Grover on Quantum Cryptanalysis

Gopal Ramesh Dahale (11840520)
Supervised by:
Dr. Dhiman Saha

Quantum Symmetric-Key Cryptanalysis (CS614)

Department of EECS
Indian Institute of Technology Bhilai

May 9, 2022

Outline

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Introduction

- Grover's search algorithm recovers key in $O(\sqrt{N})$ calls to quantum oracle where N is the key search space.
- Implementation of quantum circuit of block cipher.
- We study implementation of hardware and software efficient ciphers.
- We study SAES, SIMON 2n/mn, PRESENT, and AES-128.
- Investigation of cost of Grover's Attack with depth constraint as proposed by NIST[1]
- AES-128 under depth constraint.

[1] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Outline

SAES

Nibble oriented with block and key size of 16 bits.

$$\underbrace{b_0\,b_1\,b_2\,b_3}_{S_0}\,\underbrace{b_4\,b_5\,b_6\,b_7}_{S_1}\,\underbrace{b_8\,b_9\,b_{10}\,b_{11}}_{S_2}\,\underbrace{b_{12}\,b_{13}\,b_{14}\,b_{15}}_{S_3} = \begin{bmatrix} S_0 & S_2 \\ S_1 & S_3 \end{bmatrix} = State$$

SAES Encryption

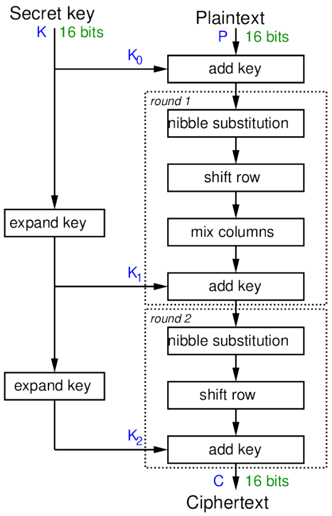


Figure: SAES encryption[2]

[2] Steven Gordon. *Cryptography Study Notes (Chapter 9)*. 2022. URL: https://sandilands.info/crypto/.

Sub Nibbles, Shift Rows and Mix Column

1. Compute the multiplicative inverse x i.e. $y = x^{-1}$ in $GF(2^4)$.
2. The result of the sbox is computed using the follows operation:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

3. The Shift Rows operation is the same as AES.

$$\begin{bmatrix} S_0 & S_2 \\ S_1 & S_3 \end{bmatrix} \longrightarrow \begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$$

4. SAES mix column.

$$\begin{bmatrix} S_0' \\ S_1' \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \end{bmatrix}$$

5. The elements of the matrix are in $\mathbb{F}_{2^4}[x]/(x^2+1)$.

Sub Nibbles, Shift Rows and Mix Column

1. Compute the multiplicative inverse x i.e. $y = x^{-1}$ in $GF(2^4)$.
2. The result of the sbox is computed using the follows operation:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

3. The Shift Rows operation is the same as AES.

$$\begin{bmatrix} S_0 & S_2 \\ S_1 & S_3 \end{bmatrix} \longrightarrow \begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$$

4. SAES mix column.

$$\begin{bmatrix} S_0' \\ S_1' \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \end{bmatrix}$$

5. The elements of the matrix are in $\mathbb{F}_{2^4}[x]/(x^2+1)$.

Sub Nibbles, Shift Rows and Mix Column

1. Compute the multiplicative inverse x i.e. $y = x^{-1}$ in $GF(2^4)$.
2. The result of the sbox is computed using the follows operation:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

3. The Shift Rows operation is the same as AES.

$$\begin{bmatrix} S_0 & S_2 \\ S_1 & S_3 \end{bmatrix} \longrightarrow \begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$$

4. SAES mix column.

$$\begin{bmatrix} S'_0 \\ S'_1 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \end{bmatrix}$$

5. The elements of the matrix are in $\mathbb{F}_{2^4}[x]/(x^2+1)$.

Sub Nibbles, Shift Rows and Mix Column

1. Compute the multiplicative inverse x i.e. $y = x^{-1}$ in $GF(2^4)$.
2. The result of the sbox is computed using the follows operation:

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (1)$$

3. The Shift Rows operation is the same as AES.

$$\begin{bmatrix} S_0 & S_2 \\ S_1 & S_3 \end{bmatrix} \longrightarrow \begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$$

4. SAES mix column.

$$\begin{bmatrix} S_0' \\ S_1' \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \end{bmatrix}$$

5. The elements of the matrix are in $\mathbb{F}_{2^4}[x]/(x^2+1)$.

Key Expansion

- The master key (16 bit) can be thought as 2 bytes B_0B_1.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes B_2B_3.
- Second round key (16 bit) can be thought as 2 bytes B_4B_5.

Key Expansion for SAES(K)

```
1  $keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
2  $keys[0] = K[0 \dots 8]$
3  $keys[1] = K[8 \dots 16]$
4  for $i = 2$ to $5$
5      if $i\%2 == 0$
6          $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
7          $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
8      else
9          $keys[i] = keys[i-2] \oplus keys[i-1]$
10 return $B_0B_1, B_2B_3, B_4B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2}||0000)$$

Key Expansion

- The master key (16 bit) can be thought as 2 bytes $B_0 B_1$.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes $B_2 B_3$.
- Second round key (16 bit) can be thought as 2 bytes $B_4 B_5$.

KEY EXPANSION FOR SAES(K)

```
1  $keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
2  $keys[0] = K[0 \dots 8]$
3  $keys[1] = K[8 \dots 16]$
4  for $i = 2$ to 5
5      if $i \% 2 == 0$
6          $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
7          $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
8      else
9          $keys[i] = keys[i-2] \oplus keys[i-1]$
10 return $B_0 B_1, B_2 B_3, B_4 B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2} || 0000)$$

Key Expansion

- The master key (16 bit) can be thought as 2 bytes $B_0 B_1$.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes $B_2 B_3$.
- Second round key (16 bit) can be thought as 2 bytes $B_4 B_5$.

Key Expansion for SAES(K)

```
1   $keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
2   $keys[0] = K[0 \dots 8]$
3   $keys[1] = K[8 \dots 16]$
4   for $i = 2$ to $5$
5       if $i \% 2 == 0$
6           $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
7           $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
8       else
9           $keys[i] = keys[i-2] \oplus keys[i-1]$
10  return $B_0 B_1, B_2 B_3, B_4 B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2} || 0000)$$

Key Expansion

- The master key (16 bit) can be thought as 2 bytes B_0B_1.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes B_2B_3.
- Second round key (16 bit) can be thought as 2 bytes B_4B_5.

KEY EXPANSION FOR SAES(K)

```
$keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
$keys[0] = K[0 \dots 8]$
$keys[1] = K[8 \dots 16]$
for $i = 2$ to $5$
    if $i\%2 == 0$
        $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
        $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
    else
        $keys[i] = keys[i-2] \oplus keys[i-1]$
return $B_0B_1, B_2B_3, B_4B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2}||0000)$$

Key Expansion

- The master key (16 bit) can be thought as 2 bytes $B_0 B_1$.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes $B_2 B_3$.
- Second round key (16 bit) can be thought as 2 bytes $B_4 B_5$.

Key Expansion for SAES(K)

```
1   $keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
2   $keys[0] = K[0 \dots 8]$
3   $keys[1] = K[8 \dots 16]$
4   for $i = 2$ to 5
5       if $i\%2 == 0$
6           $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
7           $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
8       else
9           $keys[i] = keys[i-2] \oplus keys[i-1]$
10  return $B_0 B_1, B_2 B_3, B_4 B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2} || 0000)$$

Key Expansion

- The master key (16 bit) can be thought as 2 bytes $B_0 B_1$.
- 3 Round keys.
- First round key (16 bit) can be thought as 2 bytes $B_2 B_3$.
- Second round key (16 bit) can be thought as 2 bytes $B_4 B_5$.

KEY EXPANSION FOR SAES(K)

