} &= b_i^{24} \oplus b_{i+2 \bmod 8}^{31} \quad \text{for } 0 \leq i < 8. \end{aligned}$$

   where $q = a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8$.

   (b) γ **function:** Same as it is in Push mode.

   (c) π **function:** Same as it is in Push mode.

   (d) θ **function:** Same as it is in Push mode.

   (e) σ **function:** This transformation defines how the output words $c_0, c_1 ....c_{16}$ are derived from the input words $a_0, a_1 .....a_{16}$, depending on the operational mode (push or pull). Each word a_i is assumed to be a 32-bit word.

   - **Initialization:** The first output word is computed by XORing the first input word with the constant 00000001_H:

   $$c_0 = a_0 \oplus 00000001_H$$

   - **For indices 1 to 8:** For $i = 0$ to 7, the words c_{i+1} are computed using a mode-dependent value l_i, as : $c_{i+1} = a_{i+1} \oplus l_i$, for $0 \leq i < 8$ where l_i is the input block p. $p = b^4 = (b_0^4, \ldots, b_7^4)$ is the 4th internal buffer stage .
   - **For indices 9 to 16:** For $i = 0$ to 7, the words c_{i+9} are computed using the 16th internal buffer stage $b^{16} = (b_0^{16}, \ldots, b_7^{16})$:
     $c_{i+9} = a_{i+9} \oplus b_i^{16}$ for $0 \leq i < 8$.

When Panama is used as a stream cipher, the process begins with a Push operation to input the key, followed by another Push operation to input the initialization vector(IV). After these inputs, 32 Pull operations are performed, during which the outputs are discarded. This step ensures that the internal state of Panama is fully mixed. Finally, a Pull operation produces the keystream bits.
The Summarized steps of Panama Stream cipher is illustrated in Table 1.

Slno	Cipher Mode
1	Reset
2	Push Key(256 bits)
3	Push IV(256 bits)
4	32 times Pull (no output)
5	Pull(Outputs 256 bits Keystream)

Table 1: Panama Cipher mode

2.3 Grover's Algorithm

Grover's search algorithm[9], developed by Lov Grover, is a quantum search algorithm that provides a quadratic speedup for searching unsorted databases. This makes it a crucial tool in quantum computing for tasks such as quantum cryptanalysis and optimization. The algorithm can be adapted to function as an oracle for cryptanalysis, allowing for efficient searches within the key space of stream ciphers.

Consider a stream cipher that takes as input a secret key $K \in \{0,1\}^k$ and an initialization vector $IV \in \{0,1\}^m$. It generates a keystream (ks) of length ρ bits using a deterministic function $S_{K,IV}$.
Objective: To Recover the secret key K_0 such that: $S_{K_0,IV} = ks$
Grover's Algorithm Steps:

1. **Define a Boolean function:** Construct a function $f : \{0,1\}^k \to \{0,1\}$ such that:

$$f(K) = \begin{cases} 1, & \text{if } S_{K,IV} = ks \\ 0, & \text{otherwise} \end{cases}$$

   This function identifies whether a given key K produces the target keystream.

2. **Initialize a uniform superposition over all keys:**

$$|K\rangle = \frac{1}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} |K_j\rangle$$

   This is achieved by applying Hadamard gates to all k qubits initialized in the $|0\rangle$ state.

3. **Apply Grover iteration, which consists of two steps:**
   (a) **Oracle:** Implement a quantum oracle that flips the phase of the state $|K_j\rangle$ if and only if $f(K_j) = 1$.
   (b) **Diffusion:** Apply the Grover diffusion operator to reflect the state about the average amplitude.

4. **Repeat the Grover iteration:** Approximately $O(2^{k/2})$ times to amplify the amplitude of the correct key state.

5. **Measure the quantum state:** Upon measurement, the system collapses to the correct key K_0 with high probability (at least $1/2$ if k is large enough and the iteration count is optimal).

Grover's algorithm allows us to recover the key of a stream cipher with quadratic speed-up over classical brute-force search, reducing the time complexity from $O(2^k)$ to $O(2^{k/2})$.

3 Reversible Circuits of Panama Stream Cipher

3.1 Panama Stream Cipher

The Panama stream cipher[5], developed by Joan Daemen and Craig Clapp in 1998. In Cipher mode, the algorithm requires a 256-bit key and a 256-bit initialization vector (IV) as input, producing a keystream with 256 bits at a time. To conduct an attack against a stream cipher, a quantum adversary must first implement the cipher as a reversible quantum circuit. In the following section we have designed quantum reversible circuits for Panama stream cipher.

3.2 Buffer Update function

The buffer consists of 32 stages labeled b^0 through b^{31}, where each stage b^i holds 8 words $(b_0^i, b_1^i, \ldots, b_7^i)$, amounting to 256 bits per stage. This function is denoted as $d = \lambda(b)$, where λ represents the Buffer Update function applied to the buffer register b.

i For all j except 0 and 25, the value of d^j is assigned as: $d^j = b^{j-1} \quad \text{for } j \notin \{0, 25\}$. That is, each d^j simply copies the value of the previous b register. The respective Quantum circuit for this is shown in Figure 3.

ii For $j = 0$, the update rule is: $d^0 = b^{31} \oplus q$. This means that d^0 is the bitwise XOR of b^{31} and input value q.
Push mode: q corresponds to the key or the initialization vector (IV).
Pull mode: q is the concatenation of 8 words:

$$q = a_1 \| a_2 \| a_3 \| a_4 \| a_5 \| a_6 \| a_7 \| a_8$$

. The respective Quantum circuit is shown in Figure 4.

iii For $j = 25$, the update applies at the bit level:

$$d_i^{25} = b_i^{24} \oplus b_{(i+2) \bmod 8}^{31} \quad \text{for } 0 \leq i < 8$$

. That is, for each bit index i, the i-th bit of d^{25} is computed as the XOR of:

- the i-th bit of b^{24}, and
- the $(i + 2)$ mod 8-th bit of b^{31}

The Quantum circuit for this is shown in Figure 5.

