Quantum Grover Attack on the Simplified-AES

Mishal Almazrooie
School of Computer Sciences
Universiti Sains Malaysia
Pulau Pinang, Malaysia
mishal.usm@gmail.com

Rosni Abdullah
School of Computer Sciences
Universiti Sains Malaysia
Pulau Pinang, Malaysia
rosni@usm.my

Azman Samsudin
School of Computer Sciences
Universiti Sains Malaysia
Pulau Pinang, Malaysia
azman.samsudin@usm.my

Kussay N. Mutter
School of Physics,
Universiti Sains Malaysia, USM
Pulau Pinang, Malaysia
knm@usm.my

ABSTRACT
In this work, a quantum design for the Simplified-Advanced Encryption Standard (S-AES) algorithm is presented. Also, a quantum Grover attack is modeled on the proposed quantum S-AES. First, quantum circuits for the main components of S-AES in the finite field $\mathbb{F}_2[x]/(x^4 + x + 1)$, are constructed. Then, the constructed circuits are put together to form a quantum version of S-AES. A **C-NOT** synthesis is used to decompose some of the functions to reduce the number of the needed qubits. The quantum S-AES is integrated into a black-box queried by Grover's algorithm. A new approach is proposed to uniquely recover the secret key when Grover attack is applied. The entire work is simulated and tested on a quantum mechanics simulator. The complexity analysis shows that a block cipher can be designed as a quantum circuit with a polynomial cost. In addition, the secret key is recovered in quadratic speedup as promised by Grover's algorithm.

CCS Concept
• Security and privacy→ Block and stream ciphers • Security and privacy→ Cryptanalysis and other attacks • Computing methodologies→ Quantum mechanic simulation

Keywords
Symmetric cryptography, Quantum cryptanalysis, Quantum simulation, Grover attack, Block cipher

1. INTRODUCTION
Factorizing large integer *N*, which is a product of two primes is considered as a hard mathematical problem. There is no efficient classical algorithm that runs on a classical computer that can solve the factorizing problem in a polynomial time. Hence, the hardness of such factorizing problem is employed in the asymmetric cryptographic algorithm such as RSA [1]. However, the security strength of RSA is jeopardized with the introduction of the Shor's quantum algorithm [2]. Moreover, the other alternative algorithms to RSA which are based on discrete logarithms problem [3], could also be broken in a polynomial time by using the Shor's algorithm. Subsequently, a research area, post-quantum cryptography, was introduced to study the effect of quantum mechanics on cryptography [4].

In cryptography, the asymmetric cryptographic algorithms are mainly used to exchange the secret encryption key which is used by a symmetric cryptographic algorithm. The impact of quantum computing on asymmetric cryptography is very clear in the present of quantum computer. On the other hand, the impact of quantum computing on symmetric cryptography is not as clear. One of the first papers that discussed the impact of the quantum computing on symmetric cryptography is the work by Akihiroe et al. [5]. In that work, Akihiroe et al. discussed a quantum attack that uses Grover's algorithm [6] to recover a secret key in $O(2^{n/2})$ computational steps. Related to Akihiroe's work, a quantum version of the classical Meet-in-the-Middle attack based on the quantum Ambainis algorithm [7] for solving Element Distinctness Problem was proposed by Kaplen [8]. Such attack removes the possibility to increase the key security by doubling the key size. Kaplan has carried out her study at a more abstract level by assuming that a block cipher is merely a collection of random permutations functions $f_k : \{M\} \rightarrow \{M\}$ where k is the potential key, and $k \in \{0, 1\}^n$ such that n is the key size and M is the plaintext block size. The main corollary by Kaplan is that a pair of distinct keys (k_1, k_2) used in an iterative block cipher can be extracted quantumly in $O(2^{2n/3})$ where n is the size of one key.

A note on quantum related key attack was written by Roetteler et al. [9]. Although the proposed quantum related key attack in that paper is unlikely to pose a real threat in practical because of the difficulty of querying a superposition on the related keys (as stated by Roetteler et al.), but such an effort gives clues that quantum attack on the symmetric cryptography could be as harmful as the one found on the asymmetric cryptography. Roetteler et al. modeled the related key attack as a hidden subgroup problem which could be solved in a polynomial time by using the quantum Simon's algorithm [10]. The authors reported that if the related key can be queried in a superposition then the secret key could be recovered on $O(\log N)$. Similarly to

Kaplen [7], one of the Roetteler's assumptions is that a block cipher has to be implemented efficiently as a quantum circuit. Therefore, for a quantum adversary to conduct a quantum attack against a symmetric cryptographic algorithm, the adversary must

ICSCA 2018, February 8–10, 2018, Kuantan, Malaysia

ACM ISBN 978-1-4503-5414-1/18/02…$15.00
https://doi.org/10.1145/3185089.3185122

be able to implement the targeted block cipher as a quantum circuit.

In this work, we show that the quantum design of a block cipher is not an obstacle that could prevent any possible quantum threat. Moreover, we show that a quantum circuit of a polynomial number of basic quantum gates can be designed for a block cipher by using the simplified version of the Advanced Encryption Standard (AES) [11], as a case study.

This paper is organized as follows: Sections 2 and 3 review the Simplified-AES (S-AES) cryptosystem and the quantum Grover's algorithm, respectively. The proposed quantum version of the S-AES and the complexity analysis are presented in Section 4. Grover attacks are modeled and discussed in Section 5. Quantum mechanics simulation and its results are presented in Section 6. The last Section provides the conclusion and future work.

2. THE SIMPLIFIED-AES

The simplified version of AES (S-AES) is a small variant of the AES [12]. It has the exact similar structure as AES with small parameters. The plaintext block size in S-AES is 16 bits and the key size is also 16 bits. S-AES is a Nibble-oriented algorithm compared to a Byte-oriented algorithm in AES (one Nibble is four bits). The state array of S-AES is shown in Eq. (1), where b_i is an input bit and S_i is one Nibble. S-AES have been used for the purposes of classical cryptanalysis such as in [13–15]. In this work, a quantum design of S-AES is proposed and then a straight forward quantum cryptanalysis method namely quantum exhaustive search for the secret key by using Grover's algorithm is applied.