```
1   $keys = [B_0, B_1, B_2, B_3, B_4, B_5]$
2   $keys[0] = K[0 \dots 8]$
3   $keys[1] = K[8 \dots 16]$
4   for $i = 2$ to $5$
5       if $i\%2 == 0$
6           $keys[i] = keys[i-2] \oplus RCON(i/2) \oplus$
7           $keys[i] = keys[i] \oplus Sbox(RotNib(keys[i-1]))$
8       else
9           $keys[i] = keys[i-2] \oplus keys[i-1]$
10  return $B_0 B_1, B_2 B_3, B_4 B_5$
```

- Round Constant is defined as

$$RCON(i) = (x^{i+2} || 0000)$$

Outline

QSAES18

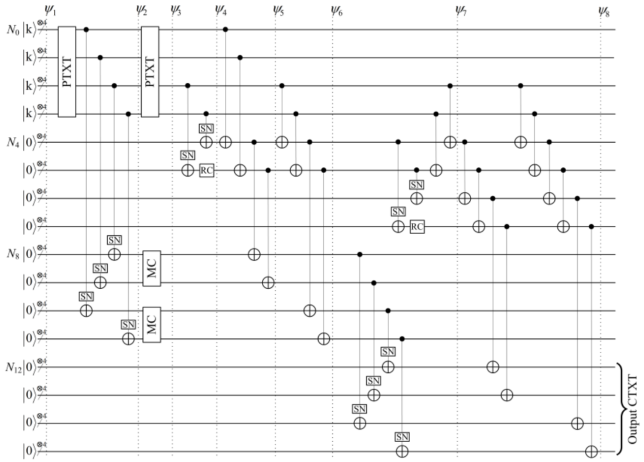


Figure: QSAES18[3]

[3] Mishal Almazrooie et al. "Quantum Grover Attack on the Simplified-AES". In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

Sub Nibbles

Fermat inversion algorithm (square and multiply method) to find multiplicative inverse in $GF(2^4)$

$$x^{-1} = x^{2^4-2} = x^{16-2} = x^{14} = x^2 \times (x^2)^2 \times ((x^2)^2)^2$$

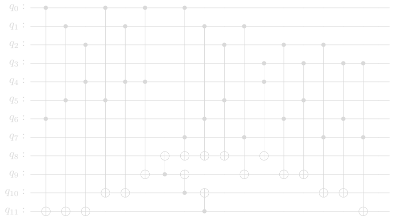


Figure: Multiplier[4]

Input: $q_0 - q_3$ and $q_4 - q_7$. Output: $q_8 - q_{11}$.

[4] Donny Cheung et al. "On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography". In: *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by Yasuhito Kawano and Michele Mosca. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 96–104. ISBN: 978-3-540-89304-2.

Sub Nibbles

Fermat inversion algorithm (square and multiply method) to find multiplicative inverse in $GF(2^4)$

$$x^{-1} = x^{2^4-2} = x^{16-2} = x^{14} = x^2 \times (x^2)^2 \times ((x^2)^2)^2$$

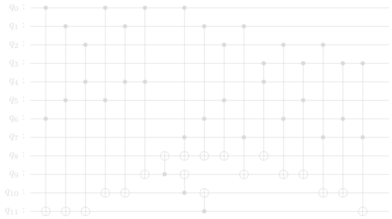


Figure: Multiplier[4]

Input: $q_0 - q_3$ and $q_4 - q_7$. Output: $q_8 - q_{11}$.

[4] Donny Cheung et al. "On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography". In: *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by Yasuhito Kawano and Michele Mosca. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 96–104. ISBN: 978-3-540-89304-2.

Sub Nibbles

Fermat inversion algorithm (square and multiply method) to find multiplicative inverse in $GF(2^4)$

$$x^{-1} = x^{2^4-2} = x^{16-2} = x^{14} = x^2 \times (x^2)^2 \times ((x^2)^2)^2$$

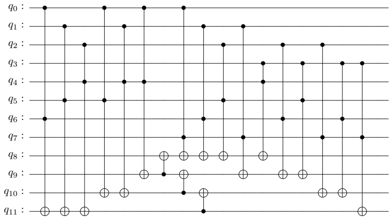


Figure: Multiplier[4]

Input: $q_0 - q_3$ and $q_4 - q_7$. Output: $q_8 - q_{11}$.

[4] Donny Cheung et al. "On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography". In: *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by Yasuhito Kawano and Michele Mosca. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 96–104. ISBN: 978-3-540-89304-2.

Sub Nibbles Contd.

Squarer circuit obtained using CNOT synthesis algorithm.

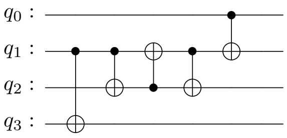


Figure: Squarer

Affine transformation circuit.

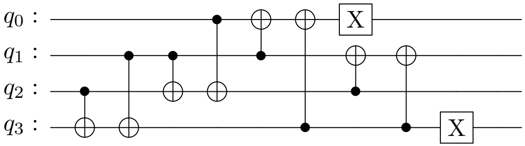


Figure: Affine transformation

Sub Nibbles Complete Circuit

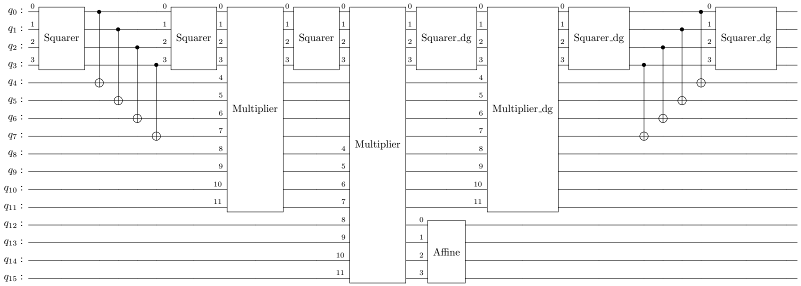


Figure: Sbox

Mix Column

Obtained using CNOT synthesis algorithm.

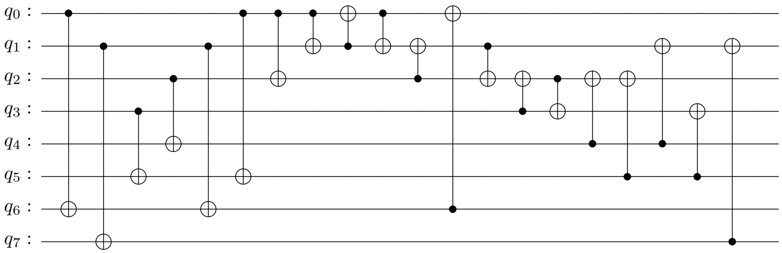


Figure: Mix column

The circuit takes 1 byte (1 column of the state matrix) and outputs the corresponding matrix multiplication in $GF(2^4)$.

Grover's Attack

Boolean function for Grover's oracle:

$$f(k) = \begin{cases} 1 & SAES(k,p) = c \\ 0 & else \end{cases}$$

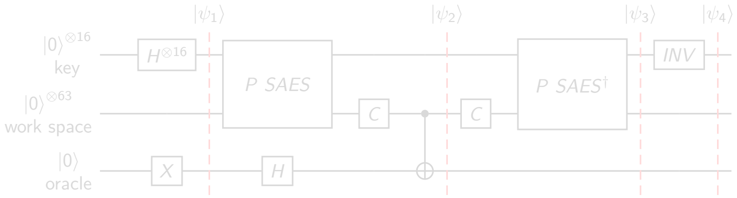


Figure: Grover's Attack on SAES

Number of iterations:

$$t = \frac{\pi}{4}\sqrt{\frac{2^k}{s}} \qquad (2)$$

$s = 2, k = 16$

$$t = \frac{\pi}{4}\sqrt{\frac{2^{16}}{2}} = 142$$

Grover's Attack

Boolean function for Grover's oracle:

$$f(k) = \begin{cases} 1 & SAES(k,p) = c \\ 0 & else \end{cases}$$

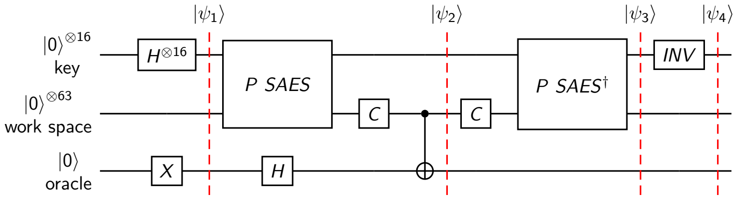


Figure: Grover's Attack on SAES

Number of iterations:

$$t = \frac{\pi}{4}\sqrt{\frac{2^k}{s}} \tag{2}$$

$s = 2, k = 16$

$$t = \frac{\pi}{4}\sqrt{\frac{2^{16}}{2}} = 142$$

Grover's Attack

Boolean function for Grover's oracle:

$$f(k) = \begin{cases} 1 & SAES(k,p) = c \\ 0 & else \end{cases}$$

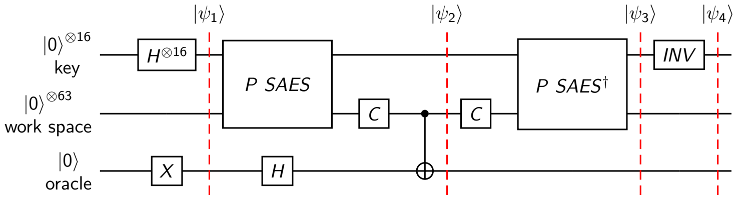


Figure: Grover's Attack on SAES

Number of iterations:

$$t = \frac{\pi}{4}\sqrt{\frac{2^k}{s}} \tag{2}$$

$s = 2, k = 16$

$$t = \frac{\pi}{4}\sqrt{\frac{2^{16}}{2}} = 142$$

Grover's Attack (r = 2)

They propose a modified version of Grover's Attack to find the unique key which is shown below.

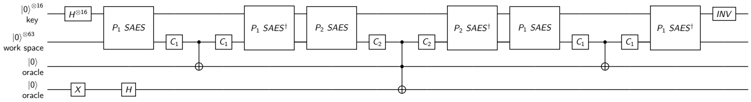


Figure: Grover's Attack to find unique key with r = 2

The corresponding boolean function is described as follows:

$$f(k) = \begin{cases} 1 & (SAES(k, p_1) = c_1) \wedge (SAES(k, p_2) = c_2) \\ 0 & else \end{cases}$$

Grover's Attack (r = 2)

They propose a modified version of Grover's Attack to find the unique key which is shown below.

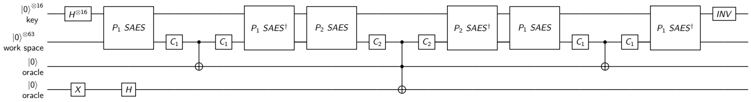


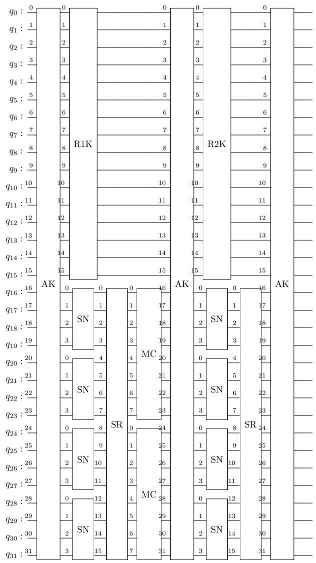
Figure: Grover's Attack to find unique key with r = 2

The corresponding boolean function is described as follows:

$$f(k) = \begin{cases} 1 & (SAES(k, p_1) = c_1) \wedge (SAES(k, p_2) = c_2) \\ 0 & else \end{cases}$$

Outline

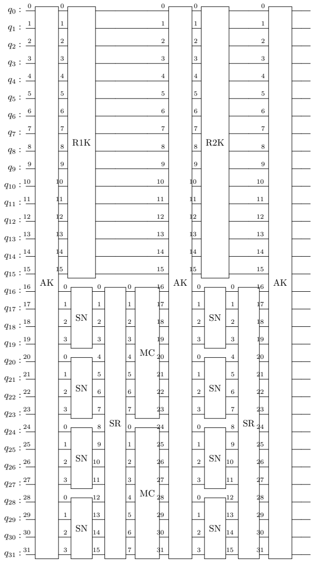
QSAES21[5]



- Only uses 32 qubits and no ancilla qubits.
- The top 16 qubits are used for storing the master key and for the process of key expansion.
- The bottom 16 qubits are used for storing plaintext, round operations, and outputting ciphertext.

[5] Kyung-Bae Jang et al. "Grover on Simplified AES". In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

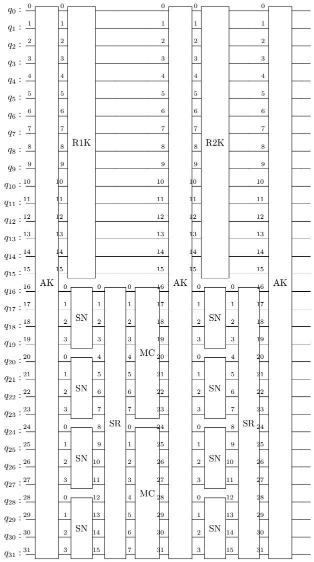
QSAES21[5]



- Only uses 32 qubits and no ancilla qubits.
- The top 16 qubits are used for storing the master key and for the process of key expansion.
- The bottom 16 qubits are used for storing plaintext, round operations, and outputting ciphertext.

[5] Kyung-Bae Jang et al. "Grover on Simplified AES". In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

QSAES21[5]



- Only uses 32 qubits and no ancilla qubits.
- The top 16 qubits are used for storing the master key and for the process of key expansion.
- The bottom 16 qubits are used for storing plaintext, round operations, and outputting ciphertext.

[5] Kyung-Bae Jang et al. "Grover on Simplified AES". In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

Sub Nibbles

- Unlike [Alm+18][6] which uses 16 qubits (4 input, 4 output, and 8 ancillae) for Sbox computation, [Jan+21a][7] uses only 4 qubits using LIGHTER-R tool [Das+19][8].
- Output is permuted so we need SWAP gates (Not measured in quantum resources).

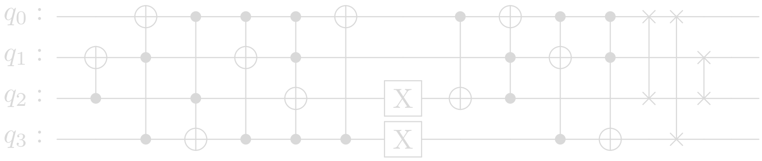


Figure: Sbox

[6] Mishal Almazrooie et al. “Quantum Grover Attack on the Simplified-AES”. In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

[7] Kyung-Bae Jang et al. “Grover on Simplified AES”. In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

[8] Vishnu Asutosh Dasu et al. “LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes”. In: *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 2019, pp. 260–265. DOI: 10.1109/SOCC46988.2019.1570548320.

Sub Nibbles

- Unlike [Alm+18][6] which uses 16 qubits (4 input, 4 output, and 8 ancillae) for Sbox computation, [Jan+21a][7] uses only 4 qubits using LIGHTER-R tool [Das+19][8].
- Output is permuted so we need SWAP gates (Not measured in quantum resources).

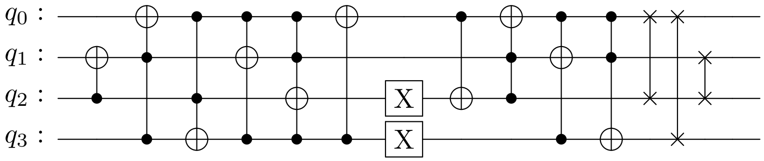


Figure: Sbox

[6] Mishal Almazrooie et al. "Quantum Grover Attack on the Simplified-AES". In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

[7] Kyung-Bae Jang et al. "Grover on Simplified AES". In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

[8] Vishnu Asutosh Dasu et al. "LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes". In: *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 2019, pp. 260–265. DOI: 10.1109/SOCC46988.2019.1570548320.

Shift Rows and Mix Column

- Shift rows operation using swap gates only whereas [Alm+18][9] used extra qubits for that with additional CNOT gates.
- Compared to [Alm+18], the authors use less number of qubits but require SWAP gates to get the correct result.