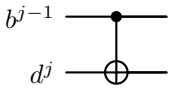


Figure 3: $d^j = b^{j-1}$

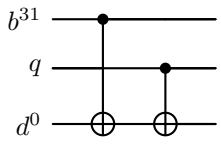


Figure 4: $d^0 = b^{31} \oplus q$

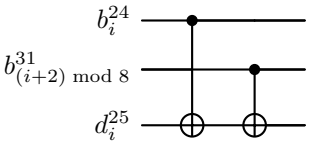


Figure 5: $d_i^{25} = b_i^{24} \oplus b_{(i+2) \bmod 8}^{31}$

Parameter	$d^j = b^{j-1}$	$d^0 = b^{31} \oplus q$	$d_i^{25} = b_i^{24} \oplus b_{(i+2) \bmod 8}^{31}$
Words per register	8	8	8
Bits per word	32	32	32
Total bits(w)	256	256	256
Total CNOT gates	$30 \times 256 =$ **7680**	$256 \times 2 =$ **512**	$8 \times 32 \times 2 =$ **512**
Gate depth (parallel)	30	2	2
Asymptotic complexity	$O(n \cdot w)$	$O(w)$	$O(w)$

Table 2: Gate complexity comparison for Buffer Update

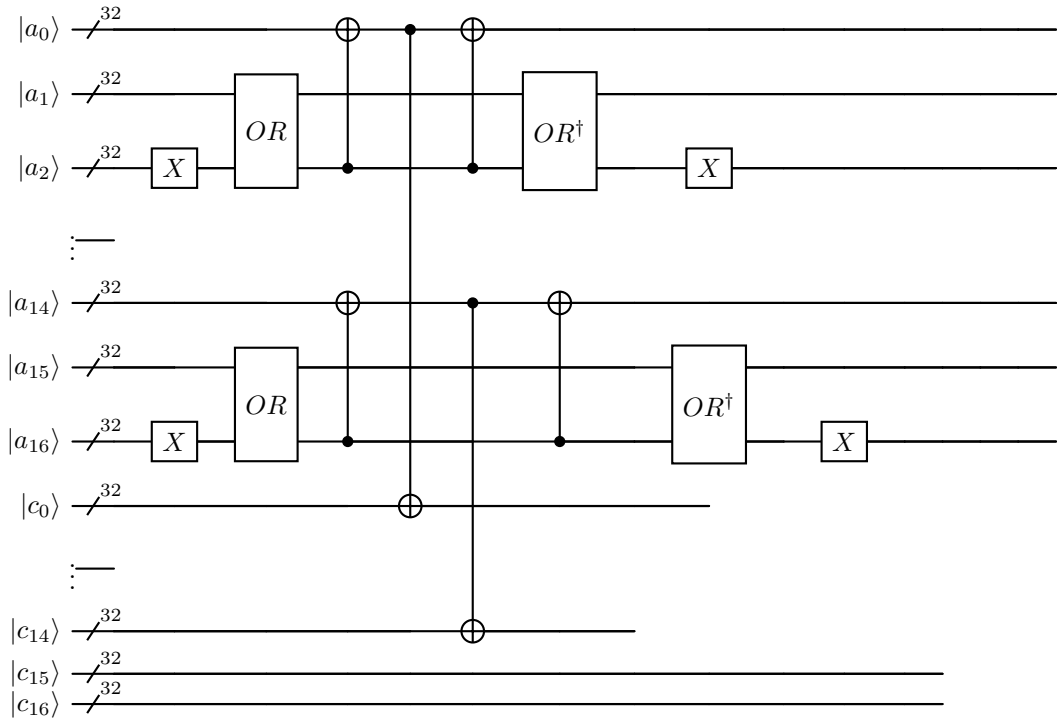


Figure 6: Quantum circuit for γ function

3.3 γ function

The γ function of the Panama stream cipher is a crucial nonlinear component of its transformation steps. It introduces nonlinearity and confusion into the internal state by combining state elements using bitwise logic operations, typically including AND, NOT, and XOR. This function is defined by: $c = \gamma(a) \leftrightarrow c_i = a_i \oplus (a_{i+1}$ OR $\overline{a_{i+2}}), 0 \leq i < 17$. The corresponding Quantum circuit for implementing this function is shown in Figure 6. In this figure, each qubit is 32 bits in size. The circuit operates as follows:
Suppose $c_0 = a_0 \oplus (a_1$ OR $\overline{a_2})$. First, the inverse of $\overline{a_2}$ is computed using X gates. Then, it is OR-ed with a_1 using an n-bit OR gate. The output of the OR gate is then XORed with a_0 to produce the final result, which is stored in c_0 using a CNOT gate. To retrieve the original values of a_0, a_1, and a_2, all gates are applied in reverse order. This process is repeated for all c_i values.

Gate Type	**Per Word (32 bits)**	**Total for 17 Words**
X (NOT)	192	3264
Toffoli	64	1088
CNOT	64	1088
Total Gates	**320**	**5440**

