Grover on Sparkle

Yujin Yang, Kyoungbae Jang, Hyunji Kim, Gyeongju Song, and Hwajeong Seo
Hansung University
yujin.yang34@gmail.com

HANSUNG UNIVERSITY

CryptoCraft LAB

- Introduction (Motivation, Contribution)
- Background (Quantum Gates, Grover's Algorithm for key search)
- Body (Sparkle SCHWAEMM 128-128, Cost estimation)
- Conclusion

Motivation (Introduction)

- **Quantum computers develop and Grover search algorithm appeared**
  - Grover search algorithm can reduce the complexity of searching for a secret key by as much as square root($\sqrt{\quad}$) in a symmetric cryptosystem.
  - The safety of ciphers based on these hard problems is threatened.
- **Studies are underway to analyze threats to the Grover algorithm for symmetric cryptosystem**
  - The field of research has expanded to lightweight ciphers in recent years.
- **KNOT is the only quantum implementation of AEAD**
  - Implementation of AEAD of SPARKLE(NIST LWC final candidate) as a quantum circuit

Contribution (Introduction)

- **Reported first quantum implementation of all parameters of SCHWAEMM**
  - SCHWAEMM is AEAD of lightweight block cipher SPARKLE

- **Optimized the quantum circuit by reducing the depth and the number of qubits**
  - Used inverse operations and fake padding
  - Implemented quantum additions in parallel

- **Estimated the quantum resources & evaluated the post-quantum security level**
  - Based on NIST security requirements
  - Applied Grover's search algorithm to proposed quantum circuit

Quantum Gates (Background)

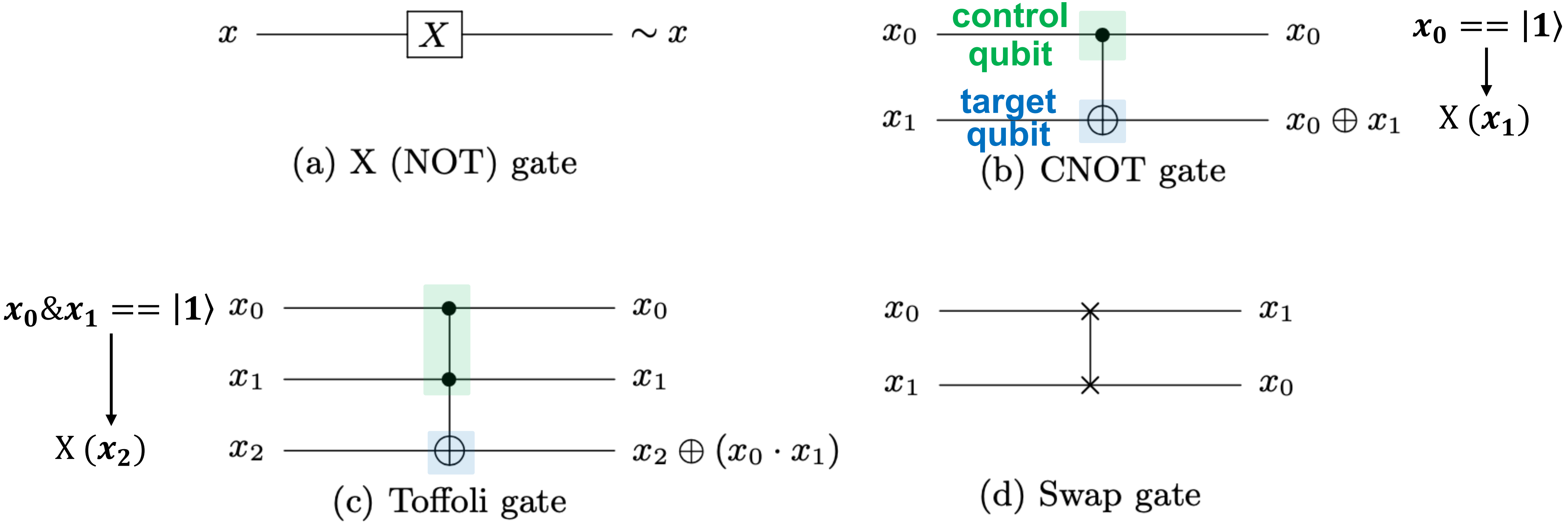
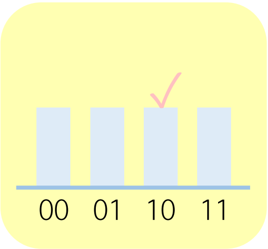