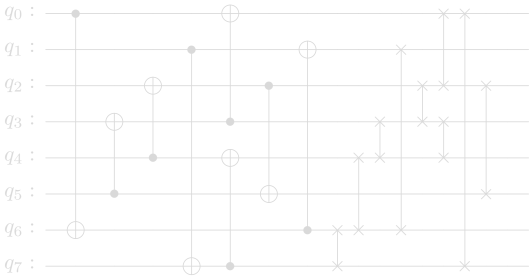


Figure: Mix column

[9] Mishal Almazrooie et al. "Quantum Grover Attack on the Simplified-AES". In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

Shift Rows and Mix Column

- Shift rows operation using swap gates only whereas [Alm+18][9] used extra qubits for that with additional CNOT gates.
- Compared to [Alm+18], the authors use less number of qubits but require SWAP gates to get the correct result.

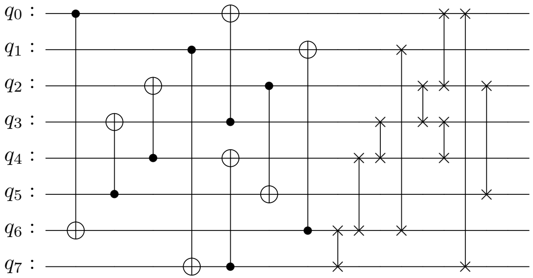


Figure: Mix column

[9] Mishal Almazrooie et al. “Quantum Grover Attack on the Simplified-AES”. In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

Key Exapansion

Round keys are generated on the fly.

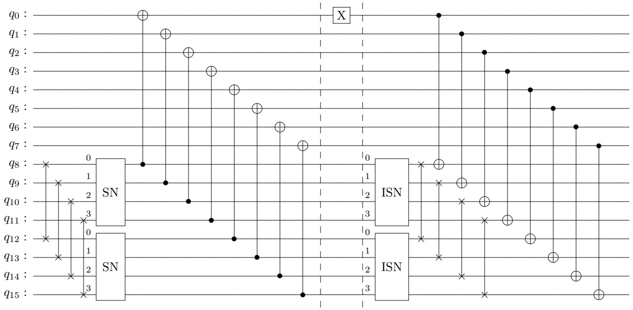


Figure: Round key 1

- Swap and substitution of B_1 then xor it with B_0.
- Add Round constant (10000000).
- Now the first 8 qubits hold B_2. To get B_3, we need to xor B_1 and B_2 but B_1 is lost.
- Inverse swap and substitution on the lower 8 qubits to get back B_1. Xor B_2 with B_1 to get B_3.

Key Exapansion

Round keys are generated on the fly.

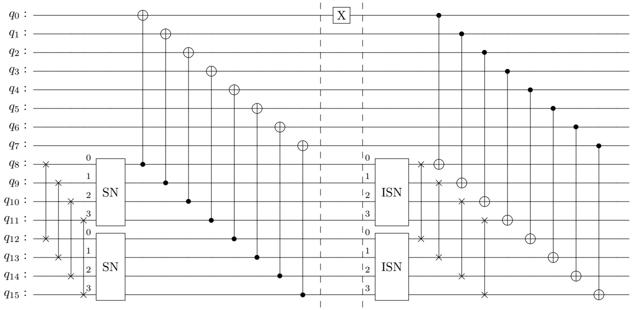


Figure: Round key 1

- Swap and substitution of B_1 then xor it with B_0.
- Add Round constant (10000000).
- Now the first 8 qubits hold B_2. To get B_3, we need to xor B_1 and B_2 but B_1 is lost.
- Inverse swap and substitution on the lower 8 qubits to get back B_1. Xor B_2 with B_1 to get B_3.

Key Exapansion

Round keys are generated on the fly.

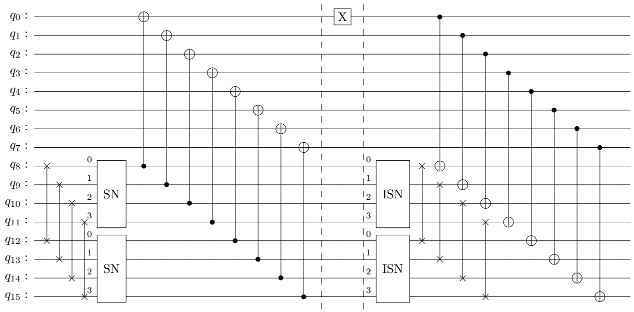


Figure: Round key 1

- Swap and substitution of B_1 then xor it with B_0.
- Add Round constant (10000000).
- Now the first 8 qubits hold B_2. To get B_3, we need to xor B_1 and B_2 but B_1 is lost.
- Inverse swap and substitution on the lower 8 qubits to get back B_1. Xor B_2 with B_1 to get B_3.

Key Exapansion

Round keys are generated on the fly.

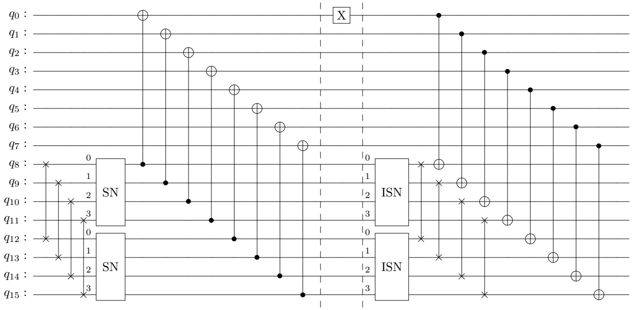


Figure: Round key 1

- Swap and substitution of B_1 then xor it with B_0.
- Add Round constant (10000000).
- Now the first 8 qubits hold B_2. To get B_3, we need to xor B_1 and B_2 but B_1 is lost.
- Inverse swap and substitution on the lower 8 qubits to get back B_1. Xor B_2 with B_1 to get B_3.

Key Exapansion

Round keys are generated on the fly.

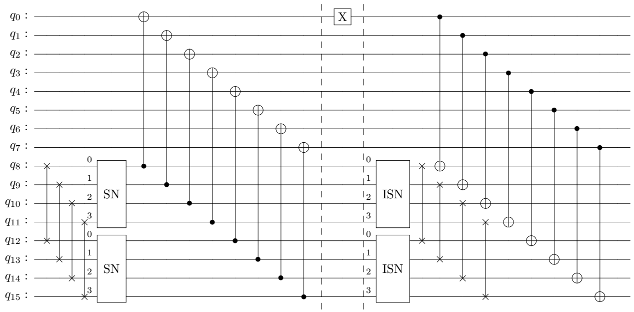
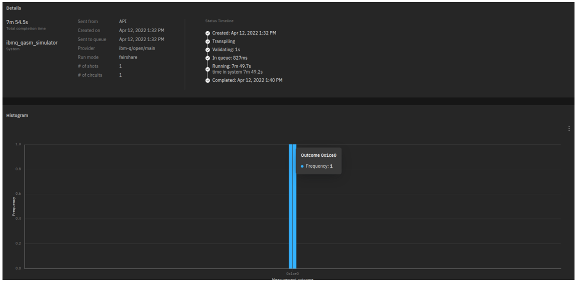


Figure: Round key 1

- Swap and substitution of B_1 then xor it with B_0.
- Add Round constant (10000000).
- Now the first 8 qubits hold B_2. To get B_3, we need to xor B_1 and B_2 but B_1 is lost.
- Inverse swap and substitution on the lower 8 qubits to get back B_1. Xor B_2 with B_1 to get B_3.

Grover's Attack

- Expt1: verified the encryption process on IBMQ QASM Simulator [Qua21][10].
- Expt2 : Superpostion of all keys on IBM's Statevector simulator for 4000 shots.


Details

7m 54.5s
Total completion time

ibmq_qasm_simulator
System

Sent from	API
Created on	Apr 12, 2022 1:32 PM
Sent to queue	Apr 12, 2022 1:32 PM
Provider	ibm-q/open/main
Run mode	fairshare
# of shots	1
# of circuits	1

Status Timeline

- Created: Apr 12, 2022 1:32 PM
- Transpiling
- Validating: 1s
- In queue: 827ms
- Running: 7m 49.7s
  time in system 7m 49.2s
- Completed: Apr 12, 2022 1:40 PM

Histogram

Outcome 0x1ce0
● Frequency: 1

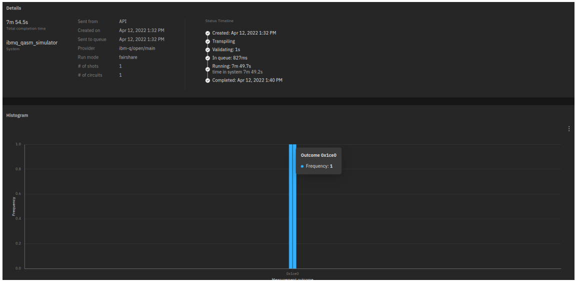
Figure: Output of encryption of plaintext 0110 1111 0110 1011 with key 1010 0111 0011 1011

Output is $0x1ce0$ in hexadecimal, which in binary is 0001 1100 1110 0000 which is the reverse of the ciphertext.

[10] IBM Quantum. *IBMQ Simulators*. 2021. URL: https://quantum-computing.ibm.com/.

Grover's Attack

- Expt1: verified the encryption process on IBMQ QASM Simulator [Qua21][10].
- Expt2 : Superpostion of all keys on IBM's Statevector simulator for 4000 shots.



Details

7m 54.5s
Total completion time

ibmq_qasm_simulator
System

Sent from	API
Created on	Apr 12, 2022 1:32 PM
Sent to queue	Apr 12, 2022 1:32 PM
Provider	ibm-q/open/main
Run mode	fairshare
# of shots	1
# of circuits	1

Status Timeline

- Created: Apr 12, 2022 1:32 PM
- Transpiling
- Validating: 1s
- In queue: 827ms
- Running: 7m 49.7s
  time in system 7m 49.2s
- Completed: Apr 12, 2022 1:40 PM

Histogram

Figure: Output of encryption of plaintext 0110 1111 0110 1011 with key 1010 0111 0011 1011

Output is 0x1ce0 in hexadecimal, which in binary is 0001 1100 1110 0000 which is the reverse of the ciphertext.

[10]IBM Quantum. *IBMQ Simulators*. 2021. URL: https://quantum-computing.ibm.com/.

Grover's Attack Contd.

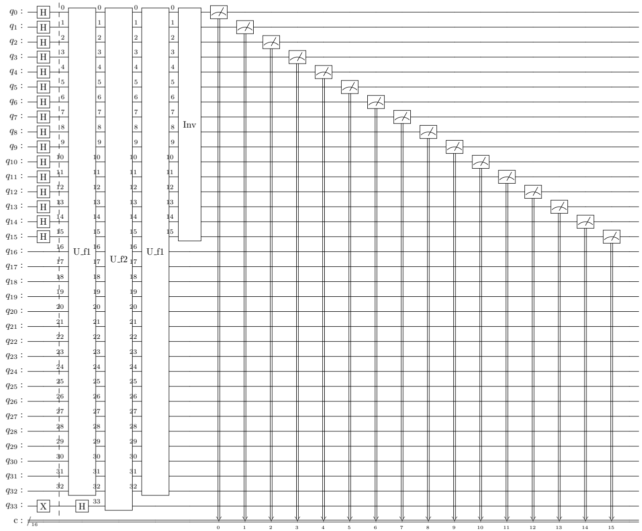


Figure: Grover's Attack for unique key with r = 2 (1 iteration)

Cost Estimates

	Qubits	X	CX	CCX	Ancilla
Key expansion	32	10	568	192	8
Encryption	32	16	512	384	-
Total	64	26	1080	576	8
Key expansion	16	19	56	48	-
Encryption	16	16	88	48	-
Total	32	35	144	96	-
My Code	32	35	144	96	-

Table: Comparison of cost for QSAES18[Alm+18] and QSAES21[Jan+21a]

Cost was heavily reduced due to optimizations in Sbox, key expansion, and mix columns circuit. Cost of Grover's Attack is :

$$2 \times 2 \times \frac{\pi}{4}\sqrt{2^{16}}$$

Cost Estimates

	Qubits	X	CX	CCX	Ancilla
Key expansion	32	10	568	192	8
Encryption	32	16	512	384	-
Total	64	26	1080	576	8
Key expansion	16	19	56	48	-
Encryption	16	16	88	48	-
Total	32	35	144	96	-
My Code	32	35	144	96	-

Table: Comparison of cost for QSAES18[Alm+18] and QSAES21[Jan+21a]

Cost was heavily reduced due to optimizations in Sbox, key expansion, and mix columns circuit. Cost of Grover's Attack is :

$$2 \times 2 \times \frac{\pi}{4}\sqrt{2^{16}}$$

Outline

SIMON 2n/mn[11]

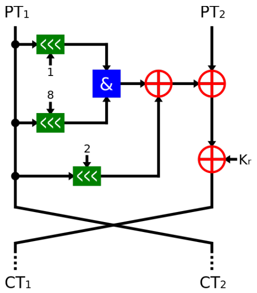


Figure: One round of SIMON [con22a]

Round Function

$$F(x,y) = (y \oplus (S^1(x) \wedge S^8(x)) \oplus S^2(x) \oplus k, x) \quad (3)$$

- $S^j(x)$ denotes left circular shift by j bits.
- PT_1, PT_2 are also referred as L_i, R_i.
- CT_1, CT_2 are also referred as L_{i+1}, R_{i+1}. L_i, R_i are n bit strings as input to the i^{th} round and k is the round key.

[11] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404.

SIMON 2n/mn[11]

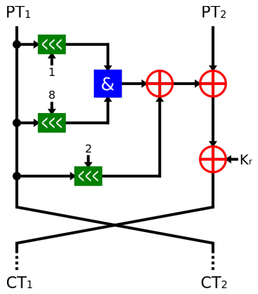


Figure: One round of SIMON [con22a]

Round Function

$$F(x, y) = (y \oplus (S^1(x) \wedge S^8(x)) \oplus S^2(x) \oplus k, x) \quad (3)$$

- $S^j(x)$ denotes left circular shift by j bits.
- PT_1, PT_2 are also referred as L_i, R_i.
- CT_1, CT_2 are also referred as L_{i+1}, R_{i+1}. L_i, R_i are n bit strings as input to the i^{th} round and k is the round key.

[11] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404.

SIMON 2n/mn[11]

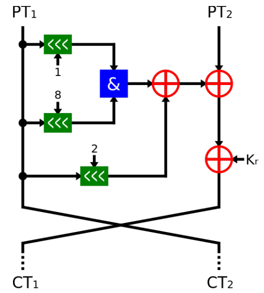


Figure: One round of SIMON [con22a]

Round Function

$$F(x,y) = (y \oplus (S^1(x) \wedge S^8(x)) \oplus S^2(x) \oplus k, x) \quad (3)$$

- $S^j(x)$ denotes left circular shift by j bits.
- PT_1, PT_2 are also referred as L_i, R_i.
- CT_1, CT_2 are also referred as L_{i+1}, R_{i+1}. L_i, R_i are n bit strings as input to the i^{th} round and k is the round key.

[11] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404.

SIMON 2n/mn[11]

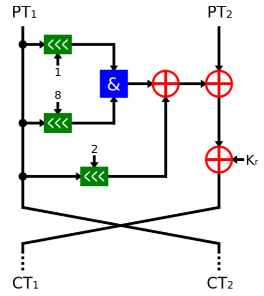


Figure: One round of SIMON [con22a]

Round Function

$$F(x, y) = (y \oplus (S^1(x) \wedge S^8(x)) \oplus S^2(x) \oplus k, x) \quad (3)$$

- $S^j(x)$ denotes left circular shift by j bits.
- PT_1, PT_2 are also referred as L_i, R_i.
- CT_1, CT_2 are also referred as L_{i+1}, R_{i+1}. L_i, R_i are n bit strings as input to the i^{th} round and k is the round key.

[11] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404.

SIMON 2n/mn[11]

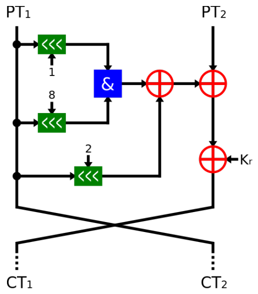


Figure: One round of SIMON [con22a]

Round Function

$$F(x,y) = (y \oplus (S^1(x) \wedge S^8(x)) \oplus S^2(x) \oplus k, x) \quad (3)$$

- $S^j(x)$ denotes left circular shift by j bits.