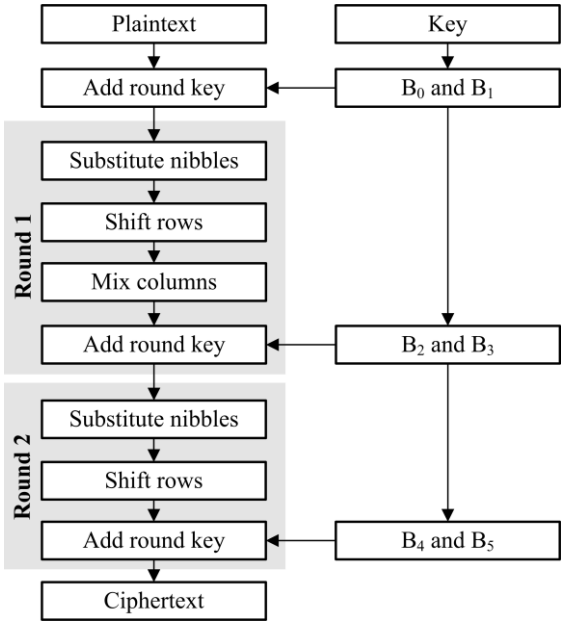

Figure 1. The block diagram of the Simplifed-AES

$$\overbrace{b_0b_1}^{S_0}\overbrace{b_2b_3}^{S_1}\overbrace{b_4b_5}^{S_2}\overbrace{b_6b_7}^{S_3}\overbrace{b_8b_9}\overbrace{b_{10}b_{11}}\overbrace{b_{12}b_{13}}\overbrace{b_{14}b_{15}} \Rightarrow State = \begin{pmatrix} S_0 & S_2 \\ S_1 & S_3 \end{pmatrix} \quad (1)$$

Figure 1 shows the block diagram of S-AES. Each step shown in the block diagram is discussed in more details during the design phase of each in Section 4. As illustrated in Figure 1, the S-AES consists of two rounds. The main four steps of S-AES are: Nibbles substitution (**SubNibbles**), rows shifting (**ShiftRows**), columns mixing (**MixColumns**), and adding round key (**AddKey**). Similarly to AES, **MixColumns** is excluded from the last round in S-AES. The subkeys generation algorithm resamples also the one in AES and it will be discussed in more details during the design phase in Section 4.

3. GROVER'S ALGORITHM

In quantum information, the quantum bit (qubit) is characterized by two orthogonal states $|0\rangle$ and $|1\rangle$. Compared to a classical bit, the qubit can be in a superposition state: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ where α and $\beta \in C$, which representing the amplitude probability such that $|\alpha|^2 + |\beta|^2 = 1$. Those states of the qubit can be expressed as vectors in two-dimensional Hilbert space $\mathcal{H}^2$ as:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}, |\psi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix} \quad (2)$$

The reader is referred to [16–19] for more information about the definitions of quantum gates and reversible circuit. The quantum search algorithm was discovered by Lov Grover [6] and named after him. Grover's search algorithm and Shor's period finding algorithm [2], along with their extensions, constitute the masterpiece algorithms of quantum computations [17].

Problem Definition: Given an unstructured database of N elements, find the element $a \in N$. This can be modeled as a function f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$, where the space $N = 2^n$, for any $x \in \{0, 1\}^n$

$$f(x) = \begin{cases} 1 & if\ x = a\ (a\ solution) \\ 0 & otherwise\ (not\ a\ solution) \end{cases} \quad (3)$$

When the database is unstructured, the element *'a'* can be found among N random elements (by assuming the uniform probability distribution) with probability of $1/N$. Therefore, on a classical computer, $O(N) = O(2^n)$ steps are needed to find *'a'*.

On the other hand, quantum computing using Grover's algorithm, the element *'a'* can be found with a significant speedup that is quadratically faster than that on any classical computer. The search through an unstructured database can be accomplished within $O(\sqrt{N})$ computational steps. Grover's algorithm is shown in Algorithm 1.

Additional discussions with circuit illustration on the Oracle and "the inversion about the mean" are conducted in Section 5. All the steps of Grover's algorithm listed in Algorithm 1, can be expressed as: $((2\,|\psi\rangle\,\langle\psi\,|\,-\,I)\,O)^R$, where O is the Oracle and R is the number of iterations.