Fig. 1: Quantum gates

Grover's Algorithm for key search (Background)

1. Through the use of Hadamard gates, n-qubit key has the same amplitude at all state of the qubits

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle$$



2. $f(x) = 1$, sign of the solution key is changed to negative

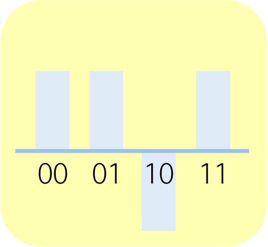
superpositioned key

Oracle

$$f(x) = \begin{cases} 1 \text{ if } Enc(key) = c \\ 0 \text{ if } Enc(key) \neq c \end{cases}$$

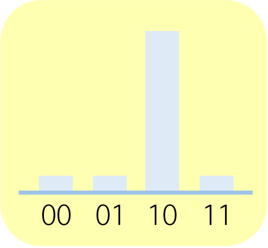
known ciphertext

ciphertext —— comparison



3. Amplify the amplitude of the negative sign state

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle|-\rangle$$



SPARKLE Permutation 1) Alzette

Algorithm 3 Quantum implementation of SPAKRLE256$_r$.

Input: 128-qubit $x_{0\sim3}$, and $y_{0\sim3}$, Adder carry $ac_{0\sim3}$, Constant $c_{0\sim7}$
Output: x, y

for $i = 0$ to r **do**
  $y_0 \leftarrow AddConstant(y_0,\ c_{(i\%8)})$
  $y_1 \leftarrow AddConstant(y_1,\ i)$
  `// Parallel Azlettes`
  $(x_0,\ y_0) \leftarrow \text{Alzette}(x_0,\ y_0,\ c_0,\ ac_0)$
  $(x_1,\ y_1) \leftarrow \text{Alzette}(x_1,\ y_0,\ c_1,\ ac_1)$
  $(x_2,\ y_2) \leftarrow \text{Alzette}(x_2,\ y_0,\ c_2,\ ac_2)$
  $(x_3,\ y_3) \leftarrow \text{Alzette}(x_3,\ y_0,\ c_3,\ ac_3)$
  `// Linear Diffusion Layer`
  $(x_{0\sim3},\ y_{0\sim3}) \leftarrow \mathcal{L}_4(x_{0\sim3},\ y_{0\sim3})$
end for
return x, y

allocate 4 carry qubits
→ reduce depth

Algorithm 1 Quantum implementation of Alzette.

Input: 32-qubit x and y, Constant c, Adder carry ac
Output: x, y

$x \leftarrow ADD((y \ggg 31),\ x,\ ac)$
$y \leftarrow \text{CNOT32}((x \ggg 24),\ y)$
$x \leftarrow AddConstant(x,\ c)$
$x \leftarrow ADD((y \ggg 17),\ x,\ ac)$
$y \leftarrow \text{CNOT32}((x \ggg 17),\ y)$
$x \leftarrow AddConstant(x,\ c)$
$x \leftarrow ADD(y,\ x,\ ac)$
$y \leftarrow \text{CNOT32}((x \ggg 31),\ y)$
$x \leftarrow AddConstant(x,\ c)$
$x \leftarrow ADD((y \ggg 24),\ x,\ ac)$
$y \leftarrow \text{CNOT32}((x \ggg 16),\ y)$
$x \leftarrow AddConstant(x,\ c)$
return x, y