- PT_1, PT_2 are also referred as L_i, R_i.
- CT_1, CT_2 are also referred as L_{i+1}, R_{i+1}. L_i, R_i are n bit strings as input to the i^{th} round and k is the round key.

[11] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers.* Cryptology ePrint Archive, Report 2013/404.

Key Expansion

- For the first m rounds, the round keys are initialized from the master key.
- For the remaining T-m rounds, use the below function:

$$k_{m+i} = \begin{cases} c_i \oplus k_i \oplus S^{-3}(k_{i+1}) \oplus S^{-4}(k_{i+1}) & m = 2 \\ c_i \oplus k_i \oplus S^{-3}(k_{i+2}) \oplus S^{-4}(k_{i+2}) & m = 3 \\ c_i \oplus k_i \oplus S^{-1}(k_{i+1}) \oplus S^{-3}(k_{i+3}) \oplus S^{-4}(k_{i+3}) & m = 4 \end{cases} \quad (4)$$

c_i denotes the round constants.

Key Expansion

- For the first m rounds, the round keys are initialized from the master key.
- For the remaining T-m rounds, use the below function:

$$k_{m+i} = \begin{cases} c_i \oplus k_i \oplus S^{-3}(k_{i+1}) \oplus S^{-4}(k_{i+1}) & m = 2 \\ c_i \oplus k_i \oplus S^{-3}(k_{i+2}) \oplus S^{-4}(k_{i+2}) & m = 3 \\ c_i \oplus k_i \oplus S^{-1}(k_{i+1}) \oplus S^{-3}(k_{i+3}) \oplus S^{-4}(k_{i+3}) & m = 4 \end{cases} \tag{4}$$

c_i denotes the round constants.

Outline

QSIMON[12]

- The path for i^{th} bit from R_0 which can be written as

$$R_2(i) = L_1(i) = R_0(i) \oplus (L_0(i+1)mod(n) \wedge L_0(i+8)mod(n)) \oplus$$
$$L_0(i+2)mod(n) \oplus k_0$$

- Use the qubits of R_0 for storing L_1 and L_0 becomes R_1. This eliminates the need for ancilla qubits.
- For computing $R_2(i)$, we can use the following operations:
  1. CCX$(L_0(i+1)mod(n), L_0(i+8)mod(n), R_0(i))$
  2. CX$(L_0(i+2)mod(n), R_0(i))$
  3. CX$(k_0(i), R_0(i))$
- n Toffoli gates and $2n$ CNOT gates for a single round.

[12] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. “Grover on SIMON”. In: Apr. 2020.

QSIMON[12]

- The path for i^{th} bit from R_0 which can be written as

$$R_2(i) = L_1(i) = R_0(i) \oplus (L_0(i+1) mod(n) \wedge L_0(i+8) mod(n)) \oplus$$
$$\mathrm{L}_0(i+2) mod(n) \oplus k_0$$

- Use the qubits of R_0 for storing L_1 and L_0 becomes R_1. This eliminates the need for ancilla qubits.
- For computing $R_2(i)$, we can use the following operations:
  1. CCX($L_0(i+1)mod(n)$, $L_0(i+8)mod(n)$, $R_0(i)$)
  2. CX($L_0(i+2)mod(n)$, $R_0(i)$)
  3. CX($k_0(i)$, $R_0(i)$)
- n Toffoli gates and $2n$ CNOT gates for a single round.

[12] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Grover on SIMON". In: Apr. 2020.

QSIMON[12]

- The path for i^{th} bit from R_0 which can be written as

$$R_2(i) = L_1(i) = R_0(i) \oplus (L_0(i+1)mod(n) \wedge L_0(i+8)mod(n)) \oplus$$
$$\text{L}_0(i+2)mod(n) \oplus k_0$$

- Use the qubits of R_0 for storing L_1 and L_0 becomes R_1. This eliminates the need for ancilla qubits.
- For computing $R_2(i)$, we can use the following operations:
  1. $CCX(L_0(i+1)mod(n), L_0(i+8)mod(n), R_0(i))$
  2. $CX(L_0(i+2)mod(n), R_0(i))$
  3. $CX(k_0(i), R_0(i))$
- n Toffoli gates and $2n$ CNOT gates for a single round.

[12] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Grover on SIMON". In: Apr. 2020.

QSIMON[12]

- The path for i^{th} bit from R_0 which can be written as

$$R_2(i) = L_1(i) = R_0(i) \oplus (L_0(i+1) mod(n) \wedge L_0(i+8) mod(n)) \oplus$$
$$\mathrm{L}_0(i+2) mod(n) \oplus k_0$$

- Use the qubits of R_0 for storing L_1 and L_0 becomes R_1. This eliminates the need for ancilla qubits.
- For computing $R_2(i)$, we can use the following operations:
  1. CCX($L_0(i+1) mod(n), L_0(i+8) mod(n), R_0(i)$)
  2. CX($L_0(i+2) mod(n), R_0(i)$)
  3. CX($k_0(i), R_0(i)$)
- n Toffoli gates and $2n$ CNOT gates for a single round.

[12] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Grover on SIMON". In: Apr. 2020.

QSIMON[12]

- The path for i^{th} bit from R_0 which can be written as

$$R_2(i) = L_1(i) = R_0(i) \oplus (L_0(i+1) mod(n) \wedge L_0(i+8) mod(n)) \oplus$$
$$\mathrm{L}_0(i+2) mod(n) \oplus k_0$$

- Use the qubits of R_0 for storing L_1 and L_0 becomes R_1. This eliminates the need for ancilla qubits.
- For computing $R_2(i)$, we can use the following operations:
  1. CCX$(L_0(i+1) mod(n), L_0(i+8) mod(n), R_0(i))$
  2. CX$(L_0(i+2) mod(n), R_0(i))$
  3. CX$(k_0(i), R_0(i))$
- n Toffoli gates and $2n$ CNOT gates for a single round.

[12] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. "Grover on SIMON". In: Apr. 2020.

QSIMON Contd.

Define three functions that together form one round of the QSIMON.

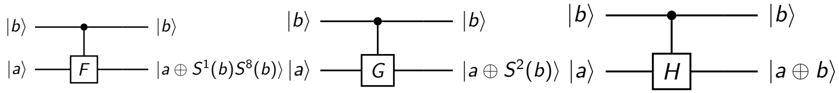


Figure: Subroutines for one round of QSIMON

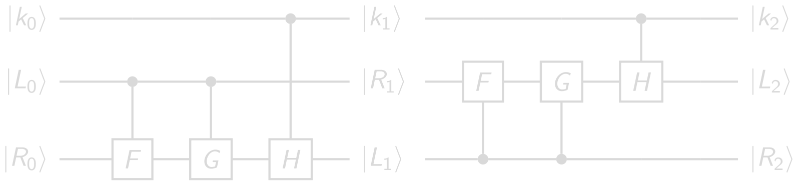


Figure: 2 Round of QSIMON

QSIMON Contd.

Define three functions that together form one round of the QSIMON.

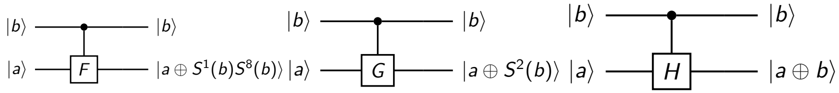


Figure: Subroutines for one round of QSIMON

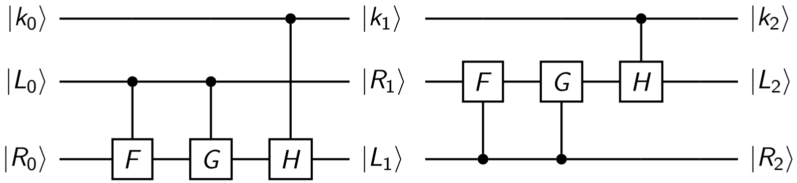


Figure: 2 Round of QSIMON

Key Expansion

Define a function as

$$R_q(a, b) = (S^{-i}(b) \oplus a, b)$$

This function will be used in the circuit of key expansion.

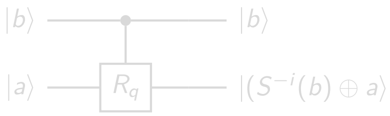


Figure: Quantum circuit for $R_q(a, b)$

Case for $m = 2$.

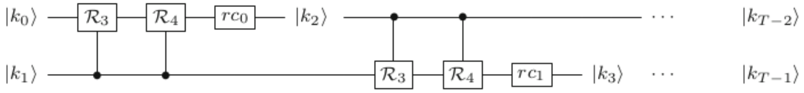


Figure: Key expansion for m = 2 [AMM20]

Key Expansion

Define a function as

$$R_q(a, b) = (S^{-i}(b) \oplus a, b)$$

This function will be used in the circuit of key expansion.

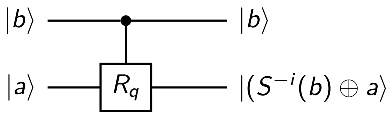


Figure: Quantum circuit for $R_q(a, b)$

Case for $m = 2$.

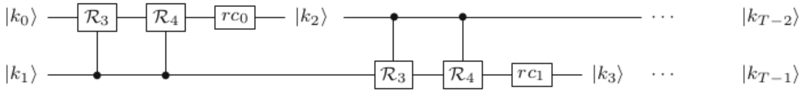


Figure: Key expansion for m = 2 [AMM20]

Key Expansion

Define a function as

$$R_q(a, b) = (S^{-i}(b) \oplus a, b)$$

This function will be used in the circuit of key expansion.

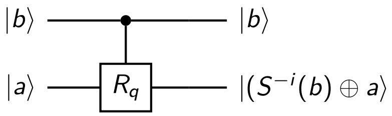


Figure: Quantum circuit for $R_q(a, b)$

Case for $m = 2$.

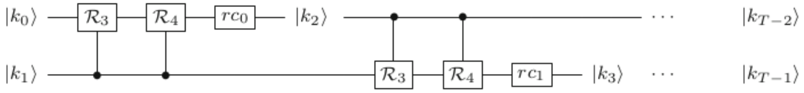


Figure: Key expansion for m = 2 [AMM20]

QSIMON (m = 2)

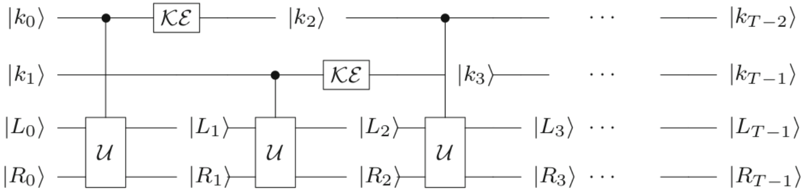


Figure: QSIMON for m = 2 [AMM20]

Grover's Attack

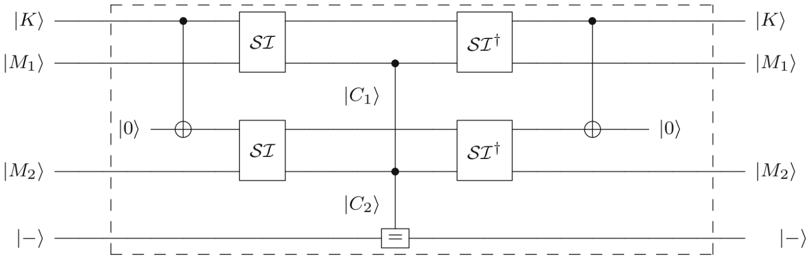


Figure: Grover's Attack on QSIMON [AMM20]

- For finding unique key we need r = 2.
- In general, we require $2nr$ qubits for messasges and mn qubits for master key.
- $O(mn + 2nr)$.

Grover's Attack

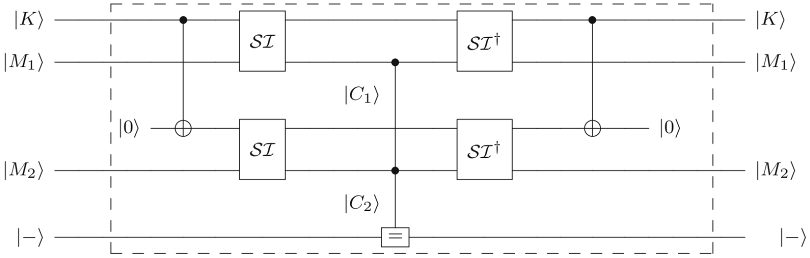


Figure: Grover's Attack on QSIMON [AMM20]

- For finding unique key we need r = 2.
- In general, we require $2nr$ qubits for messasges and mn qubits for master key.
- $O(mn + 2nr)$.

Grover's Attack

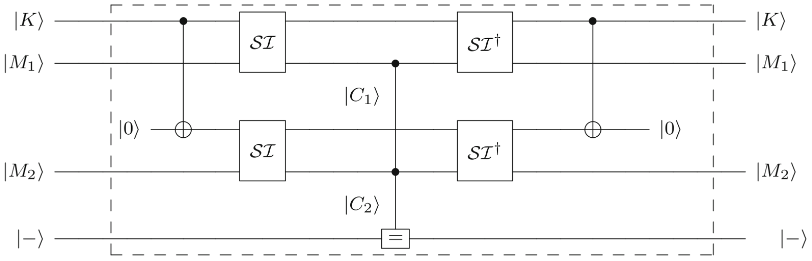


Figure: Grover's Attack on QSIMON [AMM20]

- For finding unique key we need r = 2.
- In general, we require $2nr$ qubits for messasges and mn qubits for master key.
- $O(mn + 2nr)$.

Outline

PRESENT

- Ultra-lightweight block cipher and has a substitution permutation network.
- Block length of 64 bits and 80 and 128-bit key sizes.

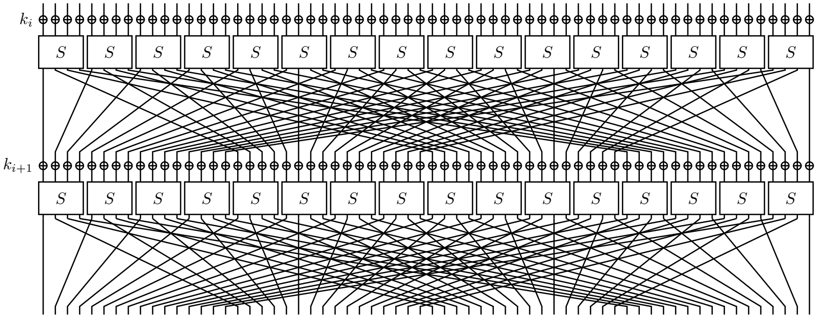


Figure: SP network for PRESENT cipher [Vik07][13]

[13] A. BogdanovL. R. KnudsenG. LeanderC. PaarA. PoschmannM. J. B. RobshawY. SeurinC. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher". In: (2007). URL: https://link.springer.com/chapter/10.1007/978-3-540-74735-2_31.

PRESENT

- Ultra-lightweight block cipher and has a substitution permutation network.
- Block length of 64 bits and 80 and 128-bit key sizes.

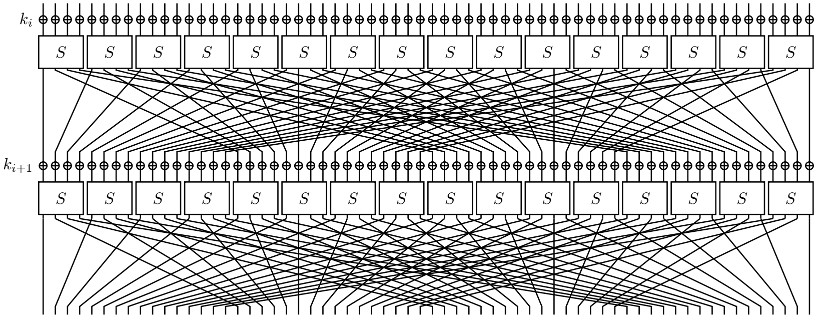


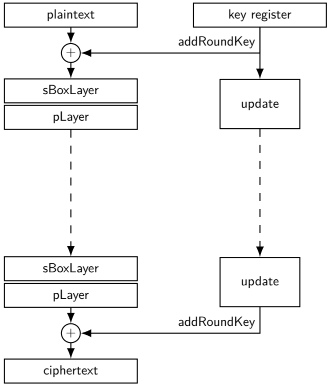
Figure: SP network for PRESENT cipher [Vik07][13]

[13] A. BogdanovL. R. KnudsenG. LeanderC. PaarA. PoschmannM. J. B. RobshawY. SeurinC. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher". In: (2007). URL: https://link.springer.com/chapter/10.1007/978-3-540-74735-2_31.

Cipher Design

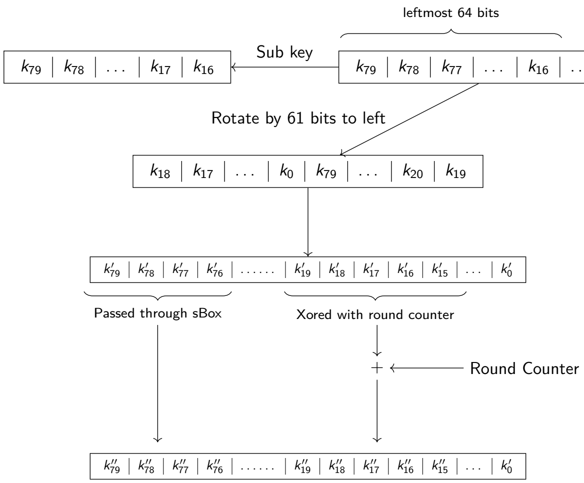
Psuedo-code

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE,$K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
addRoundKey(STATE,$K_{32}$)
```