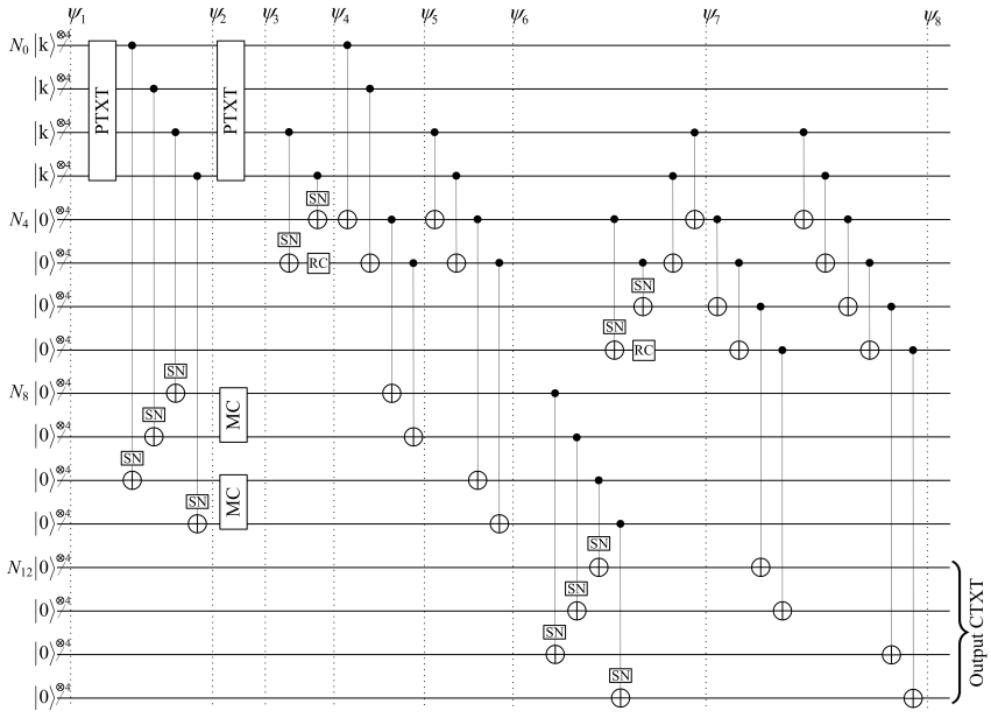

Figure 2. The proposed quantum circuit of the Simplified-AES. SN refers to SubNibbles circuit shown in Figure 5, MC refers to MixColumns circuit shown in Figure 6, and RC stands for Round Constant.

Algorithm 1: Grover's algorithm

Input: An unstructured set $N = \{a_1, a_2, \ldots, a_n\}$

Output: $a_i \in N$

Step 1: Initialization of the quantum register:

all the qubits $x^{\otimes n}$ to $|0\rangle$ state and the oracle qubit q to $|1\rangle$ state:

$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$

Step 2: Put the register in an uniformly distributed superposition:

apply **H** Hadamard gate::

$|\psi_1\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{N-1} |i\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$

Step 3: Apply Grover iterations:

for $2^{n/2}$ *times*, **do**

a − **Apply the oracle**:

$|x\rangle \xrightarrow{o} (-1)^{f(x)} |x\rangle, \qquad f(x)\ as\ in\ eq.(3)$

b – Perform Grover operator (inversion about the mean):

i. Apply $H^{\otimes n}$

ii. Conditionally shift phase

iii. Apply $H^{\otimes n}$

Step 4: Measure the quantum register

4. QUANTUM S-AES

The block diagram of the proposed quantum circuit of S-AES is shown in Figure 2. The entire design is mainly structured with a consideration of utilizing the lowest possible number of qubits even though that could lead to a growing number quantum gates needed. The main functions of S-AES are discussed in more details in the following subsections. First, the initial state of the classical S-AES which is arranged in a 2 × 2 matrix as shown in Eq. (1), it is rearranged in a column vector of 16 qubits as in Eq. (4). Each 4 qubits ($b_{i.0\text{-}3}$) constitute one quantum Nibble (N_i), and each two Nibbles ($N_{i.0\text{-}1}$) constitute one quantum Byte (B_i). Note that, since the Nibbles in S-AES resemble Bytes in AES, therefore, the Bytes orientation in S-AES resembles the Word (32 bits) in AES which is considered in the Key generation algorithm.

$$State = \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}, where\ B_i = \begin{pmatrix} N_{i,0} \\ N_{i,1} \end{pmatrix}, where\ N_i = \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \\ b_{i,3} \end{pmatrix}. \qquad (4)$$

4.1 Nibbles Substitution

The number of qubits needed to implement the S-AES algorithm as a quantum circuit is 64 qubits and 8 ancilla qubits for the substitution box (S-Box). The qubits 0–15 (N_0 - N_3) in Figure 2 are dedicated for the main key. The plaintext is implemented by using a set of **Pauli-X** gates which implemented directly on the qubits lines of the main key as shown at $|\psi_1\rangle$, and the number of **Pauli-X** gates corresponds to the number of ones in the plaintext. This step is the first key mixing of S-AES which known as adding whitening key. After that, the first encryption round starts with Nibbles substitution by using the **SubNibbles** function.

The **SubNibbles** is the step which provides the non-linear diffusion in S-AES. It takes a Nibble (a) in the form of a 4-bit column vector as an input and substitutes it with a different Nibble (b) according the following steps: (1)

Calculate the multiplicative inverse a-1 of the input a in

GF(2^4),

(2) Let $c = a^{-1}$, then, the output Nibble b from **SubNibbles** function is calculated as the following affine transformation:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (5)$$

The well-known methods of calculating the multiplicative inverse in finite fields [20], are either based on Euclidean algorithm such as in [21] or based on Fermat's little theorem as found in Itoh-Tsujii algorithm [22]. For AES, Itoh-Tsujii algorithm is the most efficient method to calculate the multiplicative inverse in the finite field GF(2^8). However, in S-AES, we have found that applying Fermat's inversion approach (square-multiply algorithm) can lead to the same cost in terms of qubits and quantum gates. Hence, square-multiply algorithm is adopted in this work to calculate the multiplicative inverse in the finite field GF(2^4) of S-AES. The finite field of S-AES is GF(2^4) with irreducible polynomial $x^4 + x + 1$ which can be written as $\mathbb{F}_2[x]/(x^4 + x + 1)$, The multiplicative inverse of an element $a \in \mathbb{F}_2[x]/(x^4 + x + 1)$, is computed by using square-multiply approach according to the following equation:

$$a^{-1} = a^{2^4-2} \Leftrightarrow a^{14} = a^2 \times (a^2)^2 \times ((a^2)^2)^2 \quad (6)$$

In Eq. (6), it can be seen that two multiplication and three square operations are needed to compute the multiplicative inverse. Hence, a multiplier circuit for S-AES in $\mathbb{F}_2[x]/(x^4 + x + 1)$, is first to be constructed. A quantum multiplier approach by Cheung et al. [23], which is derived from the classical approach in [24] is adopted in this work. Fortunately, the example illustrated in Cheung et al.'s work is in the field $\mathbb{F}_2[x]/(x^4 + x + 1)$, which is the same field used by S-AES. The multiplier circuit of S-AES is shown in Figure 3.