Table 3: Gate complexity for γ function

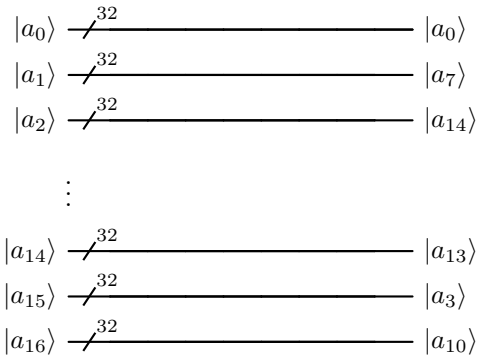


Figure 7: Quantum circuit for Permutation

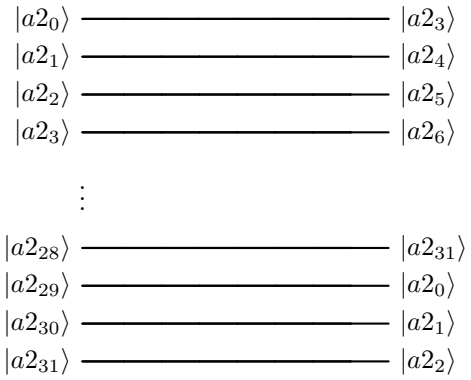


Figure 8: Shifting of Word2

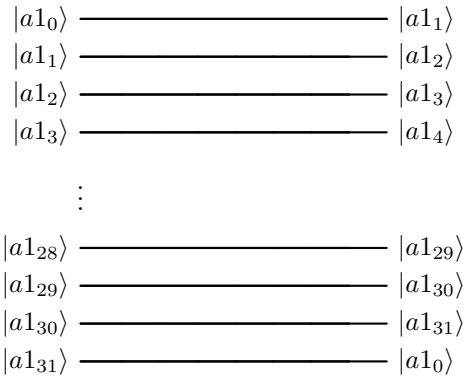


Figure 9: Shifting of Word1

3.4 π function

The permutation π consists of cyclic word shifts and a permutation of the word positions. We define T_k as a rotation of k positions from LSB to MSB. we have: $c = \pi(a) \leftrightarrow c_i = T_k(a_j)$, where $j = 7i \bmod 17$ and $k = i(i+1)/2 \bmod 32$, where i denotes the word index, j denotes the permutation order and k denotes the bit-shift positions.
The ordering of Permutation and Shifting is shown in Table 4. We do not need to add any

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j	0	7	14	4	11	1	8	15	5	12	2	9	16	6	13	3	10
k	0	1	3	6	10	15	21	28	4	13	23	2	14	27	9	24	8

Table 4: Permutation and Shifting order of words

gates to implement the permutation and shifting operations, as these operations correspond to a rearrangement of the qubits. Instead, we simply adjust the positions of the subsequent gates to ensure that the correct input wire is used.

Figure 7 represents the quantum circuit for the permutation of Words as specified in Table4. Figure 8 and 9 represents quantum circuits that performs a left circular shift of Word2 and

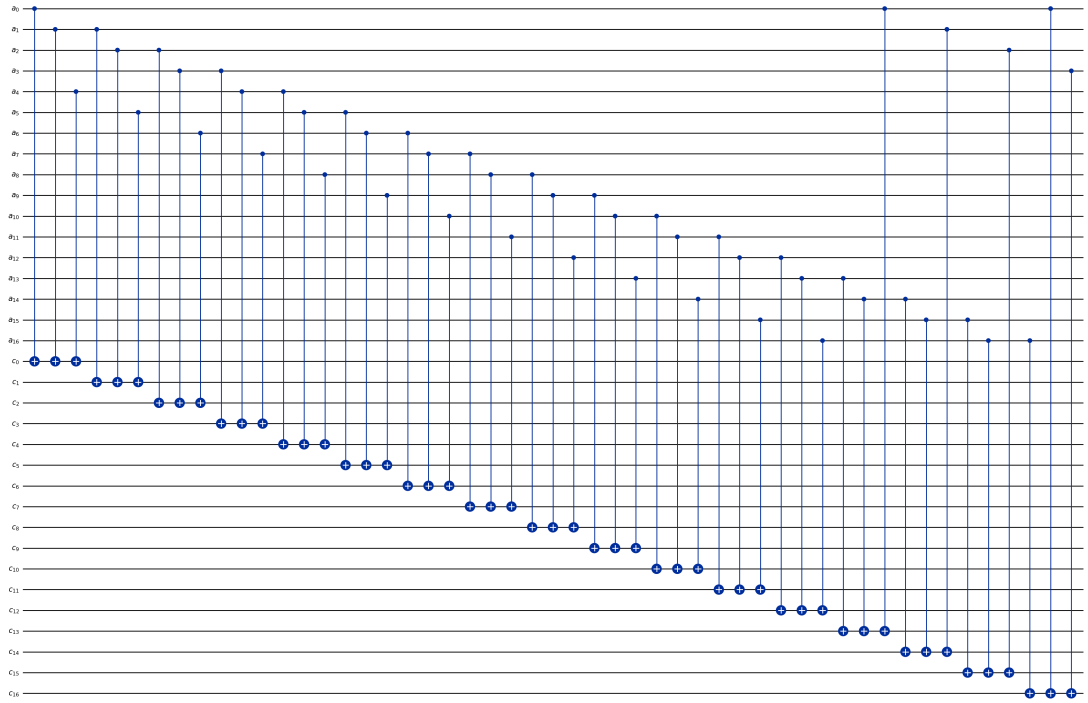


Figure 10: Quantum circuit for θ function

Word1 by 3 and 1 qubits respectively. Similarly, the remaining words in the state array can be shifted according to the bit positions specified in Table 4.

Gate Complexity: Qubit permutation and shifting are achieved through index reordering, without any physical gate execution.

3.5 θ function

We define the transformation $\theta : \{0,1\}^{17} \to \{0,1\}^{17}$ by: $c = \theta(a)$ where $c_i = a_i \oplus a_{i+1} \oplus a_{i+4}$, for $0 \leq i < 17$. All indices are taken modulo 17. This means: $a_{i+k} := a_{(i+k) \bmod 17}$. Each output is computed as : $c_0 = a_0 \oplus a_1 \oplus a_4$

$c_1 = a_1 \oplus a_2 \oplus a_5$

$c_2 = a_2 \oplus a_3 \oplus a_6$

$\vdots$

$c_{15} = a_{15} \oplus a_{16} \oplus a_2$

$c_{16} = a_{16} \oplus a_0 \oplus a_3$.

The respective Quantum circuit to compute θ function is shown in Figure 10. Each register a_i, c_i is a 32 qubit register. Each output qubit c_i is computed using exactly $3 \times 32 = 94$ CNOT gates. Total number of CNOT gates: $17 \times 94 = 1598$. No ancilla qubits are required beyond the 17 input and 17 output qubits. This circuit implements a linear reversible transformation.