Key schedule Algorithm

We discuss the 80-bit key schedule algorithm.



leftmost 64 bits
Sub key
k_{79} | k_{78} | ... | k_{17} | k_{16}
k_{79} | k_{78} | k_{77} | ... | k_{16} | ... | k_1 | k_0
Rotate by 61 bits to left
k_{18} | k_{17} | ... | k_0 | k_{79} | ... | k_{20} | k_{19}
k'_{79} | k'_{78} | k'_{77} | k'_{76} | ...... | k'_{19} | k'_{18} | k'_{17} | k'_{16} | k'_{15} | ... | k'_0
Passed through sBox
Xored with round counter
Round Counter
k''_{79} | k''_{78} | k''_{77} | k''_{76} | ...... | k''_{19} | k''_{18} | k''_{17} | k''_{16} | k''_{15} | ... | k'_0

Outline

QPRESENT[15]

- Authors used LIGHTER-R tool [Das+19][14] for optimized implementation of Sbox with no ancilla qubits

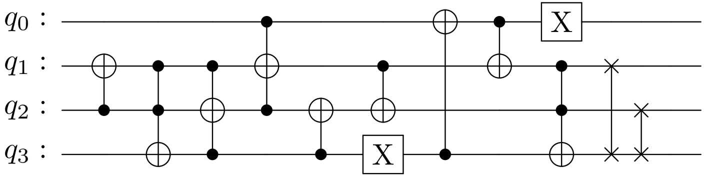


Figure: Sbox for QPRESENT

- Permutation layer can be implemented using only SWAP gates. The quantum cost for the permutation layer of QPRESENT is zero.

[14] Vishnu Asutosh Dasu et al. "LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes". In: *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 2019, pp. 260–265. DOI: 10.1109/SOCC46988.2019.1570548320.

[15] Kyungbae Jang et al. "Efficient Implementation of PRESENT and GIFT on Quantum Computers". In: *Applied Sciences* 11.11 (2021). ISSN: 2076-3417. DOI: 10.3390/app11114776. URL: https://www.mdpi.com/2076-3417/11/11/4776.

QPRESENT[15]

- Authors used LIGHTER-R tool [Das+19][14] for optimized implementation of Sbox with no ancilla qubits

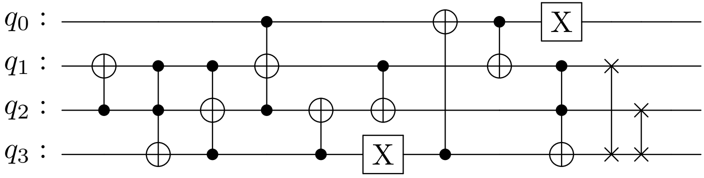


Figure: Sbox for QPRESENT

- Permutation layer can be implemented using only SWAP gates. The quantum cost for the permutation layer of QPRESENT is zero.

[14] Vishnu Asutosh Dasu et al. "LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes". In: *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 2019, pp. 260–265. DOI: 10.1109/SOCC46988.2019.1570548320.

[15] Kyungbae Jang et al. "Efficient Implementation of PRESENT and GIFT on Quantum Computers". In: *Applied Sciences* 11.11 (2021). ISSN: 2076-3417. DOI: 10.3390/app11114776. URL: https://www.mdpi.com/2076-3417/11/11/4776.

Key schedule Algorithm

The input is an 80-bit key and the output is a 64-bit round key.

```
KEY EXPANSION FOR QPRESENT($K = k_{79}k_{78}..k_0$)
1  $k = \mathrm{k}_{63}\mathrm{k}_{62}..\mathrm{k}_{0s} = k_{79}k_{78}..k_{16}$
2  $k = k \gg 19$
3  $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
4  $[k_{19}k_{18}k_{17}k_{16}k_{15}] = X[k_{19}k_{18}k_{17}k_{16}k_{15}]$
5  return $k$
```

Instead of rotating 61 bits to left, rotate 19 bits to right using SWAP gates.

Key schedule Algorithm

The input is an 80-bit key and the output is a 64-bit round key.

```
KEY EXPANSION FOR QPRESENT($K = k_{79}k_{78}..k_0$)
1   $k = \mathrm{k}_{63}\mathrm{k}_{62}..\mathrm{k}_{0\mathrm{s}} = k_{79}k_{78}..k_{16}$
2   $k = k \gg 19$
3   $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
4   $[k_{19}k_{18}k_{17}k_{16}k_{15}] = X[k_{19}k_{18}k_{17}k_{16}k_{15}]$
5   return $k$
```

Instead of rotating 61 bits to left, rotate 19 bits to right using SWAP gates.

Grover's Attack and Cost Estimates

Similar to Grover's Attack on SIMON

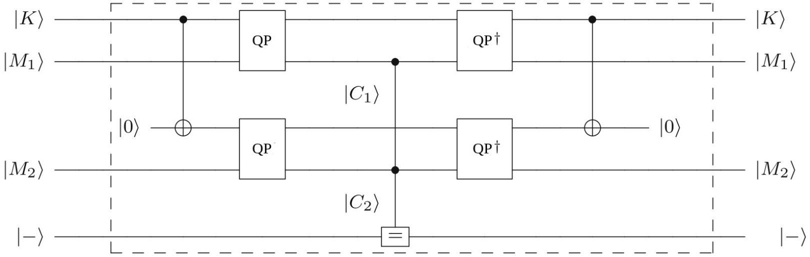


Figure: Grover's Attack on QPRESENT for r = 2

Cipher	Qubits	X	CX	CCX	Ancilla	Depth
QPRESENT 64/80	144	1118	4683	2108	-	311
QSIMON 64/128	192	1216	7396	1408	-	2643

Table: Comparison of cost for QPRESENT 64/80 [Jan+21b] and QSIMON 64/128[AMM20]

Grover's Attack and Cost Estimates

Similar to Grover's Attack on SIMON

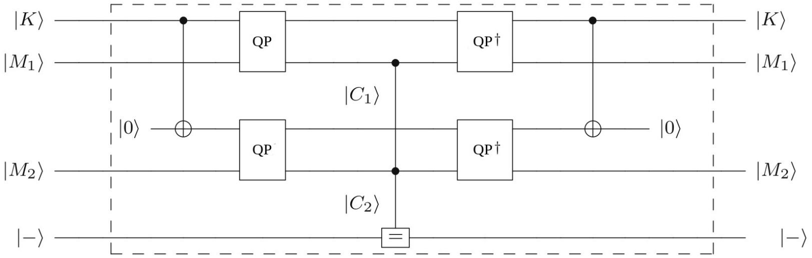


Figure: Grover's Attack on QPRESENT for r = 2

Cipher	Qubits	X	CX	CCX	Ancilla	Depth
QPRESENT 64/80	144	1118	4683	2108	-	311
QSIMON 64/128	192	1216	7396	1408	-	2643

Table: Comparison of cost for QPRESENT 64/80 [Jan+21b] and QSIMON 64/128[AMM20]

Outline

Grover on AES[16]

$N = 2^k$ be the key search space. $M \geq 1$ is the number of solutions. The probability of finding one of the M solutions after t iterations is defined as

$$p(t) = sin^2((2t+1)\theta)$$

which after solving for 1 gives $t \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{\frac{M}{N}}$.

[16] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Grover on AES[16]

$N = 2^k$ be the key search space. $M \geq 1$ is the number of solutions. The probability of finding one of the M solutions after t iterations is defined as

$$p(t) = sin^2((2t+1)\theta)$$

which after solving for 1 gives $t \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{\frac{M}{N}}$.

