Quantum Implementation of Encoding Algorithm for HQC

Yujin Oh, Sejin Lim, Kyungbae Jang and Hwajeong Seo

한성대학교
HANSUNG UNIVERSITY

CryptoCraft LAB

Introduction

Background

Proposed Method

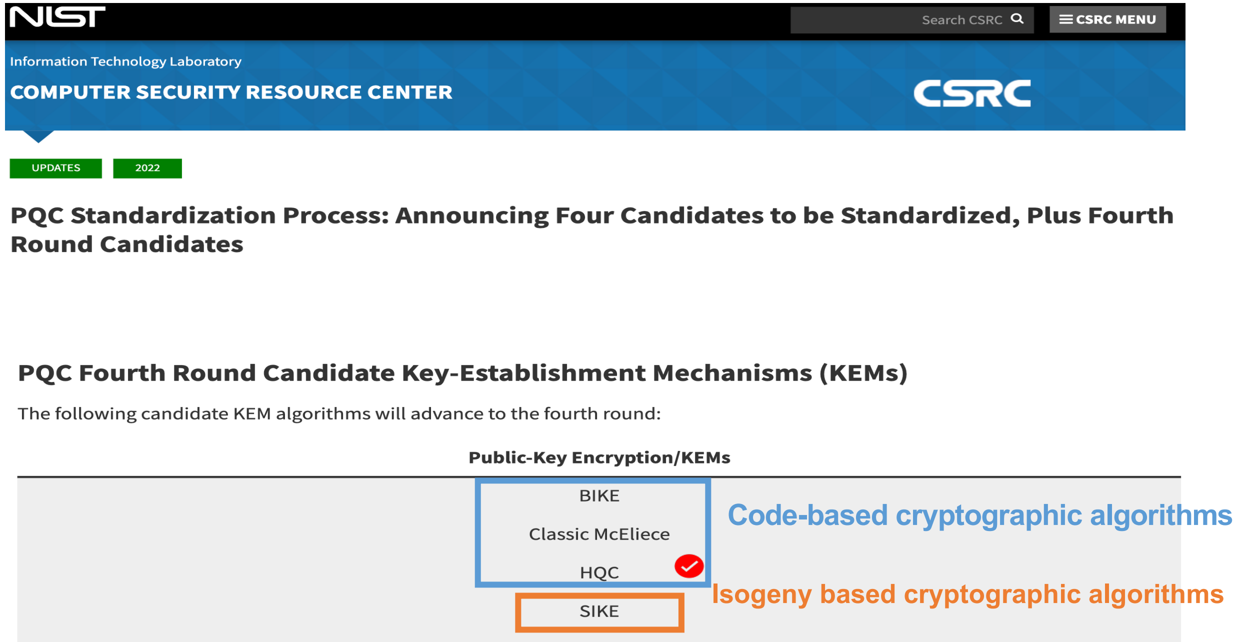
Conclusion

Introduction

- The security of current encryption systems is being threatened by quantum algorithms.
- As a result, NIST organized the **Post-Quantum Cryptography Standardization competition**.
- To establish a **quantum-resistant cryptographic system**,

  → It is essential to implement **quantum circuits** **to reassess the security** of cryptographic algorithms.
- In light of this, our paper focuses on **optimizing the quantum circuit implementation technique for HQC**,

  → One of the candidate algorithms in **NIST's Post-Quantum Cryptography Standardization Round 4**.



Background

- **HQC(Hamming Quasi-Cyclic)**
  - **Code-based cryptography** (Hamming codes with randomly generated Quasi-Cyclic codes).
  - **Quasi-Cyclic** ensures that certain relationships cycle within the matrix, optimizing operations.

    → It is possible to know the entire matrix by **only storing the first row**, efficiently **reducing the key size**.
  - The **PKE** version of HQC consists of three main processes

    → $key\ generation\ (sk\ =\ (x,y)), encryption\ (encoding)\ (ct\ =\ (u,v)), and\ decryption\ (decoding).$
  - Decoding is impossible because **the error *e*** added during encryption is **very large**

    → But only the user who has **the secret key** can easily decode by reducing ***e***
  - Decoding may fail with probability, but in the HQC paper, the authors demonstrate through a detailed and precise mathematical analysis that the **probability of failure is negligibly low** → **offer high security**

Proposed Method

- In this paper, due to the imperfect implementation of $\boldsymbol{u}$ and $\boldsymbol{v}$ used as ciphertext

  → They are separately implemented and the separated resources are estimated.