Parameter	Value
Words in input register (a)	17
Bits per word	32
Total output words (c_i)	17
Total CNOT gates	**1088**
Gate depth (parallelizable)	2
Asymptotic complexity	$O(n \cdot w)$, where $n = 17, w = 32$

Table 5: Gate complexity for θ function

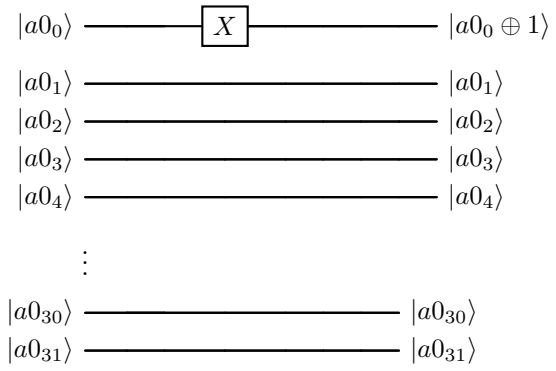


Figure 11: Circuit Diagram for $c_0 = a_0 \oplus 00000001_H$

3.6 σ Function

The transformation σ corresponds with bitwise addition of buffer and input words. It is given by (let $c = \sigma(a)$): $c_0 = a_0 \oplus 00000001_H$
$c_{i+1} = a_{i+1} \oplus l_i$ for $0 \leq i < 8$, where $l = p$ in push mode and $l = b^4$ in pull mode
$c_{i+9} = a_{i+9} \oplus b_i^{16}$ for $0 \leq i < 8$.

- **Computation of c_0:** The first output word is computed by XORing the first input word with the constant 00000001_H:

  $$c_0 = a_0 \oplus 00000001_H$$

  We are given a 32-qubit quantum register $a_0 = (a_0[31], \ldots, a_0[0])$, where $a_0[i]$ denotes the i-th qubit, with $a_0[0]$ as the least significant bit (LSB). Here, `00000001` in hexadecimal corresponds to $00000000000000000000000000000001_2$ in binary. So the XOR affects only bit 0 (LSB). This implies: Apply an X gate (NOT) to qubit $a0_0$ and Leave qubits $a0_1$ to $a0_{31}$ unchanged.

- **Computation of c_1 to c_8:** For $i = 0$ to 7, the words c_{i+1} are computed using a mode-dependent value l_i, as follows:

  $$c_{i+1} = a_{i+1} \oplus l_i$$

  where

  $$l_i = \begin{cases} p_i, & \text{in push mode} \\ b_i^4, & \text{in pull mode} \end{cases}$$

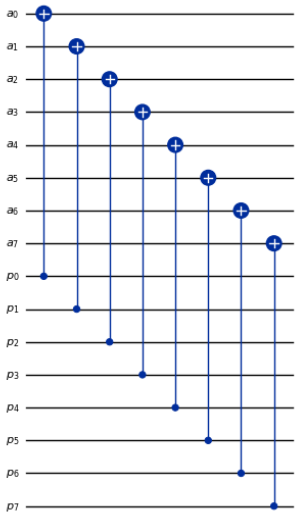


Figure 12: Quantum Circuit for $a_{i+1} \oplus p$

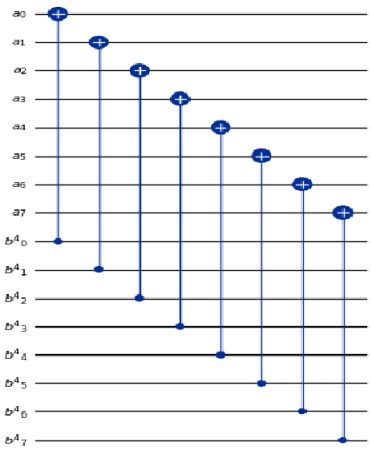
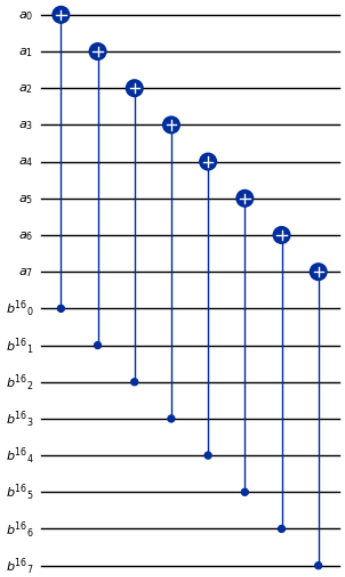


Figure 13: Quantum Circuit for $a_{i+1} \oplus b^4$

Here, $p = (p_0, \ldots, p_7)$ is the external input in push mode, and $b^4 = (b_0^4, \ldots, b_7^4)$ is the 4th internal state block in pull mode. Each qubit is a 32 qubit register. Each c_i is computed using 32 CNOT gates. Total number of gates required are $17 \times 32 = 544$. Each Figure 10 and Figure 11 requires 544 CNOT gates.

- **Computation of c_9 to c_{16}:** For $i = 0$ to 7, the words c_{i+9} are computed using the 16th internal state block $b^{16} = (b_0^{16}, \ldots, b_7^{16})$:

$$c_{i+9} = a_{i+9} \oplus b_i^{16}$$



– a_{i+1} is a 32-qubit quantum register.

– b_i^{16} is a 32-qubit slice from a 256-qubit register b^{16}.

– $\oplus$ represents bitwise quantum XOR, implemented via CNOT gates.

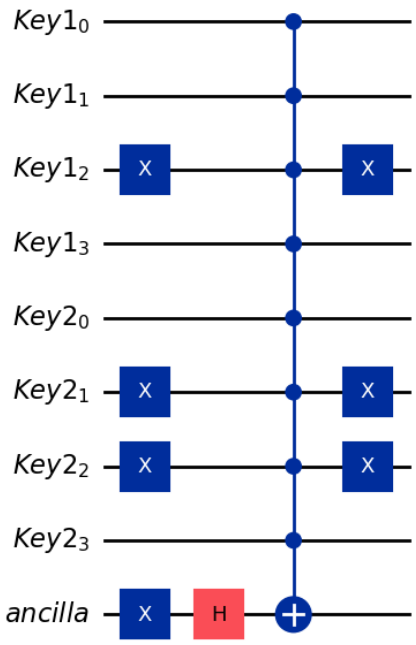
Figure 14: Quantum circuit for $c_{i+9} = a_{i+9} \oplus b_i^{16}$.