[16] Samuel Jaques et al. “Implementing Grover Oracles for Quantum Key Search on AES and LowMC”. In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Grover on AES[16]

$N = 2^k$ be the key search space. $M \geq 1$ is the number of solutions. The probability of finding one of the M solutions after t iterations is defined as

$$p(t) = sin^2((2t+1)\theta)$$

which after solving for 1 gives $t \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{\frac{M}{N}}$.

[16] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Grover on AES[16]

$N = 2^k$ be the key search space. $M \geq 1$ is the number of solutions. The probability of finding one of the M solutions after t iterations is defined as

$$p(t) = sin^2((2t+1)\theta)$$

which after solving for 1 gives $t \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{\frac{M}{N}}$.

[16] Samuel Jaques et al. “Implementing Grover Oracles for Quantum Key Search on AES and LowMC”. In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Grover on AES[16]

$N = 2^k$ be the key search space. $M \geq 1$ is the number of solutions. The probability of finding one of the M solutions after t iterations is defined as

$$p(t) = sin^2((2t+1)\theta)$$

which after solving for 1 gives $t \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{\frac{M}{N}}$.

[16] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \dots, E_K(m_r)) = (E_{K'}(m_1), \dots, E_{K'}(m_r))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r))) = 2^{-nr}$$

Key search

Let $E_K(m) = c$ denote the encryption of message $m \in \{0,1\}^n$ by key $K \in \{0,1\}^k$ to ciphertext c. Then the Grover's Oracle is defined as:

$$f(K) = \begin{cases} 1 & E_K(m_i) = c_i \\ 0 & else \end{cases}$$

It is possible that multiple keys other than K lead to the same ciphertext from the given plaintext. Call them spurious keys.

Problem

Find the optimal number r such that the probability of finding a spurious key is minimal.

Let K be the correct key and K' is spurious. Then

$$P_{K \neq K'}(E_K(m) = E_{K'}(m)) = \frac{1}{2^n}$$

for r plaintext-ciphertext pairs, we have:

$$p = P_{K \neq K'}((E_K(m_1), \ldots, E_K(m_r)) = (E_{K'}(m_1), \ldots, E_{K'}(m_r)))) = 2^{-nr}$$

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y = y) = \binom{2^k - 1}{y} p^y (1-p)^{2^k - 1 - y}$$

Approximate this to poission distribution with $\lambda = (2^k - 1)p = (2^k - 1)2^{-rn}$

$$P(Y = y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}} 2^{(k-rn)y}}{y!}$$

$P(Y = 0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \left\lceil \frac{k}{n} \right\rceil$.

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y = y) = \binom{2^k - 1}{y} p^y (1-p)^{2^k - 1 - y}$$

Approximate this to poission distribution with $\lambda = (2^k - 1)p = (2^k - 1)2^{-rn}$

$$P(Y = y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}} 2^{(k-rn)y}}{y!}$$

$P(Y = 0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \left\lceil \frac{k}{n} \right\rceil$.

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y = y) = \binom{2^k - 1}{y} p^y (1 - p)^{2^k - 1 - y}$$

Approximate this to poission distribution with $\lambda = (2^k - 1)p = (2^k - 1)2^{-rn}$

$$P(Y = y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}} 2^{(k-rn)y}}{y!}$$

$P(Y = 0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \left\lceil \frac{k}{n} \right\rceil$.

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y=y) = \binom{2^k-1}{y} p^y (1-p)^{2^k-1-y}$$

Approximate this to poission distribution with $\lambda = (2^k-1)p = (2^k-1)2^{-rn}$

$$P(Y=y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}}2^{(k-rn)y}}{y!}$$

$P(Y=0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \left\lceil \frac{k}{n} \right\rceil$.

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y = y) = \binom{2^k - 1}{y} p^y (1-p)^{2^k - 1 - y}$$

Approximate this to poission distribution with $\lambda = (2^k - 1)p = (2^k - 1)2^{-rn}$

$$P(Y = y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}} 2^{(k-rn)y}}{y!}$$

$P(Y = 0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \left\lceil \frac{k}{n} \right\rceil$.

Key search Contd.

Y be a binomially distributed random variable that describes the count of spurious keys for given key K and r plaintext-ciphertext pairs.

$$P(Y=y) = \binom{2^k - 1}{y} p^y (1-p)^{2^k-1-y}$$

Approximate this to poission distribution with
$\lambda = (2^k - 1)p = (2^k - 1)2^{-rn}$

$$P(Y=y) = \frac{e^{-\lambda}\lambda^k}{y!} \approx \frac{e^{-2^{k-rn}} 2^{(k-rn)y}}{y!}$$

$P(Y=0) \approx e^{-2^{k-rn}}$ i.e. no spurious keys. Therefore $rn > k$ or we can choose $r = \lceil \frac{k}{n} \rceil$.

Parallelization of Grover

- Two ways described by [KHJ18][17]. Inner and outer. Multiple instances of the full Grover's algorithm are run on different machines simultaneously for a reduced number of iterations in outer parallelization.
- The search space is divided into multiple disjoint subsets and each machine is assigned one subset, in case of inner parallelization.
- [Zal99][18] found that there is a gain of $\sqrt{S}$ in the number of iterations for S parallel machines. This is inefficient as we gain only $\sqrt{S}$ factor in the depth of the quantum circuit whereas the width has become S times the original. [Jaq+20][19] uses inner parallelization.

[17] Panjin Kim, Daewan Han, and Kyung Jeong. "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2". In: *Quantum Information Processing* 17 (Oct. 2018). DOI: 10.1007/s11128-018-2107-3.

[18] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Phys. Rev. A* 60 (4 Oct. 1999), pp. 2746–2751. DOI: 10.1103/PhysRevA.60.2746. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746.

[19] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Parallelization of Grover

- Two ways described by [KHJ18][17]. Inner and outer. Multiple instances of the full Grover's algorithm are run on different machines simultaneously for a reduced number of iterations in outer parallelization.
- The search space is divided into multiple disjoint subsets and each machine is assigned one subset, in case of inner parallelization.
- [Zal99][18] found that there is a gain of $\sqrt{S}$ in the number of iterations for S parallel machines. This is inefficient as we gain only $\sqrt{S}$ factor in the depth of the quantum circuit whereas the width has become S times the original. [Jaq+20][19] uses inner parallelization.

[17] Panjin Kim, Daewan Han, and Kyung Jeong. "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2". In: *Quantum Information Processing* 17 (Oct. 2018). DOI: 10.1007/s11128-018-2107-3.

[18] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Phys. Rev. A* 60 (4 Oct. 1999), pp. 2746–2751. DOI: 10.1103/PhysRevA.60.2746. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746.

[19] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Parallelization of Grover

- Two ways described by [KHJ18][17]. Inner and outer. Multiple instances of the full Grover's algorithm are run on different machines simultaneously for a reduced number of iterations in outer parallelization.
- The search space is divided into multiple disjoint subsets and each machine is assigned one subset, in case of inner parallelization.
- [Zal99][18] found that there is a gain of $\sqrt{S}$ in the number of iterations for S parallel machines. This is inefficient as we gain only $\sqrt{S}$ factor in the depth of the quantum circuit whereas the width has become S times the original. [Jaq+20][19] uses inner parallelization.

[17] Panjin Kim, Daewan Han, and Kyung Jeong. "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2". In: *Quantum Information Processing* 17 (Oct. 2018). DOI: 10.1007/s11128-018-2107-3.

[18] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Phys. Rev. A* 60 (4 Oct. 1999), pp. 2746–2751. DOI: 10.1103/PhysRevA.60.2746. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746.

[19] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Why inner parallelization?

- In outer parallelization, the probability that we find the correct key after t iterations is $p_S(t) = 1 - (1 - p(t))^S$.
- In each machine the number of iterations will be $t_S = \frac{\pi}{4\theta\sqrt{S}}$.
- General expression by using the series expansion of $sin(x)$ for larger values of S.

$$p_S(t_S) = 1 - \left(1 - \frac{\pi^2}{4S} + O\left(\frac{1}{S^2}\right)\right)^S, \sum_{y=1}^{\infty} P(Y = y) = 1 - e^{-\frac{2^{k-rn}}{S}}$$

- As S tends to ∞, the above value approaches to $1 - e^{\frac{-\pi^2}{4}} \approx 0.91$.
- This implies by just by increasing the number of parallel machines S, one cannot get a probability near 1 for finding the correct key.
- For inner parallelization, the correct key exists in one of the subsets only, and with t_S iterations, the machine has a near 1 probability of finding it and other machines will not find it.
- If spurious keys are present in a subset with the correct key not in that subset, then the spurious key can be discarded classically after the experiment. Increasing S makes the above probability small.

Quantum Circuit Design

Use AND gate instead of Toffoli gate. The decomposition of the Toffoli gate is to 7 T gates, 8 Clifford gates with a T-depth 4 and total depth 11 whereas AND gate used 4 T gates, 11 Clifford gates with T-depth 1 and total depth 8. AND gate uses one ancilla qubit which is released after the operation.

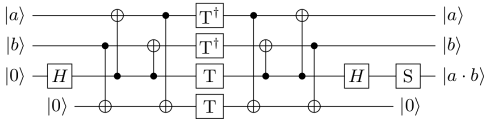


Figure: AND Gate [Jaq+20]

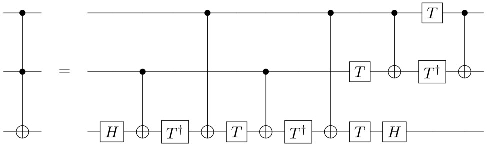


Figure: Toffoli gate decomposition[con22b]

Quantum Circuit Design

Use AND gate instead of Toffoli gate. The decomposition of the Toffoli gate is to 7 T gates, 8 Clifford gates with a T-depth 4 and total depth 11 whereas AND gate used 4 T gates, 11 Clifford gates with T-depth 1 and total depth 8. AND gate uses one ancilla qubit which is released after the operation.

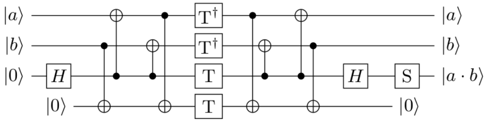


Figure: AND Gate [Jaq+20]

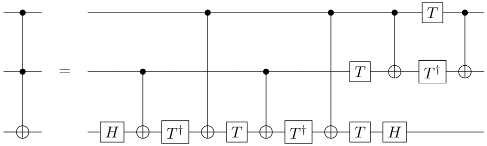


Figure: Toffoli gate decomposition[con22b]

Quantum Circuit Design

Use AND gate instead of Toffoli gate. The decomposition of the Toffoli gate is to 7 T gates, 8 Clifford gates with a T-depth 4 and total depth 11 whereas AND gate used 4 T gates, 11 Clifford gates with T-depth 1 and total depth 8. AND gate uses one ancilla qubit which is released after the operation.

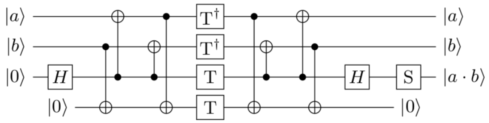


Figure: AND Gate [Jaq+20]

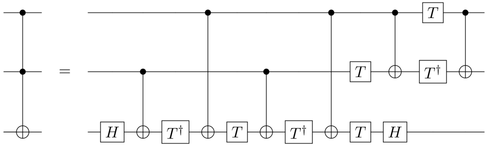


Figure: Toffoli gate decomposition[con22b]

Quantum Circuit Design

Use AND gate instead of Toffoli gate. The decomposition of the Toffoli gate is to 7 T gates, 8 Clifford gates with a T-depth 4 and total depth 11 whereas AND gate used 4 T gates, 11 Clifford gates with T-depth 1 and total depth 8. AND gate uses one ancilla qubit which is released after the operation.

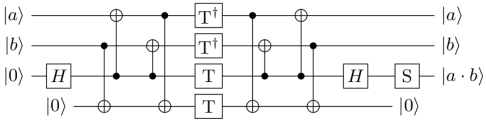


Figure: AND Gate [Jaq+20]

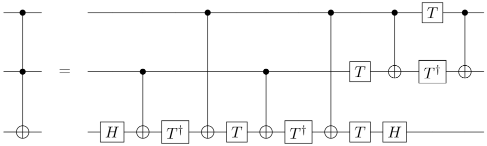


Figure: Toffoli gate decomposition[con22b]

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \quad (5)$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \quad (6)$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \quad (7)$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics

Depth limit D_{max}. Two types of cost: G-cost for the total number of gates and DW-cost which is the product of the depth and width of the circuit. $N = 2^k$ be the key search space. $S = 2^s$ is the count of parallel machines. Assume that G is the oracle for Grover has G_G gates with G_D depth using G_W qubits. Number of iterations for a probability p i.e. $t_p = \frac{sin^{-1}(\sqrt{p})/\theta - 1}{2} \approx \frac{sin^{-1}(\sqrt{p})}{2}\sqrt{\frac{N}{S}}$. Assume $sin^{-1}(\sqrt{p})/2 = c_p$.

$$D = t_p G_D \approx c_p 2^{\frac{k-s}{2}} G_D \tag{5}$$

The total gate cost over all S machines (G-cost) is:

$$G = t_p G_G S \approx c_p 2^{\frac{k+s}{2}} G_G \tag{6}$$

The total width $W = G_W S$ qubits. Therefore the DW-cost is :

$$DW \approx c_p 2^{\frac{s+k}{2}} G_D G_W \tag{7}$$

We can see that reducing S results in a reduction in DW-cost and G-cost. In case of depth constraint, attacker has to parallelize the circuit.

