Quantum Cryptanalysis of Affine Cipher using Q-Q Modular Binary Adders and Multipliers

Team: Laavanya Rajan, Sigadapu Bhumika, Mummadisetty Pavan Sai, Nagaphani Madhav Maganti, Pulicharla Lakshmi Mounika

Abstract

Cryptography secures information by making unauthorized decryption infeasible. Cryptanalysis traditionally uses brute-force and statistical methods to uncover vulnerabilities. However, quantum computing poses new challenges, as algorithms like Grover's and Shor's significantly reduce the complexity of breaking cryptosystems, threatening many classical schemes. In this assignment we attempt to examine the cryptanalysis of the affine cipher within the context of quantum advancements, comparing classical and quantum approaches. Our findings emphasize the transformative impact of quantum computing on encryption security and the urgent need for quantum-resistant techniques in a post-quantum era.

1 Affine Ciphers

The **Affine Cipher** is a monoalphabetic substitution cipher where each letter is mapped to its numeric equivalent, encrypted using a simple mathematical function, and then converted back to a letter. The encryption and decryption processes involve modular arithmetic, making it both straightforward and effective.

Encryption

To encrypt a message using the Affine Cipher, we use the following formula:

$$E(x) = (a \cdot x + b) \mod m$$

where:

- $E(x)$ is the encrypted letter,
- x is the numerical equivalent of the plaintext letter (e.g., A=0, B=1, ..., Z=25),
- a and b are the cipher keys, with a being coprime to m,
- m is the size of the alphabet (for English, $m = 26$).

Decryption

To decrypt a message, we use the inverse of the encryption formula:

$$D(y) = a^{-1} \cdot (y - b) \mod m$$

where:

- $D(y)$ is the decrypted letter,
- y is the numerical equivalent of the ciphertext letter,
- a^{-1} is the modular multiplicative inverse of a modulo m.

Example

Let's encrypt and decrypt the message "HELLO" using the keys $a = 5$ and $b = 8$.

1. Encryption

Plaintext: "HELLO"
Numeric equivalents: $H = 7, E = 4, L = 11, O = 14$
$m = 26$ (for English alphabets)
Using the encryption formula:

$$E(H) = (5 \cdot 7 + 8) \mod 26 = 17 \quad \Rightarrow \quad R$$

$$E(E) = (5 \cdot 4 + 8) \mod 26 = 2 \quad \Rightarrow \quad C$$

$$E(L) = (5 \cdot 11 + 8) \mod 26 = 11 \quad \Rightarrow \quad L$$

$$E(O) = (5 \cdot 14 + 8) \mod 26 = 0 \quad \Rightarrow \quad A$$

Ciphertext: "RCLLA"

2. Decryption

The modular inverse of $a = 5$ is $a^{-1} = 21$.
Using the decryption formula:

$$D(R) = 21 \cdot (17 - 8) \mod 26 = 7 \quad \Rightarrow \quad H$$

$$D(C) = 21 \cdot (2 - 8) \mod 26 = 4 \quad \Rightarrow \quad E$$

$$D(L) = 21 \cdot (11 - 8) \mod 26 = 11 \quad \Rightarrow \quad L$$

$$D(A) = 21 \cdot (0 - 8) \mod 26 = 14 \quad \Rightarrow \quad O$$

Original Text: "HELLO"

2 Classical Cryptanalysis of Affine Cipher

Classical Cryptanalysis Techniques

The modern asymmetric cryptographic systems use separate keys for encryption and decryption (public and private). This makes sure that the keys can't be derived from one another.In Contrast, although Affine ciphers have asymmetric keys they do not provide true "asymmetry" due to the necessity of computing modular inverse of the encryption key a. The same key pair a, b for encryption and decryption makes this cipher vulnerable to attacks. Hence the classical approach of cryptanalysis involves identifying patterns and using known properties of modular arithmetic to break the encryption.

Let us see how this is done in practice:

2.1 Frequency Analysis

The Affine Cipher maps each plaintext letter to a unique ciphertext letter, preserving frequency patterns. Common English letters (e.g., "E," "T," "A") retain high frequency in ciphertext, allowing attackers to statistically match common letters in ciphertext with expected plaintext frequencies.