Parameter	$c_{i+1} = a_{i+1} \oplus l_i$	$c_{i+9} = a_{i+9} \oplus b_i^{16}$	$c_0 = a_0 \oplus 00000001_H$
Number of operations	8	8	1
Bits per word	32	32	32
CNOTs per operation	32	32	1
Total CNOT gates	**256**	**256**	**1**
Gate depth (parallelizable)	1	1	1
Asymptotic complexity	$O(n \cdot w)$	$O(n \cdot w)$	$O(1)$

Table 6: Gate complexity for σ function

4 Grover attack on Panama Stream Cipher

A Known-IV attack is a cryptanalytic strategy applied to stream ciphers in which the Initialization Vector (IV) is assumed to be publicly available. In this model, the adversary possesses knowledge of certain IVs together with their corresponding keystreams. To enable a quantum attack on the Panama stream cipher, the initial requirement is the design of a reversible quantum implementation of the cipher. The construction of this circuit is detailed in Section 3. Once realized, the reversible Panama circuit can be incorporated into an oracle (black box), which serves as the interface for the quantum adversary. This section provides an overview of Grover's oracle and the diffusion operator.



Oracle Circuit Explanation: Figure 15 shows the Qiskit implementation of Keystream comparison circuit. The circuit implements a Grover oracle that marks a specific key, for the keystreams (1011) and (1001) stored in the `Key1` and `Key2` registers. To enable phase kickback, the ancilla qubit is first prepared in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ using a sequence of X and H gates. These key1 and Key2 will act as control bits for multi-controlled X (Toffoli) gate. The `Key1` and `Key2` registers then serve as control inputs for a multi-controlled X (Toffoli) gate, ensuring that the oracle is triggered only when the registers match the specified bitstring. Once this condition is satisfied, the ancilla qubit flips, thereby introducing a phase inversion on the marked state, which is the critical step underlying Grover's search process.

Figure 15: Keystream Comparison Circuit

Grover's Oracle: It contains 2 fundamental building blocks.

1. **Quantum Black box:** It is a quantum circuit that simulates the cipher and produces a keystream of length $k + c$ bits.

2. **Comparison circuit:** This generated keystream is then compared with the known target keystream using $k+c$ controlled NOT gates. If the two keystreams match, the oracle flips the target key, indicating that the corresponding key is a valid solution. Considering a single IV and its keystream for a given key (K), then the probability that a different key (K') also satisfies the same IV–keystream pair is

$$\Pr_{K \neq K'} \left(\hat{C}_\rho(K) = \hat{C}_\rho(K') \right) \approx \frac{2^k - 1}{2^\rho}.$$

where k is length of key and ρ is length of keystream. Extending this to l IV–keystream pairs, the probability becomes

$$\Pr_{K \neq K'} \left(\hat{C}_\rho(K) = \hat{C}_\rho(K') \right) \approx \frac{2^k - 1}{(2^\rho)^l}$$

Hence, l must be chosen such that this probability is negligible. In our experiment, we set $l = 2$, as also illustrated in Figure 15.

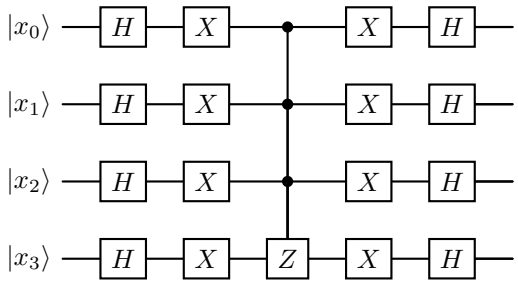


Figure 16: Diffusion Gate

Diffusion Operator: Grover's oracle inverts the phase of the key, but all keys maintain the same probability. We apply the diffusion operator to increase the amplitude of the target key while reducing the amplitude of other keys.
Diffusion gate = $H^{\otimes n} Z_0 H^{\otimes n}$ Where $Z_0 = 2\,|0\rangle\langle 0| - I$
In terms of matrix, it can be represented as

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

This Operator Z_0 will flip the phase of all states except zero state. That is

$$Z_0\,|x\rangle = \begin{cases} |x\rangle & \text{if } |x\rangle = |0^n\rangle \\ -\,|x\rangle & \text{otherwise} \end{cases}$$

In other words, it negates $|x\rangle$ if and only if logical OR of $\{x_1, x_2, ...x_n\}$ is 1. Logical OR gate in quantum is explained in section 2.1. Therefore Diffusion gate is an Logical OR gate sandwiched between Hadamard gates.

Illustration of Diffusion gate on 2-qubits :
$H^{\otimes n}\,|00\rangle = \frac{1}{4}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$.
Let us assume target state is $|10\rangle$,then
$= \frac{1}{4}\,|00\rangle + |01\rangle - |10\rangle + |11\rangle$.
Steps to apply Diffusion Gate:
Step 1: Apply Hadamard $H^{\otimes n}\frac{1}{4}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$
$= \frac{1}{4}(|00\rangle - |01\rangle + |10\rangle + |11\rangle)$
Step 2. Apply Z_0 $\frac{1}{4}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$
Step 3. Apply Hadamard $H^{\otimes n}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$
$= \frac{1}{4}(|00\rangle+|01\rangle+|10\rangle+|11\rangle+|00\rangle-|01\rangle+|10\rangle-|11\rangle-|00\rangle-|01\rangle+|10\rangle+|11\rangle-|00\rangle+|01\rangle+|10\rangle-|11\rangle)$
$= \frac{1}{4}(4\,|10\rangle) = |10\rangle$

5 Implementation of Grover Attack on Simplified Panama

The main objective of this paper is to implement a Grover attack on the Panama cipher. We have developed quantum circuits to carry out this attack. Implementing a Grover attack on the original Panama stream cipher requires a substantial number of qubits. Since current real-time

