Shor on ECC 동향

https://youtu.be/nQQeI_5xsP4

Shor 알고리즘을 사용한 공개키 암호 해킹

- 1994년, 수학자 Shor가 난제인 소인수 분해, **이산대수를 다항시간내에 해결할 수 있는 양자 알고리즘**을 제안
  - $O(e^{1.9(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}}) \rightarrow O((\log N)^2(\log\log N)(\log\log\log N))$

지수 차원 복잡도 → **(Shor 알고리즘 적용 후) 다항시간내에 해결**

Algorithms for Quantum Computation:
Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

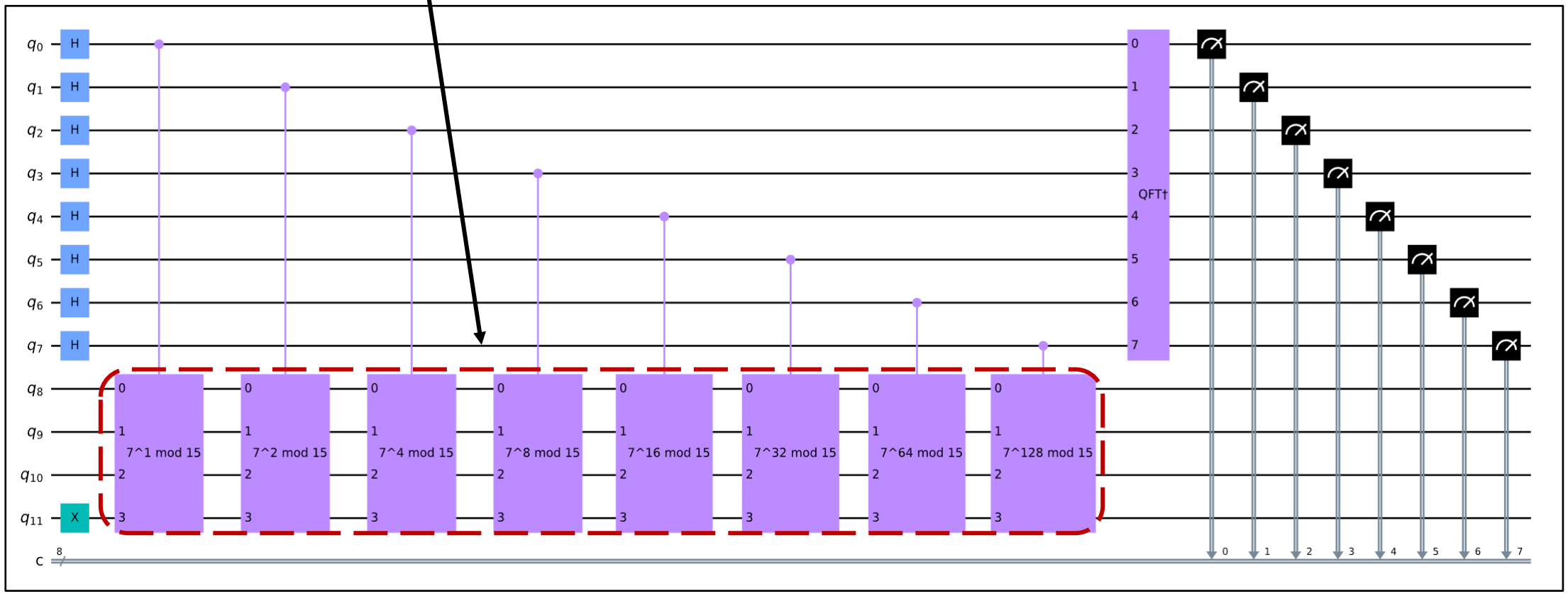
Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in com-

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was

Shor on RSA: Quantum Circuit

- Shor 알고리즘: Quantum Fourier Transform (QF), Quantum Phase (QPE), Inverse QFT로 구성 됨
  - 내부 **Modular Exponentiation 연산**의 최적화 (양자 컴퓨터 상에서)가 중요



< Shor 알고리즘 Factoring 회로 >

Shor on ECC

- **ECC의 경우, 내부 수식만이 ECDLP 타겟으로 바뀜**
  - 타원 곡선상에서의 $Q = [\alpha]P,\ \text{Secret} = \ \alpha,\ \alpha \in \{1, \ldots, r\}$

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k \bmod N\rangle$$

< Factoring에 대한 Shor 알고리즘 >

$$\frac{1}{2^{n+1}} \sum_{k,\ell=0}^{2^{n+1}-1} |k, \ell, [k]P + [\ell]Q\rangle.$$

< ECDLP에 대한 Shor 알고리즘 >

$$[2^0]P, [2^1]P, \ldots, [2^n]P \text{ and } [2^0]Q, [2^1]Q, \ldots, [2^n]Q$$

Elliptic Curve Discrete Logarithm Problem(ECDLP)