  - $\boldsymbol{u} \leftarrow \boldsymbol{r_1} + \boldsymbol{hr_2}$
  - $\boldsymbol{v} \leftarrow \boldsymbol{mG} + \boldsymbol{sr_2} + \boldsymbol{e}$

- HQC- 128, operations are carried out in the binary field of $\mathbb{F}_{2^{17668}}$.

- Due to the limitations of quantum simulation, we implement the quantum circuit by **reducing** $\mathbb{F}_{2^{12}}(\mathbb{F}_{2^{17668}} \rightarrow \mathbb{F}_{2^{12}})$
  - Primitive polynomial : $\mathbb{F}_{2^{12}}\ /(x^{11} + x^{10} + x^9 + \cdots + x + 1)$ (Derived from $(X^n - 1)\ /\ (X - 1)$

- In the operation m·G, a shortened **Reed-Solomon** code is utilized in the binary field of $\mathbb{F}_{2^8}$.
  - Primitive polynomial : $\mathbb{F}_{2^8}/(\mathrm{x}^8 + \mathrm{x}^4 + \mathrm{x}^3 + \mathrm{x}^2 + \mathrm{x}^1)$

Proposed Method

- **Multiplication**
  - We apply **WISA'22 [Jang et al.] multiplication**

    → optimized with a **Toffoli depth of one** for any field size.

  - WISA'22[Jang et al.] multiplication
    - Using the **Karatsuba algorithm recursively** and allocating additional ancilla qubits.

      → All the AND operations become independent and the operations of **all Toffoli gates in parallel.**

      → The allocated ancilla qubits can be **reused** through **reverse operations**.

[Jang et al.] K. Jang, W. Kim, S. Lim, Y. Kang, Y. Yang, and J. Hwa, "Optimized implementation of quantum binary field multiplication with toffoli depth one," in International Conference on Information Security Applications.

Proposed Method

- **Addition**
  - Addition is same as XOR operation in modular operations.

    →Only CNOT gate
  - Using these **binary field operations**

    → We can calculate $\boldsymbol{u} \leftarrow \boldsymbol{r_1} + \boldsymbol{hr_2}$, $v \leftarrow mG + \boldsymbol{sr_2} + \boldsymbol{e}$

Table 1: Required quantum resources for Binary Field Operations

Field	Arithmetic	Qubits	#CNOT	#Toffoli	Toffoli depth	Full depth
$\mathbb{F}_{2^8}$	Multiplication	81	164	27	1	26
$\mathbb{F}_{2^{12}}$	Addition	24	12	-	-	1
	Multiplication	162	495	54	1	32

Table 2: Required quantum resources for implementing $u \leftarrow r_1 + h \cdot r_2$

Field	Operation	Qubits	#CNOT	#Toffoli	Toffoli depth	Full depth
$\mathbb{F}_{2^{12}}$	$u(r_1 + h \cdot r_2)$	174	507	54	1	33

Proposed Method

- **Shortened Reed-Solomon algorithm**
  - The operation **m · G** involves more than simple multiplication

    → and one of the method used for this operation is **Shortened Reed-Solomon**.

    → $v \leftarrow \boldsymbol{mG} + sr_2 + e$

  - In this algorithm, the crucial operation involves binary field operations

    → multiply the **coefficient matrix** of the publicly available **RS-S1 polynomial** by the message vector.

  - The constant values used in the algorithm are denoted as

$constant$	K	G	$N1$
$values$	16	31	46

Proposed Method

- **Shortened Reed-Solomon algorithm**
  - Using WISA'22 multiplication

    → optimized with a **Toffoli-depth of 1**

  - The function $Copy\_gate\_value$ involves copying the $gate\_value$

    → 30 subsequent multiplications **in parallel**.

  - **All multiplication with a Toffoli-depth of 1 by parallelization.**
  - Because it repeated 16, Total Toffoli-depth is 16.

Table 3: Required quantum resources for Shortened Reed-Solomon quantum circuit implementation

shortened Reed-Solomon	Qubits	#CNOT	#Toffoli	Toffoli depth	Full depth
HQC-128	28,696	94,320	12,960	16	545

Algorithm 2 Quantum circuit implementation of shortened Reed-Solomon.