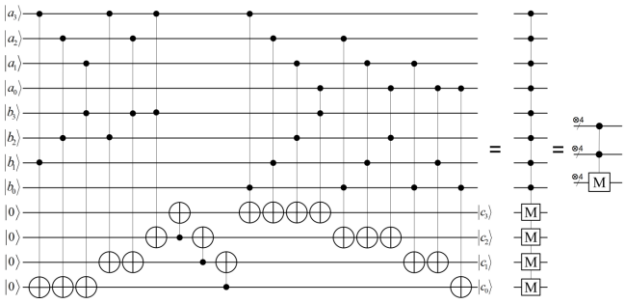

Figure 3. The multiplier quantum circuit of the finite field $\mathbb{F}_2[x]/(x^4 + x + 1)$, **The inputs to the circuit are a vector** $\boldsymbol{a} = \{a_0, a_1, a_2, a_3\}$ **and a vector** $\boldsymbol{b} = \{b_0, b_1, b_2, b_3\}$**. The output is a vector** $\boldsymbol{c} = \{c_0, c_1, c_2, c_3\}$ **such that** $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$**.**

After that, the squarer needed in Eq. (6) to calculate the multiplicative inverse, is derived from the previous calculations of the multiplier. Let vectors $\boldsymbol{a}$ and $\tilde{\boldsymbol{a}} \in \mathbb{F}_2[x]/(x^4 + x + 1)$, and let $\tilde{\boldsymbol{a}} = \boldsymbol{a} \times \boldsymbol{a}$, then $\tilde{\boldsymbol{a}}$ can be calculated according to the following:

$$\begin{pmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \tilde{a}_2 \\ \tilde{a}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \quad (7)$$

Classically, the squarer matrix shown in Eq. (7) can be implemented easily using a set of **XOR** gates. In quantum, implementing the matrix in Eq. (7) using a set of **C-NOT** gates cannot be accomplished without using extra auxiliary qubits because of the dependency among the qubits in the matrix. Thus, to remove the dependency between the qubits, **C-NOT** synthesis algorithms such as in [25] and [26] are used. In this work, the proposed **C-NOT** synthesis algorithm in [26], is adopted to decompose the matrix in Eq. (7). Eventually, the squarer circuit is shown in Figure 4. The three **C-NOT** gates at the middle of the circuit can be omitted since they represent a swapping gates set which swaps qubit 1 and qubit 2 but the design has to be readjusted accordingly then.

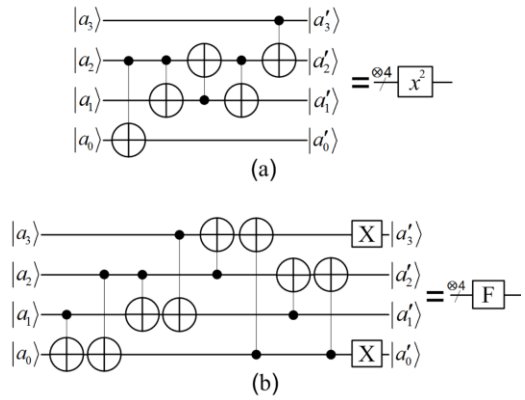

Figure 4. (a) The squarer circuit of the finite field $\mathbb{F}_2[x]/(x^4 + x + 1)$. **(b) The circuit of the affine transformation of S-AES shown in Eq. (5).**

The last phase in **SubNibbles** is to implement the matrix of the affine transformation in Eq. (5). Again, the affine matrix is first to be decomposed so it can be implemented without the need of extra qubits. Hence the **C-NOT** synthesis is again used. Finally, the vector to be XOR-ed in the affine transformation can be implemented by using two **Pauli-X** gates (see Figure 4). The complete circuit of **SubNibbles** is shown in Figure 5.

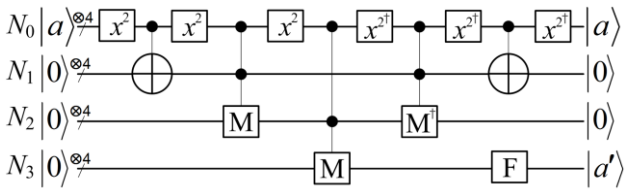

Figure 5. The complete SubNibbles circuit of S-AES.

The Nibble N_0 in Figure 5, is the input Nibble into the **SubNibbles** circuit and N_3 is the output substituted Nibble calculated by **SubNibbles**. Whereas, the other remaining eight qubits (N_1 and N_2) are ancilla qubits which are always to be cleared with every call to **SubNibbles** such that they can be reused again. Hence, the reversing process takes place in order to enable the usage of the eight ancilla qubits when the **SubNibbles** function is concurrently called.

4.2 Columns Mixing

After the Nibbles substitution, shifting the rows (**ShiftRows**) takes place in S-AES algorithm as shown in Figure 1. The **ShiftRows** function is implemented for free. At $|\psi_2\rangle$ in Figure 2, after substituting the Nibble N_0 by using **SubNibbles**, the output is placed on qubits of N_{10} and the output of N_2 is placed on qubits of N_8. Then, the third step in S-AES algorithm which is mixing the columns (**MixColumns**) takes place.

The AES works on the polynomial ring $\mathbb{F}_2[x]/(x^8+x^4+x^3+x+1)$ and the **MixColumns** in AES treats one Word (a column in its state matrix) as an element of polynomials of coefficients in $GF(2^8)$ working on the ring $\mathbb{F}_{2^8}[x]/(x^4+1)$. Similarly, the ring at which the S-AES works is $\mathbb{F}_2[x]/(x^4+x+1)$, and the **MixColumns** treats one Byte (one column which is composed of two Nibbles in the state matrix in Eq. (1)), as an element in $GF(2^4)$ working on the ring $\mathbb{F}_{2^4}[x]/(x^2+1)$. The **MixColumns** transformation in S-AES is expressed as the following matrix multiplication:

$$\begin{pmatrix}\tilde{S}_0\\ \tilde{S}_1\end{pmatrix} = \begin{pmatrix}1_{16} & 4_{16}\\ 4_{16} & 1_{16}\end{pmatrix}\cdot\begin{pmatrix}S_0\\ S_1\end{pmatrix} \qquad (8)$$

where the Nibbles S_0 and S_1 constitute one column in the state matrix in Eq. (1), and the coefficients of the matrix are in $\mathbb{F}_{24}[x]/(x^2+1)$. By using the multiplier designed in Section 4.1, the product of multiplying an element $a \in \mathbb{F}_2[x]/(x^4+x+1)$, by the constant 4, can be calculated as follows:

$$\begin{pmatrix}\tilde{a}_0\\ \tilde{a}_1\\ \tilde{a}_2\\ \tilde{a}_3\end{pmatrix} = \begin{pmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 1 & 1\\ 1 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\end{pmatrix}\cdot\begin{pmatrix}a_0\\ a_1\\ a_2\\ a_3\end{pmatrix}. \qquad (9)$$

The next step in designing a quantum circuit of the **MixColumns** is to implement the matrix in Eq. (9) into the matrix in Eq. (8). This implementation is shown in Figure 6.

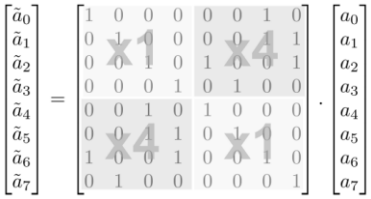

Figure 6. The MixColumns transformation from the ring $\mathbb{F}_2^4[x]/(x^2+1)$ to the ring $\mathbb{F}_2[x]/(x^4+x+1)$ by implementing the matrix in Eq. (9) into the matrix in Eq. (8).

For instance, the **MixColumns** can be implemented as a quantum circuit by using a set of **C-NOT** gates. However, implementing **MixColumns** transformation in the form of how it looks in Figure 6 will require extra auxiliary qubits. Thus, the **C-NOT** synthesis is used again to decompose the transformation in Figure 6. Eventually, the **MixColumns** quantum circuit is shown in Figure 7.

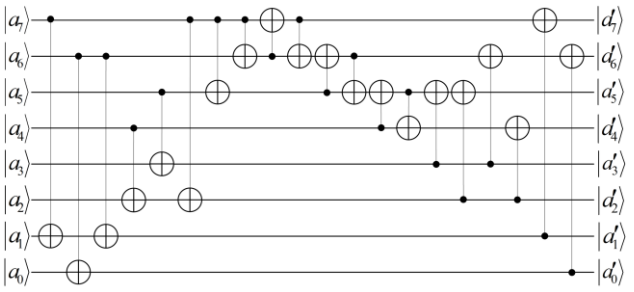

Figure 7. The quantum circuit of the MixColumns after decomposing the matrix in Figure 6 by using the C-NOT synthesis. The three C-NOT gates after the seventh gate from the left, form a swapping between qubit 7 and qubit 6. These three gates can be omitted by interchanging the positions of the two qubits and accordingly the entire design has to be maintained.

4.3 Key Generation

The key generation (expansion) algorithm of S-AES is designed similar to the key generation algorithm of AES. As mentioned in Section 4, the Byte in S-AES corresponds to the Word in AES. Similarly, the Byte is considered along with Nibbles in the key generation algorithm. The main key is 16 bits which are arranged into four Nibbles (N_0, N_1, N_2 and N_3). The concatenation of two Nibbles forms one Byte (B), in which the main key = $\{B_0, B_1\}$. The S-AES makes use of three subkeys expanded from the main key: the whitening key derived directly from the main key (B_0 and B_1). The frst encryption round subkey (B_2 and B_3) and the second round subkey (B_4 and B_5), are generated from the main key as shown in Algorithm 2.

Algorithm 2: Subkeys generation algorithm

Input: A secret key of 16 bits.
Output: A list of Bytes contained the subkeys: $K = \{B_0, B_1, \ldots, B_5\}$.
1 Initialize B_0 and B_1 directly from the secret key.
2 for $2 \le i \le 5$ **do**
3 **if** $i \equiv 0 (\text{mod } 2)$ **then**
4 $K[i] = K[i-2] \oplus \text{RCON}(i/2) \oplus \text{SubNibbles}(\text{RotNibbles}(K[i-1]))$
5 **else**
6 $K[i] = K[i-2] \oplus K[i-1]$

The function of **RotNibbles()** is to rotate the two Nibbles of a Byte and **SubNibbles** is the same function as found in Section 4.1. **RCON()** is a function to calculate the constants used in the key generation algorithm. Those constants are computed in $\mathbb{F}_2[x]/(x^4+x+1)$ such that $\text{RCON}(i)=(x^{i+2} \parallel 0000)$. For instant, RCON[1]=10000000 and RCON[2]=00110000

As shown in the proposed block diagram in Figure 2, the first 16 qubits N_0 - N_3 are dedicated for the main key. Therefore, $B_0 = N_0 \parallel N_1$, and $B_1 = N_2 \parallel N_3$. Since utilizing the lowest number of qubits is the main concern in this work, eight qubits instead of 16 qubits are dedicated for the Bytes (B_2-B_3) of the first round subkey, and another eight qubits for the Bytes (B_4-B_5) of the second round subkey.