1-branch

$$x \leftarrow x + (y \ggg 31)$$
$$y \leftarrow y \oplus (x \ggg 24)$$
$$x \leftarrow x \oplus c$$

Addition – CDKM ripple-carry adder
Rotation – logical Swap
XOR – CNOT gate

SPARKLE Permutation 2) Diffusion operator

Algorithm 3 Quantum implementation of SPAKRLE2

Input: 128-qubit $x_{0\sim3}$, and $y_{0\sim3}$, Adder carry $ac_{0\sim3}$, Const

Output: x, y

```
for i = 0 to r do
    y_0 ← AddConstant(y_0, c_(i%8))
    y_1 ← AddConstant(y_1, i)

    // Parallel Azlettes
    (x_0, y_0) ← Alzette(x_0, y_0, c_0, ac_0)
    (x_1, y_1) ← Alzette(x_1, y_0, c_1, ac_1)
    (x_2, y_2) ← Alzette(x_2, y_0, c_2, ac_2)
    (x_3, y_3) ← Alzette(x_3, y_0, c_3, ac_3)

    // Linear Diffusion Layer
    (x_0~3, y_0~3) ← L_4(x_0~3, y_0~3)
end for
return x, y
```

Algorithm 2 Quantum implementation of $\mathcal{L}_4$.

Input: 128-qubit $x_{0\sim3}$, and $y_{0\sim3}$

Output: x, y

```
// Feistel round
Transform x_0:                                  ▷ Compute()
    x_0 ← CNOT32(x_1, x_0)
    x_0L ← CNOT16(x_0R, x_0L)
y_2R ← CNOT16(x_0L, y_2R)
y_2L ← CNOT16(x_0R, y_2L)
y_2 ← CNOT32(y_0, y_2)
y_3R ← CNOT16(x_0L, y_3R)
y_3L ← CNOT16(x_0R, y_3L)
y_3 ← CNOT32(y_1, y_3)
Reverse(transform x_0)                          ▷ Uncompute()

Transform y_0:                                  ▷ Compute()
    y_0 ← CNOT32(y_1, y_0)
    y_0L ← CNOT16(y_0R, y_0L)
x_2R ← CNOT16(y_0L, x_2R)
x_2L ← CNOT16(y_0R, x_2L)
x_2 ← CNOT32(x_0, x_2)
x_3R ← CNOT16(y_0L, x_3R)
x_3L ← CNOT16(y_0R, x_3L)
x_3 ← CNOT32(x_1, x_3)
Reverse(transform y_0)                          ▷ Uncompute()

// Branch permutation
(x_0, x_2) ← SWAP32(x_0, x_2)
(x_1, x_3) ← SWAP32(x_1, x_3)
(y_0, y_2) ← SWAP32(y_0, y_2)
(y_1, y_3) ← SWAP32(y_1, y_3)
(x_0, x_1) ← SWAP32(x_0, x_1)
(y_0, y_1) ← SWAP32(y_0, y_1)
return x, y
```