Cost Metrics Contd.

We can run at most $t_{max} = D_{max}/G_D$ iterations of G. For probablitiy p of finding the correct key, we calculate S i.e. $p = sin^2((2t_{max}+1)\sqrt{\frac{S}{N}})$. This gives:

$$S = \frac{(sin^{-1}(\sqrt{p}))^2 N}{(2\frac{D_{max}}{G_D}+1)^2} \approx c_p^2 2^k \frac{G_D^2}{D_{max}^2} \tag{8}$$

$$G = c_p^2 2^k \frac{G_D G_G}{D_{max}} \tag{9}$$

$$DW = c_p^2 2^k \frac{G_D^2 G_W}{D_{max}} \tag{10}$$

Cost Metrics Contd.

We can run at most $t_{max} = D_{max}/G_D$ iterations of G. For probablitiy p of finding the correct key, we calculate S i.e. $p = sin^2((2t_{max}+1)\sqrt{\frac{S}{N}})$. This gives:

$$S = \frac{(sin^{-1}(\sqrt{p}))^2 N}{(2\frac{D_{max}}{G_D}+1)^2} \approx c_p^2 2^k \frac{G_D^2}{D_{max}^2} \tag{8}$$

$$G = c_p^2 2^k \frac{G_D G_G}{D_{max}} \tag{9}$$

$$DW = c_p^2 2^k \frac{G_D^2 G_W}{D_{max}} \tag{10}$$

Cost Metrics Contd.

We can run at most $t_{max} = D_{max}/G_D$ iterations of G. For probablitiy p of finding the correct key, we calculate S i.e. $p = sin^2((2t_{max}+1)\sqrt{\frac{S}{N}})$. This gives:

$$S = \frac{(sin^{-1}(\sqrt{p}))^2 N}{(2\frac{D_{max}}{G_D}+1)^2} \approx c_p^2 2^k \frac{G_D^2}{D_{max}^2} \tag{8}$$

$$G = c_p^2 2^k \frac{G_D G_G}{D_{max}} \tag{9}$$

$$DW = c_p^2 2^k \frac{G_D^2 G_W}{D_{max}} \tag{10}$$

Cost Metrics Contd.

We can run at most $t_{max} = D_{max}/G_D$ iterations of G. For probablitiy p of finding the correct key, we calculate S i.e. $p = sin^2((2t_{max}+1)\sqrt{\frac{S}{N}})$. This gives:

$$S = \frac{(sin^{-1}(\sqrt{p}))^2 N}{(2\frac{D_{max}}{G_D}+1)^2} \approx c_p^2 2^k \frac{G_D^2}{D_{max}^2} \tag{8}$$

$$G = c_p^2 2^k \frac{G_D G_G}{D_{max}} \tag{9}$$

$$DW = c_p^2 2^k \frac{G_D^2 G_W}{D_{max}} \tag{10}$$

Cost Metrics Contd.

We can run at most $t_{max} = D_{max}/G_D$ iterations of G. For probablitiy p of finding the correct key, we calculate S i.e. $p = sin^2((2t_{max} + 1)\sqrt{\frac{S}{N}})$. This gives:

$$S = \frac{(sin^{-1}(\sqrt{p}))^2 N}{(2\frac{D_{max}}{G_D} + 1)^2} \approx c_p^2 2^k \frac{G_D^2}{D_{max}^2} \tag{8}$$

$$G = c_p^2 2^k \frac{G_D G_G}{D_{max}} \tag{9}$$

$$DW = c_p^2 2^k \frac{G_D^2 G_W}{D_{max}} \tag{10}$$

Outline

QAES[23]

- Authors compared various previously proposed Sbox designs on the G-cost and DW-cost metrics by reconstructing them.
- [BP11] Sbox was effective in terms of G-cost and DW-cost hence they chose it.

Sbox	CNOT	Clifford	T	M	T-depth	full depth	width	DW
[Gra+15][20]	8683	1028	3584	0	217	1692	44	74,448
[BP10][21]	818	264	164	41	35	497	41	20,377
[BP11][22]	654	184	136	34	6	101	137	13,837

Table: Comparison of cost of sboxes

- Add round key operation can be implemented simply using 128 CNOT gates from the key to the state.
- Shift rows can be implemented using SWAP gates only and requires zero cost.

[20] Markus Grassl et al. "Applying Grover's algorithm to AES: quantum resource estimates". In: (Dec. 2015).

[21] Joan Boyar and René Peralta. "A New Combinational Logic Minimization Technique with Applications to Cryptology". In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[22] Joan Boyar and René Peralta. "A depth-16 circuit for the AES S-box.". In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[23] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

QAES[23]

- Authors compared various previously proposed Sbox designs on the G-cost and DW-cost metrics by reconstructing them.
- [BP11] Sbox was effective in terms of G-cost and DW-cost hence they chose it.

Sbox	CNOT	Clifford	T	M	T-depth	full depth	width	DW
[Gra+15][20]	8683	1028	3584	0	217	1692	44	74,448
[BP10][21]	818	264	164	41	35	497	41	20,377
[BP11][22]	654	184	136	34	6	101	137	13,837

Table: Comparison of cost of sboxes

- Add round key operation can be implemented simply using 128 CNOT gates from the key to the state.
- Shift rows can be implemented using SWAP gates only and requires zero cost.

[20] Markus Grassl et al. "Applying Grover's algorithm to AES: quantum resource estimates". In: (Dec. 2015).

[21] Joan Boyar and René Peralta. "A New Combinational Logic Minimization Technique with Applications to Cryptology". In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[22] Joan Boyar and René Peralta. "A depth-16 circuit for the AES S-box.". In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[23] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

QAES[23]

- Authors compared various previously proposed Sbox designs on the G-cost and DW-cost metrics by reconstructing them.
- [BP11] Sbox was effective in terms of G-cost and DW-cost hence they chose it.

Sbox	CNOT	Clifford	T	M	T-depth	full depth	width	DW
[Gra+15][20]	8683	1028	3584	0	217	1692	44	74,448
[BP10][21]	818	264	164	41	35	497	41	20,377
[BP11][22]	654	184	136	34	6	101	137	13,837

Table: Comparison of cost of sboxes

- Add round key operation can be implemented simply using 128 CNOT gates from the key to the state.
- Shift rows can be implemented using SWAP gates only and requires zero cost.

[20] Markus Grassl et al. "Applying Grover's algorithm to AES: quantum resource estimates". In: (Dec. 2015).

[21] Joan Boyar and René Peralta. "A New Combinational Logic Minimization Technique with Applications to Cryptology". In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[22] Joan Boyar and René Peralta. "A depth-16 circuit for the AES S-box.". In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[23] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

QAES[23]

- Authors compared various previously proposed Sbox designs on the G-cost and DW-cost metrics by reconstructing them.
- [BP11] Sbox was effective in terms of G-cost and DW-cost hence they chose it.

Sbox	CNOT	Clifford	T	M	T-depth	full depth	width	DW
[Gra+15][20]	8683	1028	3584	0	217	1692	44	74,448
[BP10][21]	818	264	164	41	35	497	41	20,377
[BP11][22]	654	184	136	34	6	101	137	13,837

Table: Comparison of cost of sboxes

- Add round key operation can be implemented simply using 128 CNOT gates from the key to the state.
- Shift rows can be implemented using SWAP gates only and requires zero cost.

[20] Markus Grassl et al. "Applying Grover's algorithm to AES: quantum resource estimates". In: (Dec. 2015).

[21] Joan Boyar and René Peralta. "A New Combinational Logic Minimization Technique with Applications to Cryptology". In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[22] Joan Boyar and René Peralta. "A depth-16 circuit for the AES S-box.". In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[23] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

QAES[23]

- Authors compared various previously proposed Sbox designs on the G-cost and DW-cost metrics by reconstructing them.
- [BP11] Sbox was effective in terms of G-cost and DW-cost hence they chose it.

Sbox	CNOT	Clifford	T	M	T-depth	full depth	width	DW
[Gra+15][20]	8683	1028	3584	0	217	1692	44	74,448
[BP10][21]	818	264	164	41	35	497	41	20,377
[BP11][22]	654	184	136	34	6	101	137	13,837

Table: Comparison of cost of sboxes

- Add round key operation can be implemented simply using 128 CNOT gates from the key to the state.
- Shift rows can be implemented using SWAP gates only and requires zero cost.

[20] Markus Grassl et al. "Applying Grover's algorithm to AES: quantum resource estimates". In: (Dec. 2015).

[21] Joan Boyar and René Peralta. "A New Combinational Logic Minimization Technique with Applications to Cryptology". In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[22] Joan Boyar and René Peralta. "A depth-16 circuit for the AES S-box.". In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[23] Samuel Jaques et al. "Implementing Grover Oracles for Quantum Key Search on AES and LowMC". In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES Contd.

- The authors compared two variants of the mix column.
- In-place which does not require the use of ancilla qubits which saves the width.
- Not in-place and requires ancilla qubits but saves the depth of the circuit.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
In place	1108	0	0	0	0	111	128	14,208
[Max19][24]	1248	0	0	0	0	22	318	6,996

Table: Comparison of cost of mix column variants

The authors chose [Max19] mix column variant due to its low DW cost. The DW cost is mainly affected by the G_D^2 term and therefore it is crucial to minimize the depth of the oracle used, here its mix column.

[24] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

Key Expansion

Generate keys on the fly which do not require ancilla qubits.

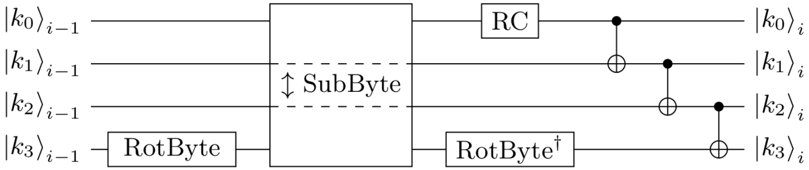


Figure: AES-128 key expansion [Jaq+20]

$|k_j\rangle_i$ represent the j^{th} word (4 bytes) of the i^{th} round key.

QAES-128 circuit

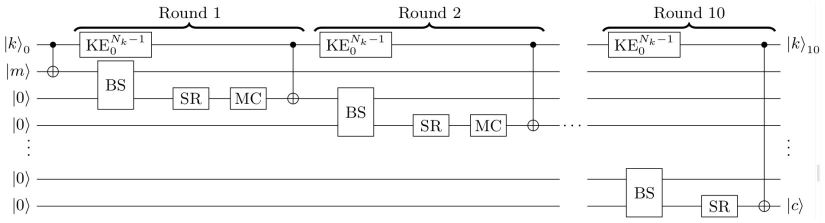


Figure: QAES-128 [Jaq+20]

Each wire represents 4 words (128 qubits). $|k\rangle$ represents the master key and m represents the message.BS is Sbox, SR is shift rows and MC is mix column. Here we have used an in-place version of mix columns.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	2,91,150	83,116	54,400	13,600	120	2,827	1,785	50,46,195
QAES-128 [Max19][25]	2,93,730	83,236	54,400	13,600	120	2,094	2,937	61,50,078

Table: QAES-128 cost of both variants of mix columns

[25] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES-128 circuit

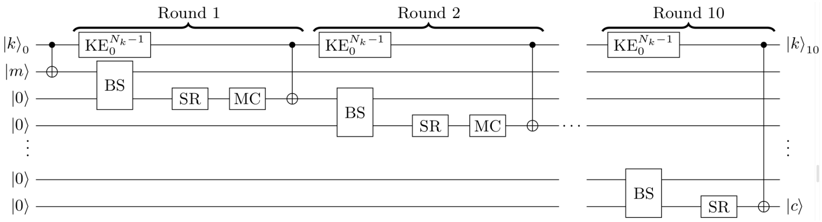


Figure: QAES-128 [Jaq+20]

Each wire represents 4 words (128 qubits). $|k\rangle$ represents the master key and m represents the message. BS is Sbox, SR is shift rows and MC is mix column. Here we have used an in-place version of mix columns.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	2,91,150	83,116	54,400	13,600	120	2,827	1,785	50,46,195
QAES-128 [Max19][25]	2,93,730	83,236	54,400	13,600	120	2,094	2,937	61,50,078