The frst round subkey generation starts at $|\psi_3\rangle$ as shown in Figure 2. The **RotNibbles** is implemented for free (without utilizing new qubits) in the same way as in **ShiftRows**. After the

Nibbles N_2 and N_3 are substituted by new Nibbles by using **SubNibbles**, the positions of the output Nibbles are interchanged such that the output of N_2 is placed on Nibble N_5 and N_3 on N_4 as seen at $|\psi_4\rangle$. Then, N_4 and N_3 are XOR-ed with the constant 10000000 by placing a single **Pauli-X** gate on the fourth qubit of the Nibble N_5. To produce the frst Byte B_2 of the subkey, Nibbles N_0 and N_1 of Byte B_0 are XOR-ed with N_4 and N_5. For instance, B_2 is generated and is XOR-ed with the corresponding part of the ciphertext of the first round as shown at $|\psi_5\rangle$. Lastly, the Byte B_1 (N_2 and N_3), is XOR-ed with B_2 to produce the second Byte B_3 which is XOR-ed with the second half of the ciphertext as seen at $|\psi_6\rangle$. The second round subkey is generated similarly as found in the first round. Whereas, two **Pauli-X** gates are implemented on qubits 0 and 1 of the Nibble N_7 which represents XOR-ing with the constant 00110000. Since $B_4 = B_2 \oplus$ RCON[2] $\oplus$ SubNibbles (RotNibbles(B_3)), and B_2 and B_3 are both generated on the same qubits lines (N_4 and N_5), then B_2 is regenerated again by reversing the action of **C-NOT** gates. Subsequently, N_4 and N_5 carry the value of B_2 and this process can be seen at $|\psi_7\rangle$ in Figure 2. Finally, after the subkey is being XOR-ed with the ciphertext from the second round encryption, the output ciphertext of S-AES is obtained on the Nibbles N_{12} - N_{15}.

4.4 Complexity Analysis

The relation between the number of the qubits and the quantum gates is inversely proportional. Although utilizing the lowest possible number of qubits in this work will lead to the increase in the number of gates, but the quantum design still shows a polynomial number of gates. Calculating the quantum cost of implementing S-AES indicates the actual cost of implementing the original AES. Hence, the quantum cost of the proposed quantum circuit of S-AES is calculated and reported in Table 1.

Table 1. Quantum cost of implementing S-AES.

	No. of X-gates	No. of C-NOT	No. of Toffoli	No. of Qubits	No. of Ancilla
Key generation	10	568	192	32	8
Encryption	16	512	384	32	
Total	26	1080	576	64	8

5. QUANTUM S-AES IN A BLACK-BOX

For a quantum adversary to conduct an attack against a symmetric block cipher, having the block cipher implemented as a "reversible" quantum circuit is the first requirement that the quantum adversary needs to fulfill. Note that, the proposed quantum circuit as shown in Figure 2, is until the output ciphertext is produced. Therefore, to have a complete reversible circuit, all the operations implemented should be reversed which means that the total cost of a complete reversible circuit is the product of the cost in Table 1 multiply by 2 and the result is still polynomial. Consequentially, the quantum implementation of a block cipher is not an obstacle that could prevent a quantum adversary from conducting an attack. Note that, this is one of the assumptions of the quantum related-key attack in [9].

Once a quantum adversary implemented a block cipher as a complete reversible quantum circuit, that quantum circuit can be integrated as a function within an Oracle or Black-box. This Black-box is a sort of Boolean function which is a basic building component in other quantum algorithms such as Grover's algorithm [6] or Simon's algorithm [10]. Since a Grover attack against S-AES is being considered in this work, the first step in mounting such attack is to construct the Black-box as shown in Algorithm 1 line 10. The quantum circuit of S-AES is modeled as a Boolean function $f(k)$ to replace the one shown in Eq. (3), as follows:

$$f(k) = \begin{cases} 1, & if\ SAES(k, p_*) = c_*; \\ 0, & otherwise \end{cases} \tag{10}$$

$$|k\rangle \xrightarrow{O} (-1)^{f(k)}|k\rangle \tag{11}$$

where $|k\rangle$ is the state's space of the key, O is the Oracle queried on $|k\rangle$, and in the Boolean function $f(k)$, the pair p_* and c_* is a chosen pair of a Plaintext and its corresponding Ciphertext.

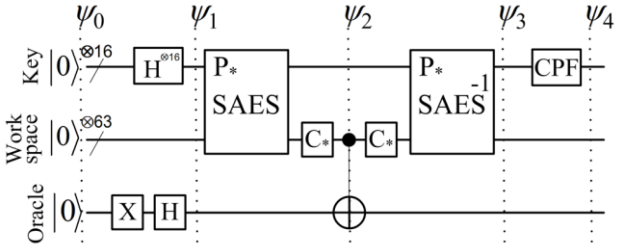

Figure 8. Applying Grover attack on S-AES.

Figure 8 illustrates the complete model of the Grover attack against S-AES. At the initialization phase (at $|\psi_0\rangle$), all the qubits including the Oracle qubit are set to $|0\rangle$ which make the initial state of the system as: $|\psi_0\rangle = |0\rangle^{\otimes 16} \otimes |0\rangle$. Then after setting the Oracle qubit to $|1\rangle$ by using a **Pauli-X** gate, the key qubits and the Oracle qubit are set to a superposition by using a set of Hadamard **H** gates. Thus, the system state at $|\psi_1\rangle$ becomes:

$$|\psi 1\rangle = \mathrm{H}\,|0\rangle^{\otimes 16} \otimes \mathrm{H}\,|1\rangle$$

$$= \frac{1}{256}\sum_{i=0}^{2^{16}-1} \Big|i\Big\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{12}$$

The Oracle or the Black-box shown in Eq. (11), is queried at $|\psi_2\rangle$. Note that, to implement the chosen ciphertext c_*, a set of **Pauli-X** gates is used. After the solution being marked at $|\psi_2\rangle$, the reversing process of S-AES takes place at $|\psi_3\rangle$. Finally, the Grover operator (Conditionally Phase Flip CPF), takes place at $|\psi_4\rangle$. The CPF is implemented by using a set of **Pauli-X** and **H** gates.

6. EXPERIMENTS AND RESULTS

Both of the proposed quantum circuit of S-AES in Section 4, and the Grover attack model in Section 5, are simulated and tested on *libquantum* which is a C library used to simulating quantum mechanics [27]. Three experiments are conducted in this work.