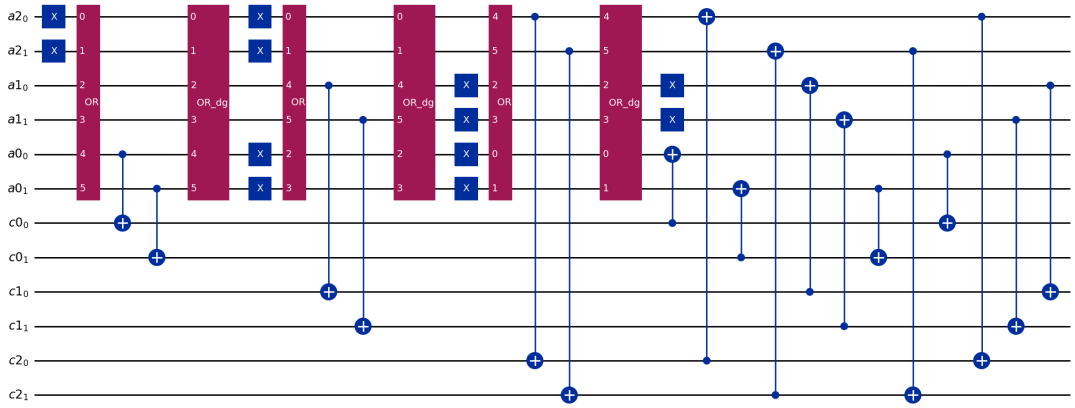


Figure 19: Quantum Circuit for Gamma function

quantum systems cannot support this many qubits, we have instead focused on executing a Grover attack on a simplified version of the Panama stream cipher. In this simplified version, we assume the state consists of three words, each containing two qubits, and the buffer contains four words, with each word also having two qubits. The following sections will describe the respective quantum circuits for this simplified version.

5.1 Buffer Update

In Simplified Panama, Buffer is 8 qubit register i.e 4 words of 2 qubits each. The transformations are defined as $b_0 = b_0 \oplus K$, $b_1 = b_0$, $b_2 = b_1$ and $b_3 = b_2$. Qubit b_0 is Exored with K and stored into b_0. The qubits b_0, b_1 and b_2 are shifted right by 2 bits to get b_1, b_2 and b_3. Quantum circuits for Buffer update function is shown in Figure 17 and Figure 18.

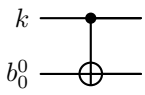


Figure 17: $b_0 = b_0 \oplus K$

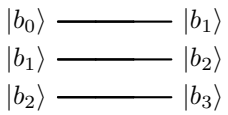


Figure 18: Shifting of Buffer

5.2 γ function

Since the state array consists of three words, the state is updated as follows:
$c_0 = a_0 \oplus (a_1 \text{ OR } \overline{a_2})$
$c_1 = a_1 \oplus (a_2 \text{ OR } \overline{a_0})$
$c_2 = a_2 \oplus (a_0 \text{ OR } \overline{a_1})$
The respective Quantum circuit is shown in figure 19. In this diagram, the first step involves computing the logical OR between a_{i+1} and a_{i+2}, which is then XORed with a_i. The diagram for the OR function is shown in Figure 1, illustrating how to perform the OR operation on two qubit numbers. This process requires three qubits, a specific number of gates (noted as 3

NOT), and one Toffoli gate to OR two qubits. To implement the complete Gamma function for a three-word state, we need a total of 12 qubits, 12 X gates, 3 OR gates, 3 OR adjoints, and 18 CNOT gates.

5.3 π function

In the π function, permutations are applied among words, and shifting occurs within each word. we have: $c = \pi(a) \leftrightarrow c_i = T_k(a_j)$
where $j = 2i \bmod 3$ and $k = (i + 1) \bmod 2$.
The ordering of Permutation and Shifting is shown in Table 7. Here i denotes index of word,

i	0	1	2
j	0	2	1
k	0	1	1

Table 7: Permutation and Shifting order of words

j denotes Permutation order and k denotes Shifting bit positions. The order of permutation among three words is [0, 2, 1], and the shifting for each word is as follows: the first word is shifted 1 bit to the left, the second word is shifted 2 bits to the left, and the third word is shifted 1 bit to the left. Quantum circuit for implementing this is shown in Figure 20 and Figure 21.

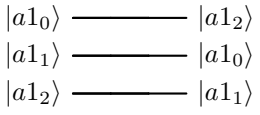


Figure 20: Shifting of Word1

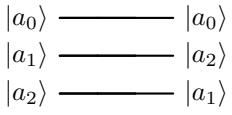


Figure 21: Permutation of Words

5.4 θ function

The transformation of state is defined as : $c_0 = a_0 \oplus a_1$, $c_1 = a_1 \oplus a_2$, $c_2 = a_2 \oplus a_0$
Since these transformations only involves XOR, can be implemented as Quantum circuit using only CNOT gates. Each register is a 2 qubit, each cmputation requires only 4 CNOT gates. Then for restoring results back into State registers 6 CNOT gates are used. Totally 18 CNOT gates are used.The respective Quantum circuit is shown in Figure 22.

5.5 σ function

The Transformation of state defined as : $c_0 = a_0 \oplus 01$, $c_1 = a_1 \oplus k$, $c_2 = a_2 \oplus b_0$
To compute c_0, we add a constant value of 01 to a_0. This can be implemented in a quantum circuit using CNOT gates and an X gate. Similarly, the input key bit is XORed with a_1 to calculate c_1, which requires only 4 CNOT gates. In the same manner, c_2 is obtained by XORing the buffer register b_0 with a_2, also requiring 4 CNOT gates. Once the calculations are complete, all results are restored back into the state registers a_0, a_1, and a_2.