$t_x \leftarrow (t_x \oplus (t_x \lll 16) \lll 16)$

calculate only necessary gates
→ save CNOT gates

Inverse operator
→ save 2 qubits

$t_x = x_0, t_y = y_0$

SWAP gate

SCHWAEMM128-128

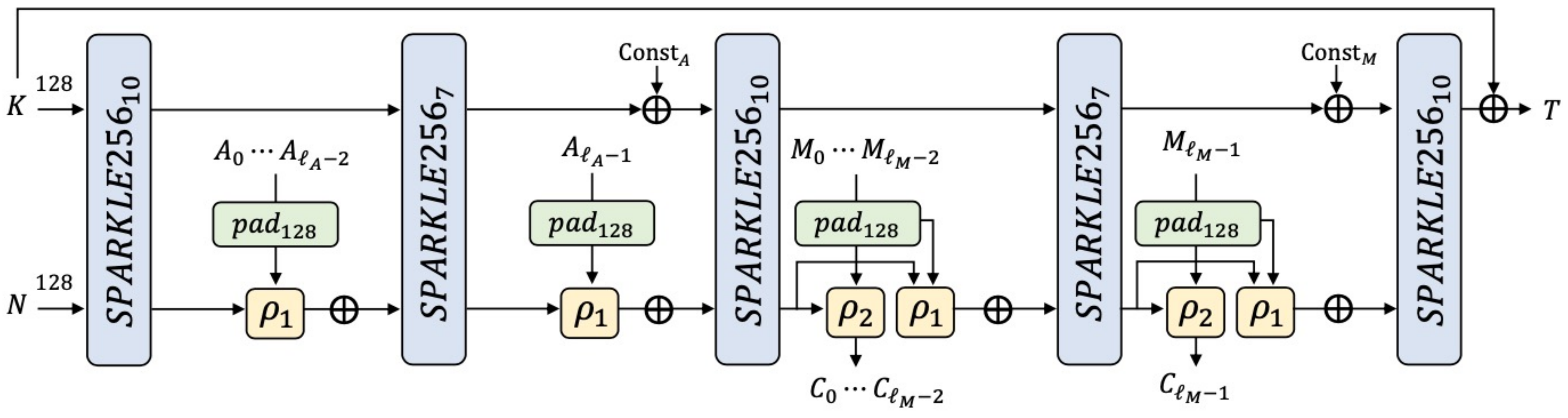


Fig. 2: Process of SCHWAEMM128-128

Padding Data **Processing of associated data** **Finalization**

State Initialization **Encrypting**

Padding Data & State Initialization

Padding Data

r: length of block ($i = -|M| - 1 \; mod \; r$)

$$Pad_r(M) = M||00000001||0^i$$

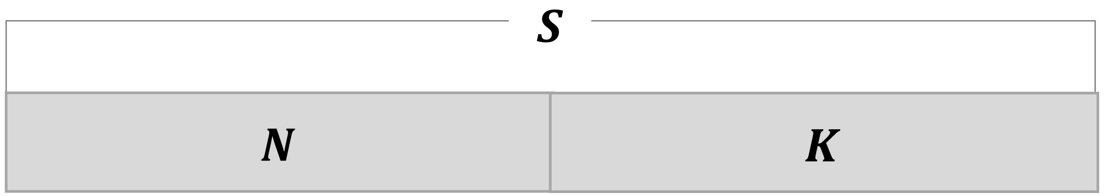
- The padded Associated data/Message are only used in ρ_1 function $\longrightarrow$ "fake padding"
- Don't allocate padding qubits to compute only useful data in ρ_1 $\longrightarrow$ save 32 CNOT gates

State Initialization

Inner state S, Nonce N, Key K

- use CNOT gates

$$S = N||K$$



Processing of associated data

Processing of associated data

Algorithm 4 Quantum implementation of processing of associated data.

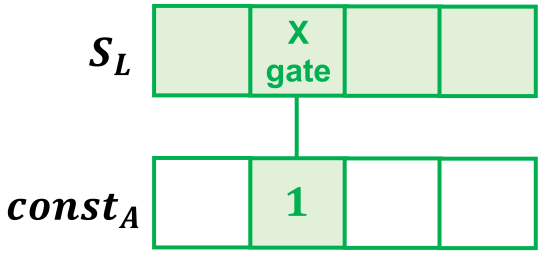
Input: State S, Associated data A, Constant of A $const_A$
Output: S

$S_R \leftarrow AddConstant(const_A,\ S_R)$

// $\rho_1(S_L,\ A)$
$S_L \leftarrow \text{SWAP64}(S_{L1},\ S_{L2})$
$S_{L1} \leftarrow \text{CNOT64}(S_{L2},\ S_{L1})$
$S_L \leftarrow \text{CNOT32}(A,\ S_L)$
$S_L \leftarrow \text{X}(S_L)$
$S_L \leftarrow \text{CNOT128}(S_R,\ S_L)$