The first experiment is to verify the functionality of the proposed quantum S-AES by comparing the ciphertext with the output of the classical S-AES. The example illustrated by the designers of S-AES [12], is: plaintext p_1= 0110 1111 0110 1011 (which is in ASCII corresponds to "ok"), key k = 1010 0111 0011 1011, and output ciphertext c_1= 0000 0111 0011 1000. The classical S-AES can be expressed as: SAES(k, p_1) = c_1. In the quantum experiment setup, we implemented the same plaintext but rather than implementing the key, a superposition was set for the key qubits to acquire all possibilities of the key. Thus, the quantum S-AES can be expressed as in Eq. (13). The result of this experiment is reported in Table 2.

$$\mathrm{SAES}(\mathrm{H}(\bigotimes_{i=0}^{15} k_i), \bigotimes_{i=0}^{15} p_i) = \sum_{i=0}^{2^{16}-1} (\bigotimes_{i=0}^{15} c_i) \quad (13)$$

Table 2. Results of encrypting p_1 by all possible keys (2^{16} = 65536 keys).

Seq.	Key	Cipherkey	Probability
0	|0000000000000000⟩	|0001101110011000⟩	1.525879×10^{-05}
1	|0000000000000001⟩	|0100011101101010⟩	1.525879×10^{-05}
⋮	⋮	⋮	⋮
42811	|1010011100111011⟩	|0000011100111000⟩	1.525879×10^{-05}
⋮	⋮	⋮	⋮
65534	|1111111111111110⟩	|1100100000011111⟩	1.525879×10^{-05}
65535	|1111111111111111⟩	|1101011111100110⟩	1.525879×10^{-05}

The quantum Grover attack is simulated in the second experiment. At first, the number of the "unrolled" iterations t of Grover's algorithm is calculated by using: $t = \pi/4(\sqrt{\frac{2^k}{s}})$ where t is the total number of iterations, k is the number of the qubits of the key, and s number of the solutions. Surprisingly, more than One key have been detected (precisely two keys), such that they can map the same plaintext p_1 to the similar ciphertext c_1. Hence, the number of the iterations is recalculated as: $t = \pi/4(\sqrt{\frac{2^{16}}{2}}) \Rightarrow t = 142$. The results of this experiment is reported in Table 3.

Table 3. Grover attack using a single pair.

Seq.	Key	Cipherkey	Probability
0	|0000000000000000⟩	|0001101110011000⟩	2.010316×10^{-10}
1	|0000000000000001⟩	|0100011101101010⟩	2.010317×10^{-10}
⋮	⋮	⋮	⋮
42079	|1010010001011111⟩	|0000011100111000⟩	4.999734×10^{-01}
⋮	⋮	⋮	⋮
42811	|1010011100111011⟩	|0000011100111000⟩	4.999734×10^{-01}
⋮	⋮	⋮	⋮
65534	|1111111111111110⟩	|1100100000011111⟩	2.010317×10^{-10}
65535	|1111111111111111⟩	|1101011111100110⟩	2.010323×10^{-10}

To "uniquely" determine the secret key among other undesirable keys, more than one pair of plaintext and ciphertext are needed. The number of the required pairs to conduct an exhaustive key search attack is computed as [28, Chapter 7] and [9]: $r > 2k/n$, where r is the number of pairs, k is the number of the bits of the key, and n is the number of plaintext bits. However, two pairs are adequate to recover the secret key of S-AES. Initially, the Grover attack model shown in Figure 8, is designed for a single pair. Hence, we propose a new Grover attack model to uniquely detect the secret key of S-AES by using just one more qubit as shown in Figure 9.

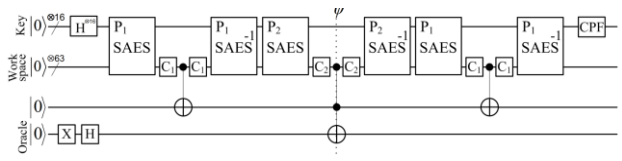

Figure 9. Applying Grover attack on S-AES using two pairs of plaintext and ciphertext.

The Boolean function $f(k)$ shown in Eq. (10), is reconstructed as follows:

$$f(x) = \begin{cases} 1 & if\ SAES(k, p_1) = c_1 \wedge SAES(k, p_2) = c_2; \\ 0 & otherwise \end{cases} \quad (14)$$

The results of this experiment are listed in Table 4. The two pairs of plaintext and ciphertext used in this experiment: p_1 and c_1 are not changed, and p_2=1100111100010011, and c_2=1101000111101101.

Table 4. Grover attack using two pairs.

Seq.	Key	Cipherkey	Probability
0	|0000000000000000⟩	|0001101110011000⟩	1.791471×10^{-10}
1	|0000000000000001⟩	|0100011101101010⟩	1.791471×10^{-10}
⋮	⋮	⋮	⋮
42079	|1010010001011111⟩	|0000011100111000⟩	1.791471×10^{-10}
⋮	⋮	⋮	⋮
42811	|1010011100111011⟩	|0000011100111000⟩	9.999362×10^{-01}
⋮	⋮	⋮	⋮
65534	|1111111111111110⟩	|1100100000011111⟩	1.791471×10^{-10}
65535	|1111111111111111⟩	|1101011111100110⟩	1.791471×10^{-10}

7. CONCLUSION

An explicit detailed quantum design of the Simplified-AES cipher was presented in this work. The design was structured to utilize the lowest possible number of qubits. After constructing the circuits of the basic components of S-AES, those circuits were combined together to form the quantum S-AES. A **C-NOT** synthesis was used for decomposition purposes. The conducted complexity analysis shows a polynomial quantum cost in implementing the block cipher S-AES. Consequently a polynomial cost is also expected in the AES quantum implementation. In addition, a quantum Grover attack was modeled to exhaustively search for the secret key. The proposed quantum S-AES was integrated into a Black-box which is queried in Grover's algorithm. All of the proposals were simulated and tested and the results were compared with the classical S-AES results. The secret key was recovered in $\pi/4(\sqrt{2^k})$ computational steps exactly as expected by Grover's algorithm. Generally, such a quantum design of a classical block cipher is initiated for quantum cryptanalysis. In this work, we show that a quantum circuit of a block cipher can be constructed with a polynomial cost. Therefore, the quantum design is not an obstacle that could prevent a quantum adversary from conducting any possible quantum attack. Hence, it would be interesting to go a step further by investigating those possible quantum threats to the symmetric cryptosystems. For example, a quantum framework of the classical Algebraic attack applied to S-AES in [13], could be beneficial for the security analysis of classical block ciphers against quantum threats.