Table: QAES-128 cost of both variants of mix columns

[25] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES-128 circuit

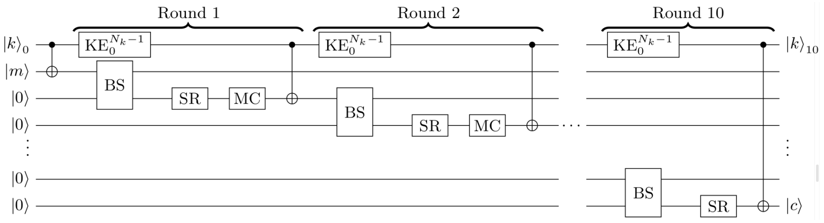


Figure: QAES-128 [Jaq+20]

Each wire represents 4 words (128 qubits). $|k\rangle$ represents the master key and m represents the message.BS is Sbox, SR is shift rows and MC is mix column. Here we have used an in-place version of mix columns.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	2,91,150	83,116	54,400	13,600	120	2,827	1,785	50,46,195
QAES-128 [Max19][25]	2,93,730	83,236	54,400	13,600	120	2,094	2,937	61,50,078

Table: QAES-128 cost of both variants of mix columns

[25] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES-128 circuit

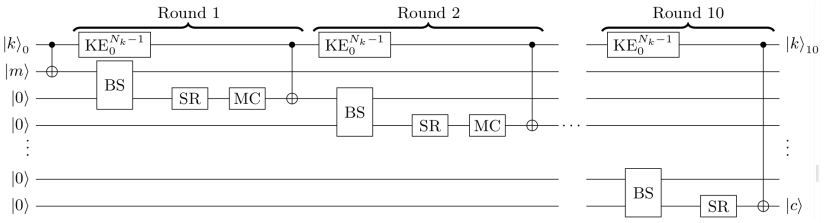


Figure: QAES-128 [Jaq+20]

Each wire represents 4 words (128 qubits). $|k\rangle$ represents the master key and m represents the message.BS is Sbox, SR is shift rows and MC is mix column. Here we have used an in-place version of mix columns.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	2,91,150	83,116	54,400	13,600	120	2,827	1,785	50,46,195
QAES-128 [Max19][25]	2,93,730	83,236	54,400	13,600	120	2,094	2,937	61,50,078

Table: QAES-128 cost of both variants of mix columns

[25] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

QAES-128 circuit

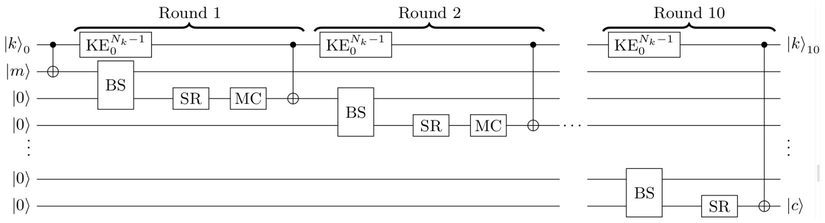


Figure: QAES-128 [Jaq+20]

Each wire represents 4 words (128 qubits). $|k\rangle$ represents the master key and m represents the message.BS is Sbox, SR is shift rows and MC is mix column. Here we have used an in-place version of mix columns.

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	2,91,150	83,116	54,400	13,600	120	2,827	1,785	50,46,195
QAES-128 [Max19][25]	2,93,730	83,236	54,400	13,600	120	2,094	2,937	61,50,078

Table: QAES-128 cost of both variants of mix columns

[25] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

Grover's Attack

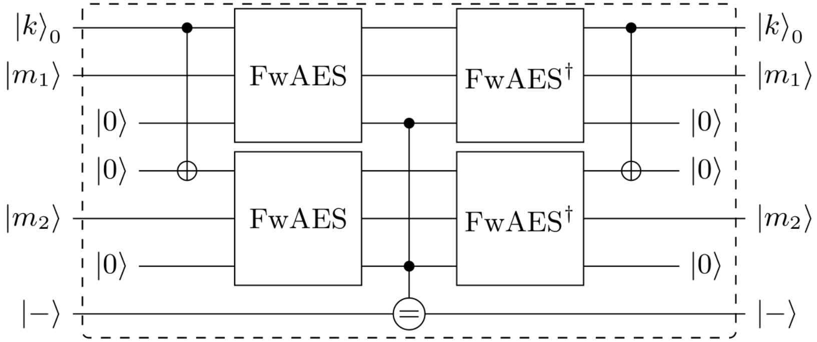


Figure: Grover's Attack on QAES-128 [Jaq+20]

MC	CNOT	Clifford	T	M	T-depth	full depth	width	DW
QAES-128 In place	5,85,051	1,69,184	1,09,820	27,455	121	2,815	3,329	93,71,135
QAES-128 [Max19][26]	5,89,643	1,68,288	1,09,820	27,455	121	2,096	5,633	1,18,06,768

Table: QAES-128 Grover's Oracle cost of both variants of mix columns (r=2)

[26] Alexander Maximov. "AES MixColumn with 92 XOR gates". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$

$G_D = 2,815 \approx 1.37 \times 2^{11}$

$G_W = 3,329 \approx 1.62 \times 2^{11}$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$

$G_D = 2,815 \approx 1.37 \times 2^{11}$

$G_W = 3,329 \approx 1.62 \times 2^{11}$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates

Let's calculate the cost estimates of Grover's Attack on QAES-128 in place with r = 2 with and without depth constraint.

$$G_G = 5,85,051 + 1,69,184 + 1,09,820 + 27,455 = 8,91,510 \approx 1.7 \times 2^{19}$$
$$G_D = 2,815 \approx 1.37 \times 2^{11}$$
$$G_W = 3,329 \approx 1.62 \times 2^{11}$$

Therefore the cost estimates without depth constraints keeping $S = 1$ is:

$$D \approx 1.37 \times 2^{11} \times 2^{64} = 1.37 \times 2^{75}$$
$$G \approx 1.7 \times 2^{19} \times 2^{64} = 1.7 \times 2^{83}$$
$$DW \approx 1.37 \times 2^{11} \times 1.62 \times 2^{11} \times 2^{64} \approx 1.1 \times 2^{87}$$

Similarly, the cost estimates with depth constraint of 2^{40} are:

$$S \approx 2^{128} \times 1.37^2 \times 2^{22} \times 2^{-80} \approx 1.87 \times 2^{70}$$
$$G \approx 2^{128} \times 1.37 \times 2^{11} \times 1.7 \times 2^{19} \times 2^{-40} \approx 1.16 \times 2^{119}$$
$$DW \approx 2^{128} \times 1.37^2 \times 2^{22} \times 1.62 \times 2^{11} \times 2^{-40} \approx 1.52 \times 2^{122}$$

Cost Estimates Contd.

- NIST[16][27] has proposed a maximum of 2^{170}/MAXDEPTH quantum gates for AES-128 but this does not take into account the effects of parallelization.
- For MAXDEPTH $= 2^{40}$, [16] has bounded the count of quantum gates by $2^{170}/2^{40} = 2^{130}$.
- From the above calculation of G-cost we can see that the number of gates required by AES-128 is much less after parallelization (2^{119}).

[27] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Cost Estimates Contd.

- NIST[16][27] has proposed a maximum of 2^{170}/MAXDEPTH quantum gates for AES-128 but this does not take into account the effects of parallelization.
- For MAXDEPTH $= 2^{40}$, [16] has bounded the count of quantum gates by $2^{170}/2^{40} = 2^{130}$.
- From the above calculation of G-cost we can see that the number of gates required by AES-128 is much less after parallelization (2^{119}).

[27] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Cost Estimates Contd.

- NIST[16][27] has proposed a maximum of 2^{170}/MAXDEPTH quantum gates for AES-128 but this does not take into account the effects of parallelization.
- For MAXDEPTH $= 2^{40}$, [16] has bounded the count of quantum gates by $2^{170}/2^{40} = 2^{130}$.
- From the above calculation of G-cost we can see that the number of gates required by AES-128 is much less after parallelization (2^{119}).

[27] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

Outline

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: http://www.libquantum.de/.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: http://www.libquantum.de/.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf`.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: `http://www.libquantum.de/`.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. `https://github.com/Gopal-Dahale/QSKC-Grover`. 2022.

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: http://www.libquantum.de/.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: http://www.libquantum.de/.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

Conclusion

- We briefly studied hardware and software-friendly ciphers.
- We designed Quantum circuits for SAES, SIMON, PRESENT and optimized on qubits.
- As a result, the quantum architecture is not a barrier to a quantum adversary carrying out any potential quantum attack.
- We discussed the parallelization of Grover's algorithm and its cost estimates which are used in modeling the quantum circuit and estimating the quantum resources for AES-128.
- AES-128 was studied with and without depth constraints under the rules proposed by NIST [16][28].
- We implemented SAES and QSAES in python using Qiskit[ANI+21][29] and libquantum[Wei][30] to verify the results. We were unable to run a successful Grover's Attack as it was time-consuming. The code and obtained results are open-sourced [Dah22][31] .

[28] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[29] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[30] Hendrik Weimer. URL: http://www.libquantum.de/.

[31] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

References I

[Zal99] Christof Zalka. “Grover’s quantum searching algorithm is optimal”. In: *Phys. Rev. A* 60 (4 Oct. 1999), pp. 2746–2751. DOI: 10.1103/PhysRevA.60.2746. URL: https://link.aps.org/doi/10.1103/PhysRevA.60.2746.

[Vik07] A. BogdanovL. R. KnudsenG. LeanderC. PaarA. PoschmannM. J. B. RobshawY. SeurinC. Vikkelsoe. “PRESENT: An Ultra-Lightweight Block Cipher”. In: (2007). URL: https://link.springer.com/chapter/10.1007/978-3-540-74735-2_31.

[Che+08] Donny Cheung et al. “On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography”. In: *Theory of Quantum Computation, Communication, and Cryptography*. Ed. by Yasuhito Kawano and Michele Mosca. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 96–104. ISBN: 978-3-540-89304-2.

References II

[BP10] Joan Boyar and René Peralta. “A New Combinational Logic Minimization Technique with Applications to Cryptology”. In: May 2010, pp. 178–189. ISBN: 978-3-642-13192-9. DOI: 10.1007/978-3-642-13193-6_16.

[BP11] Joan Boyar and René Peralta. “A depth-16 circuit for the AES S-box.”. In: vol. 2011. Jan. 2011, p. 332. ISBN: 978-3-642-30435-4. DOI: 10.1007/978-3-642-30436-1_24.

[Bea+13] Ray Beaulieu et al. *The SIMON and SPECK Families of Lightweight Block Ciphers*. Cryptology ePrint Archive, Report 2013/404. https://ia.cr/2013/404. 2013.

[Gra+15] Markus Grassl et al. “Applying Grover’s algorithm to AES: quantum resource estimates”. In: (Dec. 2015).

References III

[16] *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[Alm+18] Mishal Almazrooie et al. "Quantum Grover Attack on the Simplified-AES". In: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA 2018. Kuantan, Malaysia: Association for Computing Machinery, 2018, pp. 204–211. ISBN: 9781450354141. DOI: 10.1145/3185089.3185122. URL: https://doi.org/10.1145/3185089.3185122.

[KHJ18] Panjin Kim, Daewan Han, and Kyung Jeong. "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2". In: *Quantum Information Processing* 17 (Oct. 2018). DOI: 10.1007/s11128-018-2107-3.

References IV

[Das+19] Vishnu Asutosh Dasu et al. “LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes”. In: *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. 2019, pp. 260–265. DOI: 10.1109/SOCC46988.2019.1570548320.

[Max19] Alexander Maximov. “AES MixColumn with 92 XOR gates”. In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 833.

[AMM20] Ravi Anand, Arpita Maitra, and Sourav Mukhopadhyay. “Grover on SIMON”. In: Apr. 2020.

[Jaq+20] Samuel Jaques et al. “Implementing Grover Oracles for Quantum Key Search on AES and LowMC”. In: May 2020, pp. 280–310. ISBN: 978-3-030-45723-5. DOI: 10.1007/978-3-030-45724-2_10.

[ANI+21] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

References V

[Jan+21a] Kyung-Bae Jang et al. "Grover on Simplified AES". In: *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2021, pp. 1–4. DOI: 10.1109/ICCE-Asia53811.2021.9642017.

[Jan+21b] Kyungbae Jang et al. "Efficient Implementation of PRESENT and GIFT on Quantum Computers". In: *Applied Sciences* 11.11 (2021). ISSN: 2076-3417. DOI: 10.3390/app11114776. URL: https://www.mdpi.com/2076-3417/11/11/4776.

[Qua21] IBM Quantum. *IBMQ Simulators*. 2021. URL: https://quantum-computing.ibm.com/.

[con22a] Wikipedia contributors. *Simon (cipher)*. Wikipedia, 2022. URL: https://en.wikipedia.org/wiki/Simon_(cipher).

[con22b] Wikipedia contributors. *Toffoli Gate*. Wikipedia, 2022. URL: https://en.wikipedia.org/wiki/Toffoli_gate.

References VI

[Dah22] Gopal Ramesh Dahale. *QSKC-Grover*. https://github.com/Gopal-Dahale/QSKC-Grover. 2022.

[Gor22] Steven Gordon. *Cryptography Study Notes (Chapter 9)*. 2022. URL: https://sandilands.info/crypto/.

[Wei] Hendrik Weimer. URL: http://www.libquantum.de/.