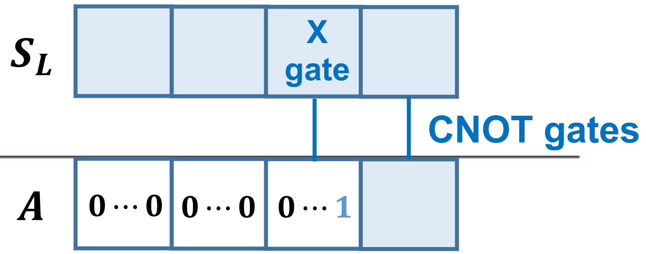
// SPRAKLE Permutation
$S \leftarrow \text{SPARKLE256}_{10}(S)$
return S

calculate only corresponding point (CNOT gates → X gate)
→ save gate resources & qubits



calculate only corresponding point
1 qubit(padded data ‘1’): CNOT gates → X gate

→ save CNOT gates



Encrypting & Finalization

Encrypting

Finalization

Algorithm 5 Quantum implementation of encrypting and finalization.

Input: State S, Message M, Constant of M $const_M$, Key K, Ciphertext C

Output: $C||S_R$

Encrypting:

$//C \leftarrow trunc_t(\rho_2(S_L, M)))$

$C \leftarrow$ Allocate new qubits of length $|M|$

$AddConstant(M(\text{classical}), C)$

$C \leftarrow \text{CNOT32}(S_L, C)$

[copy M(classical) to C] use X gates instead of CNOT gates

→ save gate resources

Finalization:

$S_R \leftarrow AddConstant(const_M, S_R)$

calculate only corresponding point (CNOT gates → X gate)

→ save gate resources & qubits

$//\ \rho_1(S_L, M)$

$S_L \leftarrow \text{SWAP64}(S_{L1}, S_{L2})$

$S_{L1} \leftarrow \text{CNOT64}(S_{L2}, S_{L1})$

$S_L \leftarrow \text{CNOT32}(M, S_L)$

$S_L \leftarrow \text{X}(S_L)$

$S_L \leftarrow \text{CNOT128}(S_R, S_L)$

//SPARKLE Permutation

$S \leftarrow \text{SPARKLE256}_{10}(S)$

$//S_R \oplus K$

$S_R \leftarrow \text{CNOT128}(K, S_R)$

return $C||S_R$

Cost Estimation for Grover Key Search

<Oracle operation>

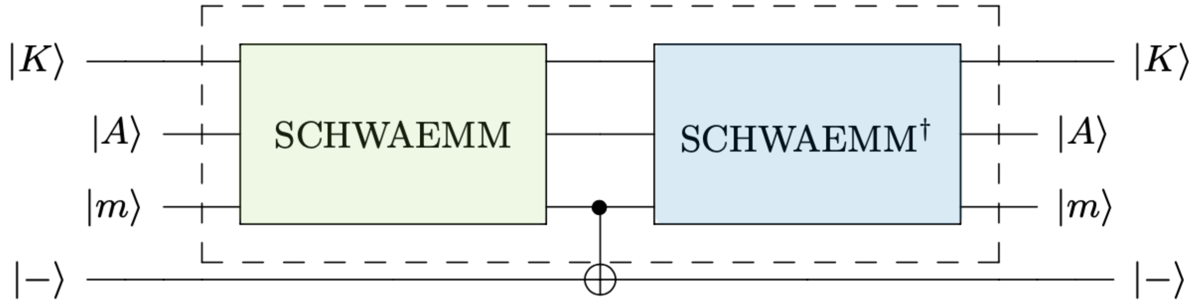


Fig. 3: Grover's oracle on SCHWAEMM

Encryption operation → Reverse operation

⇒ SCHWAEMM quantum circuit works X 2

Cost Estimation for Grover Key Search

Table 1: Quantum resources required for SCHWAEMM quantum circuits in detail.