8. REFERENCES

[1] Rivest R. L., Shamir A., and Adleman L., 1978. A Method for Obtaining Digital Signatures and Public-key Cryptosystems, *Commun. ACM*, 21, 2, 120-126 (February 1978) doi:10.1145/359340.359342

[2] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM J. Comput.* 26, 5, 1484-1509 (October 1997) doi:10.1137/S0097539795293172

[3] Taher ElGamal. 1985. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory*, 31 (4): 469-472. doi:10.1109/TIT.1985.1057074. (conference version appeared in CRYPTO'84, pp.10-18)

[4] Daniel J. Bernstein, Johannes Buchmann, Erik Dahmen. 2008. Post-Quantum Cryptography, Post Quantum

Cryptography (1st ed.). *Springer Publishing Company, Incorporated.*

[5] Yamamura Akihiro, Ishizuka Hirokazu. 2000. Quantum cryptanalysis of block ciphers, *Algebraic Systems, Formal Languages and Computations.* RIMS Kokyuroku, 1166, 235-243

[6] L.K. Grover. 1996. A fast quantum mechanical algorithm for database search, it In Proc. of the 28th Annual ACM Symposium on Theory of Computing (STOC), 212-219

[7] A. Ambainis. 2007. Quantum Walk Algorithm for Element Distinctness, *SIAM Journal on Computing*, 37:210-239,

[8] Marc Kaplan. 2014. Quantum Attacks Against Iterated Block Ciphers ,http://arxiv.org/abc/1410.1434.

[9] Martin Roetteler, Rainer Steinwandt. 2015. A note on quantum related-key attacks, *Information Processing Letters*, V. 115, Issue 1, Pages 40-44, ISSN 0020-0190,

[10] Don Simon. 1994. On the power of quantum computation, *Proceedings of the 35th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 116-123

[11] NIST, 2001. Specification for the ADVANCED ENCRYPTION STANDARD (AES), *Federal Information Processing Standards Publication 197*, (November 2001)

[12] Musa, M.A., Schaefer, E.R., and Wedig, S., 2003. A simplified AES algorithm and its linear and differential cryptanalyses, *Cryptologia* **27**(2) 148-177

[13] Simmons, S., 2009. Algebraic cryptanalysis of simplified AES, *Cryptologia* **33**(4) 305–314

[14] Saeed Rania and Bhery Ashraf. 2015. Cryptanalysis of Simplified-AES Using Intelligent Agent, *Springer International Publishing* Hybrid Artificial Intelligent Systems: 10th International Conference, HAIS 2015, Bilbao, Spain, Proceedings 173–187

[15] Samantha Campbell, Max Grinchenko, and William Smith, 2013. Linear Cryptanalysis of Simplifed AES Under Change of S-Box, *Cryptologia* **37**(2) 120–138

[16] Williams Colin P., 2011. Explorations in Quantum Computing, *Springer London*, Pages 51–122, ISBN: 978-1-84628-887-6, DOI: 10.1007/978-1-84628-887-6_2, http: //dx.doi.org/10.1007/978-1-84628-887-6_2.

[17] N. David Mermin, 2007. Quantum Computer Science: An Introduction, *Cambridge University Press*, New York, NY, USA.

[18] Michael A. Nielsen, and Isaac L. Chuang, 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition (10th ed.),*Cambridge University Press*, New York, NY, USA

[19] Eleanor Rieffel, and Wolfgang Polak, 2011. Quantum Computing: A Gentle Introduction (1st ed.). *The MIT Press.*

[20] Jorge Guajardo and Christof Paar, 2002. Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography and Codes, *Designs, Codes and Cryptography* **25** (2) 207–216

[21] Lee E. J., D. S. Kim, and P. J. Lee, 1998. Speed-up of F*pm* Arithmetic for Eliptic Curve Cryptosystems, *In ICICS 98* Berlin (1998)

[22] Itoh T. and S. Tsujii, 1988. A Fast Algorithm for Computing Multiplicative Inverse in GF(2^m) Using Normal Basis, *Information and Computation* **78** 171–177

[23] Donny Cheung, Dmitri Maslov, Jimson Mathew, and Dhiraj K. Pradhan, 2008. On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography, In Theory of Quantum Computation, Communication, and Cryptography, Yasuhito Kawano and Michele Mosca (Eds.). Lecture Notes In Computer Science, *Springer-Verlag, Berlin, Heidelberg* Vol. 5106. 96-104. DOI=http://dx.doi.org/10.1007/978-3-540-89304-2_9

[24] A. Reyhani-Masoleh and M. A. Hasan, 2004. Low complexity bit parallel architectures for polynomial basis multiplication over GF(2^m), *In IEEE Transactions on Computers*, vol. **53**, no. 8, pp. 945-959, (Aug 2004). doi: 10.1109/TC.2004.47

[25] Fuyou Fan, Guowu Yang, Gang Yang, William N. N. Hung, 2015. A Synthesis Method of Quantum Reversible Logic Circuit Based on Elementary Qutrit Quantum Logic Gates, *Journal of Circuits, Systems, and Computers* **24**, 08, 1550121

[26] Ketan N. Patel, Igor L. Markov, and John P. Hayes, 2008. Optimal synthesis of linear reversible circuits, *Quantum Info. Comput.* **8**, 3, 282-294

[27] Simulation of quantum mechanics, http://www.libquantum.de/. Retrieved 20 Nov 2017.

[28] Alfred J. Menezes, Paul C. Van Ourschot, and Scott A. Vanstone, 2001. Handbook of Applied Cryptography, *CRC Press.*