Input: 8-qubit array $msg[K]$, $RS\_POLY[G-1]$, $cdw[N1]$, $gate\_value$, $copy[G-2]$, $ancilla$ qubits array $ac[30]$

Output: cdw

```
for i = 0 to G − 1 do
    RS_POLY[i] ← CNOT8(RS_COEFS, RS_POLY[i])
end for
for i = 0 to K do
    gate_value[i] ← CNOT8(cdw[N1 − K − 1], gate_value[i])
    gate_value[i] ← CNOT8(msg[K − 1 − i], gate_value[i])
    for j = 0 to G − 2 do
        copy[j] ← Copy_gate_value (gate_value[i], copy[j])
    end for
    tmp[0] ← Multiplication(gate_value[i], RS_POLY[0], ac[0])
    for j = 1 to G − 1 do
        tmp[j] ← Multiplication(copy[i], RS_POLY[j], ac[j])
    end for
    for j = N1 − K − 1 to 0 do
        cdw[j] ← CNOT8(cdw[j − 1], cdw[j])
        cdw[j] ← CNOT8(tmp[j], cdw[j])
    end for
    cdw[0] ← CNOT8(tmp[0], cdw[0])
    for j = 0 to G − 2 do
        copy[j] ← Copy_gate_value (gate_value[i], copy[j])
    end for
end for
for i = 0 to K do
    cdw[i] ← CNOT8(msg[i], cdw[i + 30])
end for
return cdw
```

Proposed Method

- **Shortened Reed-Solomon algorithm**
  - We utilize the **reverse** operation (Line 19-21 in Algorithm 2)

    → To **reuse** the qubit array ($copy[j]$),
  - The reverse operation employs only **CNOT gates**

    → With **minimal impact on full depth**
  - The **CNOT gates** of reverse operation are further **parallelized**

    → **resulting in an even more smaller impact on the full depth.**

Table 3: Required quantum resources for Shortened Reed-Solomon quantum circuit implementation

shortened Reed-Solomon	Qubits	#CNOT	#Toffoli	Toffoli depth	Full depth
HQC-128	28,696	94,320	12,960	16	545

Algorithm 2 Quantum circuit implementation of shortened Reed-Solomon.

Input: 8-qubit array $msg[K]$, $RS\_POLY[G-1]$, $cdw[N1]$, $gate\_value$, $copy[G-2]$, $ancilla$ qubits array $ac[30]$

Output: cdw

```
1:  for i = 0 to G − 1 do
2:      RS_POLY[i] ← CNOT8(RS_COEFS, RS_POLY[i])
3:  end for
4:  for i = 0 to K do
5:      gate_value[i] ← CNOT8(cdw[N1 − K − 1], gate_value[i])
6:      gate_value[i] ← CNOT8(msg[K − 1 − i], gate_value[i])
7:      for j = 0 to G − 2 do
8:          copy[j] ← Copy_gate_value (gate_value[i], copy[j])
9:      end for
10:     tmp[0] ← Multiplication(gate_value[i], RS_POLY[0], ac[0])
11:     for j = 1 to G − 1 do
12:         tmp[j] ← Multiplication(copy[i], RS_POLY[j], ac[j])
13:     end for
14:     for j = N1 − K − 1 to 0 do
15:         cdw[j] ← CNOT8(cdw[j − 1], cdw[j])
16:         cdw[j] ← CNOT8(tmp[j], cdw[j])
17:     end for
18:     cdw[0] ← CNOT8(tmp[0], cdw[0])
19:     for j = 0 to G − 2 do
20:         copy[j] ← Copy_gate_value (gate_value[i], copy[j])
21:     end for
22: end for
23: for i = 0 to K do
24:     cdw[i] ← CNOT8(msg[i], cdw[i + 30])
25: end for
26: return cdw
```

Conclusion

- In this paper, we propose a quantum circuit implementation for the core operations involving **binary field arithmetic** and shortened **Reed-Solomon code** in the **encoding process** of the HQC PKE version, a fourth-round candidate algorithm in the NIST competition.

  → It is expected that the presented quantum circuit will contribute to **the security analysis of HQC**.
- In the future, we plan to **complete the encoding** operations by implementing other operations.
- We will also implement the **key generation** and **decoding** to complete **the entire quantum circuit of HQC**.
- We plan to adjust the range for **feasible simulations** and **expand the binary field** to maximum extent.

Q & A