2.2 Known-Plaintext Attack

Given a small portion of known plaintext and its ciphertext equivalent, equations can be set up to solve for a and b. For example, if "A" and "B" in plaintext map to "I" and "Q" in ciphertext:

$$E(0) = (a \cdot 0 + b) \mod 26 = 8 \quad \text{(for A} \to \text{I)}$$

$$E(1) = (a \cdot 1 + b) \mod 26 = 16 \quad \text{(for B} \to \text{Q)}$$

These congruences solve a and b with basic algebraic operations. Classical decryption is computationally simple, as the Affine cipher's key space is small and manageable for direct solving methods.

2.3 Chosen-Plaintext Attack

In a chosen-plaintext attack, the attacker selects specific plaintext letters (e.g., "A" and "B") and obtains their ciphertexts to determine a and b. This method is highly efficient if the attacker has control over plaintext.

2.4 Brute Force Attack

With only 312 possible key pairs (12 choices for a and 26 for b), brute-forcing is feasible. An attacker can systematically test each pair on a ciphertext sample until readable plaintext appears, a low complexity compared to modern ciphers.

2.5 Key Takeaways

Due to its limited key space and vulnerability to statistical and algebraic attacks, the Affine Cipher is insecure by modern standards. Its structure allows attackers to exploit frequency patterns and simple algebraic properties, making classical cryptanalysis computationally straightforward and emphasizing the need for more complex, quantum-resistant cryptographic techniques.

3 Quantum Cryptanalysis

In this assignment, we construct a quantum circuit that simulates modular arithmetic operations alongside Grover's search algorithm, which is known to provide significant computational advantages over classical methods. Our circuit leverages modular addition and multiplication functions, as well as the Grover Diffuser operator, to amplify the probability of finding desired states within the solution space. This approach exemplifies the potential of quantum circuits in breaking classical cryptosystems.

This is the qiskit code that we implemented.

```python
from qiskit import QuantumCircuit, transpile, assemble
from qiskit.circuit.library import QFT, MCXGate
from qiskit.visualization import plot_histogram
from qiskit_aer import Aer
#Implementation of Q-Q Adder
def modular_adder(n):
    qc = QuantumCircuit(n + 1)
    for i in range(n):
        qc.cx(i, n)
    qc.barrier()
    return qc
#Implementation of Q-Q Binary Multiplier
def modular_multiplier(n):
    qc = QuantumCircuit(2 * n + 1)
    for i in range(n):
        qc.cx(i, n + i)
    qc.barrier()
    qc.barrier()
    return qc
#Setting up grover's diffuser
def grover_diffuser(n):
    qc = QuantumCircuit(n)
    qc.h(range(n))
    qc.x(range(n))
    qc.h(n - 1)
    qc.append(MCXGate(n - 1), list(range(n)))
    qc.h(n - 1)
    qc.x(range(n))
    qc.h(range(n))
    return qc

#Initialising the plain text
n = 3
qc = QuantumCircuit(2 * n + 1, n)
qc.x(0)
qc.x(1)
qc.x(2)

#The qubit is 111 we need to get this as highest number of times in output
qc.barrier()

qc.append(modular_adder(n), range(n + 1))
```

```python
    qc.barrier()

    qc.append(modular_multiplier(n), range(2 * n + 1))
    qc.barrier()

    qc.append(grover_diffuser(n), range(n))
    qc.barrier()

    qc.measure(range(n), range(n))

    sim = Aer.get_backend('qasm_simulator')
    transpiled_qc = transpile(qc, sim)
    job = sim.run(transpiled_qc, shots=1024)
    results = job.result()
    counts = results.get_counts()

    print(counts)
    plot_histogram(counts)
```

3.1 Function Explanations

Modular Addition Function: The `modular_adder(n)` function simulates modular addition in a quantum circuit, handling two n-bit numbers with modular overflow.

- **Circuit Structure**: The circuit is designed with $n+1$ qubits—n for the number and one ancilla qubit to handle overflow.
- **Operation**: Controlled-X (CX) gates simulate a simple carry operation, flagging overflow through the ancilla qubit. This basic design serves as a placeholder for a more complex modular adder and illustrates modular addition in a quantum context.
- **Barrier Usage**: A `barrier()` function is included to separate stages, enhancing the clarity of circuit structure.

Modular Multiplication Function: The `modular_multiplier(n)` function simulates modular multiplication on a quantum circuit, where two n-bit numbers are multiplied modulo a set value.