- **Elliptic Curve Diffie-Hellman**
  - 타원 곡선상에서의 **Secret Point를 공유하여 키 교환을 수행**하는 프로토콜
  - **Known**
    - Point P on Curve
  - **Secret**
    - Bob : Integer α
    - Alice : Integer β
  - **ECDLP**
    - $P_\alpha = [\alpha]P$일 때, P_α로부터 α를 알아내는 문제
    - Bob → $P_\alpha = [\alpha]P$를 Alice에게 전달
    - Alice → $P_\beta = [\beta]P$를 Bob에게 전달
    - Bob과 Alice만의 Shared Secret Point : $\boldsymbol{P_{\alpha\beta}} = [\alpha \cdot \beta]P = [\alpha]P_\beta = [\beta]P_\alpha$
      - **(전달받은 Point)** × **(자신의 Secret)**

Shor on ECC

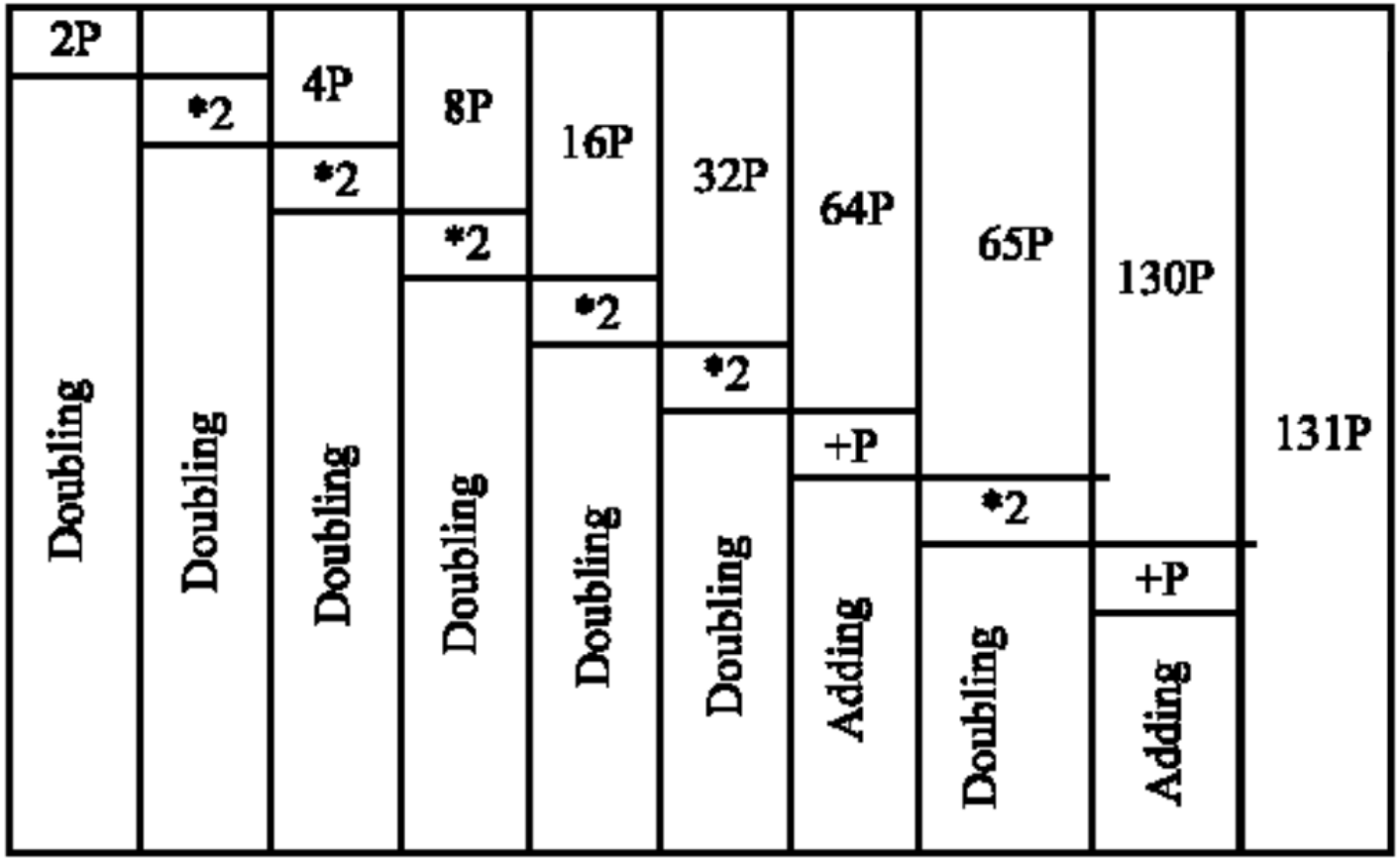
- **이산 대수 문제** (Discrete Logarithm Problems)에 안전성을 기반하는 ECC도 Shor 알고리즘을 활용한 양자 컴퓨터의 공격에 **안전성이 무너짐**

- **Shor's Paper:** RSA를 깨는 방법을 주요 예제로 서술 함과 동시에, 타원 곡선 상에서의 이산 대수 문제로도 Shor 알고리즘이 확장 될 수 있음을 보임