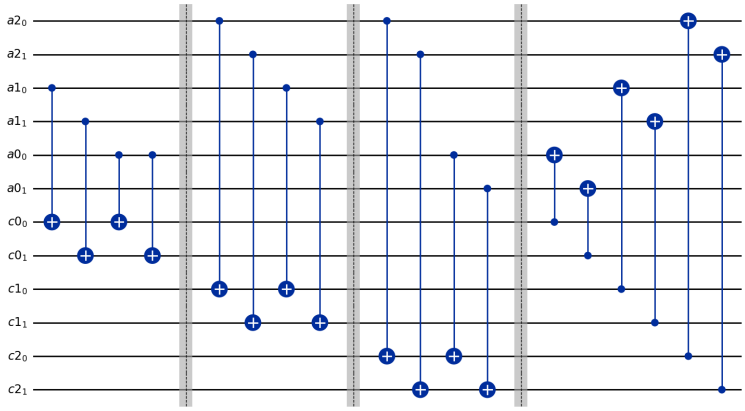


Figure 22: Quantum circuit for Theta function

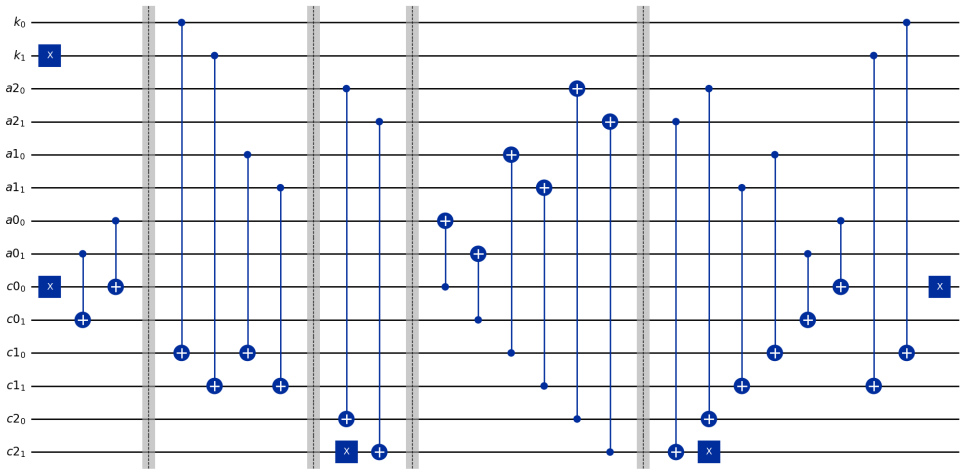


Figure 23: Quantum Circuit for Sigma function

5.6 Grover attack on Simplified Panama

Input Setting: For key searching using Grover's algorithm, we select two elements (IV1 and IV2) and generate two keystreams by utilizing the superposition of the key K with the Panama Stream cipher. First, we establish the key and the elements of the Initialization Vector (IV) that will be input into the oracle. We begin by applying the Hadamard gate to all qubits of the key K, which creates a superposition state where all values (0 to 3) can exist simultaneously. Next, we set the known elements of the IV, specifically IV1 and IV2, by applying X gates. Since the initial values of the qubits IV1 and IV2 are both 0, we apply X gates to the qubits corresponding to the values of 1. For instance, with IV1 represented as 0b01 and IV2 as 0b10, we apply the X gate to the least significant qubit of IV1 and to the second least significant qubit of IV2, respectively.