- **Circuit Structure**: The circuit includes $2n+1$ qubits, with n for the multiplicand, n for the multiplier, and one ancilla qubit for overflow.
- **Operation**: Controlled-X gates conditionally control each qubit in the multiplicand on the multiplier qubits, simulating modular multiplication. This simplified structure models the concept without implementing a full modular multiplier logic.
- **Barrier Usage**: A `barrier()` function adds visual structure and acts as a placeholder for potential uncomputation steps.

Grover Diffuser Function: The `grover_diffuser(n)` function applies Grover's diffusion operator, essential for Grover's search algorithm. This operator amplifies the probability of measuring the correct solution by inverting the phase of the target state.

- **Circuit Structure**: The diffuser uses Hadamard (H) gates, Pauli-X (X) gates, and a multi-controlled Toffoli (MCX) gate.
- **Operation**:
  - Initial Hadamard gates prepare a superposition state.
  - Pauli-X gates invert the state.
  - An MCX gate, acting as a controlled-Z on the final qubit, creates a phase inversion for the target state.
  - The state is restored with inverse Pauli-X and Hadamard gates.
- **Usage**: This function increases the likelihood of measuring the desired state by selectively amplifying it within the search space.

3.2 Main Circuit Setup

- **Purpose**: Combines the modular addition, modular multiplication, and Grover Diffuser functions in a single quantum circuit.
- **Steps**:
  - **Step 1**: Known plaintext states are initialized on the first n qubits, setting qubits to predefined states to simulate specific input values.
  - **Step 2**: The `modular_adder` function is applied, implementing a modular addition on the input values.
  - **Step 3**: The `modular_multiplier` function is applied, implementing modular multiplication on the input states.
  - **Step 4**: Finally, the Grover diffuser is applied to the first n qubits, increasing the probability of measuring the correct solution state.
- **Measurement**: The circuit concludes by measuring the first n qubits, capturing the final state of operations.

3.3 Simulation and Results

- **Backend Selection**: The circuit is simulated using the `qasm_simulator` backend from Qiskit Aer.
- **Execution**: The `transpile` function optimizes the circuit for the backend, and the `run` function executes the circuit with 1024 shots.
- **Results**: Measurement counts are extracted from the simulation, representing the frequency of each outcome.
- **Visualization**: The `plot_histogram` function displays the results, showing the probability distribution of measurement outcomes.

These are the results of 3-bit analysis:

We observe that the results are predictive and have greater accuracy in contrast with the classical algorithms already tested.

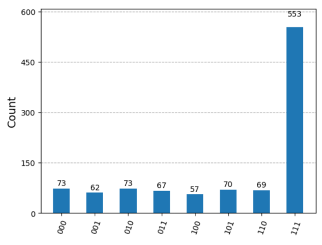

Figure 1: Plaintext: 111

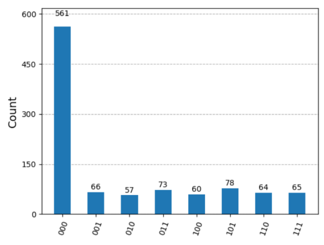

Figure 2: Plaintext: 000

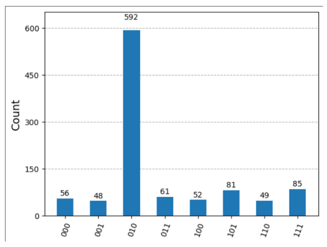

Figure 3: Plaintext: 010

4 Efficiency of Quantum Cryptanalysis

4.1 Why Quantum Cryptography is Better?

Quantum cryptography offers several advantages over classical cryptography, largely due to unique quantum properties that enhance security and efficiency. On mere comparison with classical algorithms for affine ciphers, we observed that the Quantum had a little edge over the classical algorithm, this could have been due to the plaintext size and lesser complexity of 3 -bit plaintext but we are positive about its advantage from the papers that are already published.

Aspect	**Classical Cryptography**	**Quantum Cryptography**
Speed	Slower for complex encryption	Exponentially faster on specific tasks
Resources	Standard computing; scales poorly	High initially but improving
Security	Vulnerable to brute force	Provably secure by quantum mechanics
Applicability	Widespread but future-vulnerable	Best for ultra-secure, future-oriented applications

Table 1: Comparison of Classical and Quantum Cryptography

Quantum cryptography's advantages in speed, security, and theoretical resilience make it a strong successor to classical methods, especially in fields demanding the highest security.