Cipher	#CNOT	#1qCliff	$\#T$	T-depth	#qubits	Full depth
SCHWAEMM-128/128	278,656	94,511	204,960	9,760	612	59,687
SCHWAEMM-256/128	460,744	156,497	338,184	10,736	870	65,783
SCHWAEMM-192/192	460,680	156,497	338,184	10,736	870	65,783
SCHWAEMM-256/256	670,688	227,606	491,904	11,712	1,128	71,906

X 2

Table 2: Quantum resources required for Grover's oracle on SCHWAEMM

Cipher	#CNOT	#1qCliff	$\#T$	T-depth	#qubits	Full depth
SCHWAEMM-128/128	557,312	189,022	409,920	19,520	613	119,374
SCHWAEMM-256/128	921,488	312,994	676,368	21,472	871	131,566
SCHWAEMM-192/192	921,360	312,994	676,368	21,472	871	131,566
SCHWAEMM-256/256	1,341,376	455,212	983,808	23,424	1,129	143,812

Cost Estimation for Grover Key Search

Table 3: Quantum resources required for Grover's key search on SCHWAEMM

Cipher	Total gates	Total depth	Cost	NIST security	
SCHWAEMM-128/128	$1.732 \cdot 2^{83}$	$1.431 \cdot 2^{80}$	$1.239 \cdot 2^{164}$	2^{170}	**AES-128**
SCHWAEMM-256/128	$1.431 \cdot 2^{84}$	$1.577 \cdot 2^{80}$	$1.128 \cdot 2^{165}$	2^{170}	
SCHWAEMM-192/192	$1.431 \cdot 2^{116}$	$1.577 \cdot 2^{112}$	$1.128 \cdot 2^{229}$	2^{233}	**AES-192**
SCHWAEMM-256/256	$1.041 \cdot 2^{149}$	$1.723 \cdot 2^{144}$	$1.795 \cdot 2^{293}$	2^{298}	**AES-256**

Attack cost = Total gates X Total depth

[NIST's post-quantum security requirements]

Ciphers should be comparable to or higher than the Grover attack cost for AES

Cost Estimation for Grover Key Search

Table 3: Quantum resources required for Grover's key search on SCHWAEMM

Cipher	Total gates	Total depth	Cost		NIST security	
SCHWAEMM-128/128	$1.732 \cdot 2^{83}$	$1.431 \cdot 2^{80}$	$1.239 \cdot 2^{164}$	$<$	2^{170}	AES-128
SCHWAEMM-256/128	$1.431 \cdot 2^{84}$	$1.577 \cdot 2^{80}$	$1.128 \cdot 2^{165}$	$<$	2^{170}	
SCHWAEMM-192/192	$1.431 \cdot 2^{116}$	$1.577 \cdot 2^{112}$	$1.128 \cdot 2^{229}$	$<$	2^{233}	AES-192
SCHWAEMM-256/256	$1.041 \cdot 2^{149}$	$1.723 \cdot 2^{144}$	$1.795 \cdot 2^{293}$	$<$	2^{298}	AES-256

Attack is possible with fewer quantum resources

⇒ Appropriate security level cannot be achieved

SCHWAEMM is exposed to attack at a lower cost than AES with same key size.

Cost Estimation for Grover Key Search

<Table 3> NIST security

- AES attack cost estimated by NIST is the result of 2016

- If the results of significantly reducing the quantum attack cost are presented, the estimated cost in post-quantum security requirements should be conservatively evaluated.

- Recently, Implementations for optimizing quantum circuits for AES have been proposed.

- Compared with the attack cost for Jaques et al's AES estimated, SCHWAEMM achieves an appropriate.

Conclusion

Conclusion

- Our implementation has been optimized by applying various techniques to minimize the cost
- Focused on reducing the depth complexity as well as the qubit complexity.

Future work

- Analyzing the cost of Grover's attack for other final candidate algorithms of NIST LWC
- Evaluating the post-quantum security strength for other final candidate algorithms of NIST LWC

yujin.yang34@gmail.com

Q & A