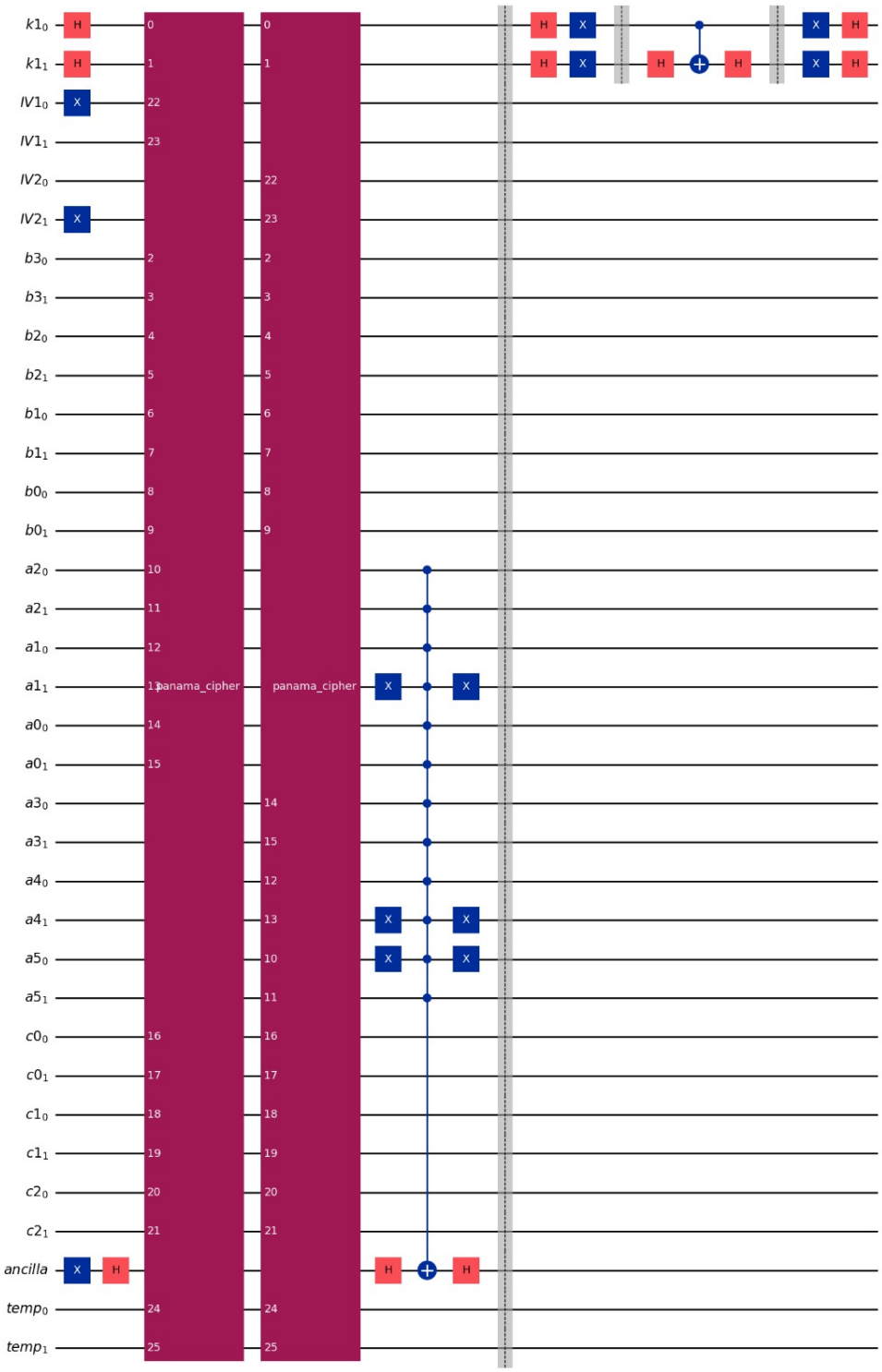


Figure 24: Grover attack on Panama

Grover's Oracle: Figure 24 depicts the Grover attack on a simplified Panama stream cipher. K_1 (2 bits) represents the key, while IV_1 and IV_2 denote the initialization vector bits. Qubits $b_3 - b_0$ form the buffer, $a_0 - a_2$ store the state and keystream for IV_1 , and $a_3 - a_5$ serve the same role for IV_2. Qubits $c_0 - c_2$, along with temporary and ancilla qubits, act as auxiliary qubits. In Figure 24, we present the design of the Grover search circuit for recovering the key corresponding to a known IV–keystream pair: (IV1; ks1) = (01, 101111) and (IV2; ks2) = (10, 100110). The generated keystreams are stored in the state array qubits (a0, a1, a2) for IV1 and (a3, a4, a5) for IV2. The known keystream values, 0b101111 and 0b100110, are initialized using X gates. An X gate is applied to the zero-valued qubit in each keystream to enable its use as a control qubit for the Controlled-Z (CZ) gate. As a result, all state array qubits are set to 1, allowing them to serve as control qubits for the CZ gate. When the generated keystreams match the known keystreams, Controlled-Z gate gate applies a phase flip to the corresponding key K. Since Grover's search identifies the key through iterative applications of the oracle and diffusion operator, the generated keystreams are reset to zero by performing the inverse of the Panama stream cipher, preparing the circuit for the next search iteration. As the key is only 2 bits long, a single Grover iteration is sufficient.

Diffusion Operator: After Grover's oracle flips the phase of the target key, the diffusion operator is applied to the key qubits to amplify its amplitude. A detailed description of the diffusion operator is provided in Section 4. Following diffusion, measurement yields the target key with high probability.

6 Conclusion

We have successfully implemented Grover's cryptanalysis on Panama cipher. This work presents a comprehensive quantum implementation of both the original Panama Cipher and a simplified version of the Panama Stream Cipher. Initially, quantum circuits were developed for each fundamental component of the cipher, which were then integrated to form the complete quantum version. The resulting quantum cipher was embedded into a black-box oracle, enabling its use within Grover's algorithm. All designs were simulated and validated using the Qiskit toolkit. As expected from Grover's algorithm, the secret key can be successfully recovered in approximately $\frac{\pi}{4}\sqrt{k}$ computational steps. Although the Panama stream cipher can theoretically be broken in $O(2^{128})$ time, this complexity doesn't make Panama cipher insecure with present computational power of our systems.

References

[1] Swamy Naidu Allu and Appala Naidu Tentu. Quantum cryptanalysis on a5/1 stream cipher. International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 14 pp. 128-137, 2022.

[2] Sarbani Sen andSoumen Bajpayee, Imon Mukherjee, Goutam Paul, and Bhupendra Singh. Quantum quest: Resource requirement of rc4 cryptanalysis under grover's lens. IEEE Calcutta Conference (CALCON), DOI: 10.1109/CALCON63337.2024.10914073, 2024.

[3] Bhagwan Bathe, Ravi Anand, and Suman Dutta. Evaluation of grover's algorithm toward quantum cryptanalysis on chacha". Quantum Information Processing (2021) 20:394, https://doi.org/10.1007/s11128-021-03322-7, 2021.

[4] Xavier Bonnetain, Maria Naya Plasencia, and Andre Schrottenloher. Quantum security analysis of aes. IACR Trans. Symmetric Cryptol., 2019.

[5] Joan Daemen and Craig Clapp. Fast hashing and stream encryption with panama. FSE'98 LNCS 1372, pp. 60–74, 1998.

[6] Martin Hell, Thomas Johansson, and Willi Meier. Grain - a stream cipher for constrained environments. http://www.ecrypt.eu.org/stream, 2005.

[7] Martin Hell, Thomas Johansson, Willi Meier, Jonathan Sonnerup, and Hirotaka Yoshida. Grain-128 aead - a lightweight aead stream cipher. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/grain-128aead-spec-round2.pdf, 2017.

[8] Brandon Langenberg, Hai Pham, and Rainer Steinwandt. Reducing the cost of implementing aes as a quantum circuit. IACR Cryptol. ePrint Arch., 2019:854, 2019.

[9] Grover L.K. A fast quantum mechanical algorithm for database search. In:Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA,May 22-24, 1996,pp. 212–219. ACM, New York, 1996.

[10] Almazrooie M, Samsudin A, Abdullah R, and Mutter K.N. Quantum reversible circuit of aes-128. Quantum information processing 17, 1–30, 2018.

[11] Grassl M, Langenberg B, Roetteler M, and Steinwandt R. Applying grover's algorithm to aes:quantum resource estimates. In: Takagi, T. (ed.) Post-quantum Cryptography. PQCrypto. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer, Cham, 2016.

[12] Surajit Mandal, Ravi Anand, Mostafizar Rahman, Santanu Sarkar, and Takanori Isobe. Implementing grover's on aes-based aead schemes. Scientifc Reports 14:21105,https://doi.org/10.1038/s41598-024-69188-8, (2024).

[13] M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information. Cambridge University Press, 2010.

[14] NIST. Submission requirements and evaluation criteria for the postquantum cryptography standardization process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf, 2016.

[15] NIST. Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197, November 2001.

[16] Shor P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509, 1997.

[17] IBM Quantum. Qiskit documentation. available at https://qiskit.org/documentation/, accessed July 15, 2025, 2025.

[18] Anand R, Maitra S, Maitra A, Mukherjee C.S, and Mukhopadhyay S. Resource estimation of grovers kind quantum cryptanalysis against fsr based symmetric ciphers. Cryptology ePrint Archive, Report 2020/1438,https://eprint.iacr.org/2020/1438, 2020.