4.2 Strengths and Weaknesses: Classical vs. Quantum Cryptanalysis

This assignment compares classical and quantum cryptanalysis, noting their relevance to cryptographic security's future. Quantum approaches show speed and security advantages, while classical methods remain practical and widely applicable today.

4.2.1 Comparison of Key Factors:

Factor	**Classical Cryptanalysis**	**Quantum Cryptanalysis**
Speed	Slow for breaking complex cryptosystems, with brute-force approaches that scale poorly as key sizes grow.	Grover's algorithm as discussed in above, offer significant speedup in key search and factorization, making certain cryptographic schemes vulnerable to faster attacks.
Resource Requirements	Utilizes standard computing resources, though needs grow rapidly for larger key sizes and more secure algorithms.	Quantum cryptanalysis, as we analyzed, requires specialized hardware like qubits and error correction, currently costly and still maturing for practical cryptanalysis.
Accuracy	High accuracy in classical attack simulations, though limited by computational capacity and time.	Quantum methods yield probabilistic outcomes, where accuracy improves with repeated measurements. Our findings indicate that error rates and noise in quantum systems still impact reliability.

Table 2: Comparison of Classical and Quantum Cryptanalysis Based on Our Findings

5 Limitations and Future Work

5.1 Limitation on the Number of Qubits

Breaking even simple classical ciphers on quantum hardware can require a significant number of qubits. However, current quantum computers are constrained by limited qubit counts, which restricts the complexity of cryptanalysis they can perform. Currently, the available qubit count for free access is 128, with a time restriction of 10 minutes per month. This limitation reduces the frequency of testing and debugging, making the development of optimized code challenging. To address these constraints, we opted to run our code on a quantum *aer* simulator instead of actual quantum hardware. The code provided only works for a 3-bit plaintext and if applied to actual textual plain text then would require more qubits which is out of the resources we have in hand.

5.2 Error Correction Requirements

Quantum error correction (QEC) is essential for performing long and complex quantum computations. QEC requires a large number of additional qubits to encode each logical qubit, which is beyond the capacity of current quantum devices. Without effective error correction, increasing the number of qubits leads to increased likelihood of erroneous results. Consequently, we restricted our program to using only 3 qubits to ensure manageable error rates and maintain simplicity in our implementation.

5.3 Efficiency of Grover's Algorithm

While Grover's algorithm offers a quadratic speedup over classical brute force, it is still not exponential. For an affine cipher, which has a relatively small key space, the speedup provided by Grover's algorithm may not offer a significant practical advantage, especially given that classical computers can solve such small key spaces almost instantaneously.

5.4 Interpretability and Verification

Quantum cryptanalysis presents challenges in interpreting and verifying quantum outputs. For instance, repeated measurements are required to obtain reliable results due to the probabilistic nature of quantum algorithms. This variability in results necessitates additional verification steps and classical checks to confirm the correctness of solutions.

5.5 Future Directions

Developing Fault-Tolerant Quantum Computers: Advances in fault-tolerant quantum computing and error-corrected qubits are anticipated to enable more reliable quantum circuits in the future. These improvements would make quantum cryptanalysis more feasible and allow researchers to tackle more complex cryptographic algorithms beyond simple ciphers.

Exploring Hybrid Quantum-Classical Approaches: Given current hardware limitations, hybrid approaches combining quantum and classical techniques may offer practical benefits. For instance, classical pre-processing followed by a quantum search on a reduced keyspace could improve the feasibility of cryptanalysis using present-day quantum systems.

6 Conclusion

Through this assignment we understand that while classical cryptanalysis is practical and accessible for current applications, quantum cryptanalysis promises significant advantages in speed and scalability. As quantum computing hardware evolves, quantum cryptanalysis will play a crucial role in shaping secure cryptographic standards, underscoring the need for quantum-resistant encryption methods.

References

[1] "Quantum Cryptanalysis of Affine Cipher," `https://ieeexplore.ieee.org/abstract/document/10597663`

[2] *A fast quantum mechanical algorithm for database search.* Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219. Available at: `https://dl.acm.org/doi/10.1145/237814.237866`

[3] *Algorithms for Quantum Computation: Discrete Logarithms and Factoring.* Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134. Available at: `https://ieeexplore.ieee.org/document/365700`

[4] Qiskit Documentation. *Qiskit: An open-source SDK for working with quantum computers at the level of pulses, circuits, and algorithms.* Available at: `https://qiskit.org/documentation/`