- Shor on ECC ??
  - 타원 곡선에서의 **스칼라 곱셈에 대한 최적화 양자 회로** 구현이 중요
    → Shor 알고리즘에서 이산 대수 문제를 해결할 때 **가장 많은 비용이 드는 산술**

Shor on ECDLP

- Curve 상에서의 Scalar Multiplication? → **Double and ADD** Scalar Multiplication
  - Point addition의 연속
    - **Ex) 5P**
      1. $P + P = 2P$
      2. $2P + 2P = 4P$
      3. $4P + 1P = 5P$



Shor on ECDLP

- **Point Addition:** 다양한 연산들의 조합 → **양자 회로 상에서 최적화** (중요)

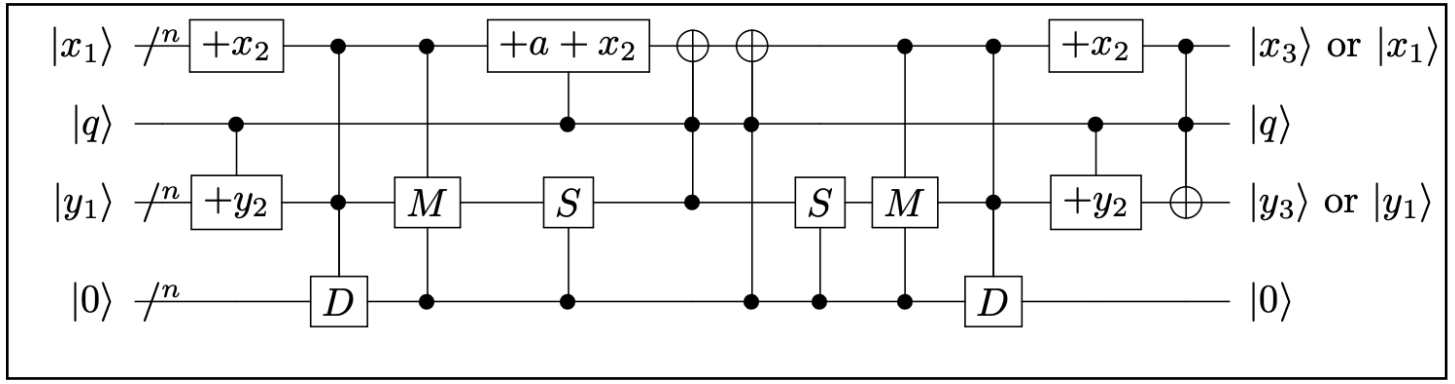
Two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2) \neq \pm P_1$ are added to produce $P_1 + P_2 = P_3 = (x_3, y_3)$ as

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \quad y_3 = (x_2 + x_3)\lambda + x_3 + y_2 \text{ with } \lambda = (y_1 + y_2)/(x_1 + x_2).$$

and $P_1 \neq -P_1$ is doubled to produce $[2]P_1 = (x_3, y_3)$ as

$$x_3 = \lambda^2 + \lambda + a, \quad y_3 = x_1^2 + (\lambda + 1)x_3 \text{ with } \lambda = x_1 + y_1/x_1.$$

to Quantum



S: Squaring
M: Multiplication
D: Division (Inversion)

Shor on ECC: Related Work

- 타원 곡선 상에서의 이산 대수 문제를 해결하기 위한 양자 자원들을 추정 (NIST curves 대상)
  - **ASIACRYPT (2017)** : "Quantum resource estimates for computing elliptic curve discrete logarithms"
    - 타원곡선에서의 이산대수문제를 해결하는데 필요한 양자 자원들을 추정함으로써 **RSA보다 ECC가 더 양자컴퓨터의 공격에 취약함**을 보임 (Prime ECC)
  - **PQCrypto (2020)** : "Improved quantum circuits for elliptic curve discrete logarithms"
    - ASIACRYPT(2017)의 결과보다 **큐비트 수, Depth 모두 줄임** (Prime ECC)
  - **CHES (2020)** : "Concrete quantum cryptanalysis of binary elliptic curves"
    - **Binary ECC에 대한 Shor 알고리즘 적용 시**, Prime ECC보다 **더 적은 자원으로 공격 가능함을 보임**
    - **Karatsuba 곱셈을 활용한 곱셈 및 Inversion 최적화 (양자 컴퓨터 상에서)**
  - **IEEE ACCESS (2023)** : "Depth-optimization of Quantum Cryptanalysis on Binary Elliptic Curves"
    - **Karatsuba 곱셈 개선 및 다른 Inversion 연산 사용 → 큐비트 수 보다는 Depth 최적화 중심** (Binary ECC)
  - **RSA Conference (2023)** : "Concrete quantum cryptanalysis of binary elliptic curves via Addition Chain"
    - **Arbitrary addition chain 적용, 두가지 버전의 Inversion 제공** (Binary ECC)

CHES Paper

Concrete quantum cryptanalysis of binary elliptic curves

Gustavo Banegas[1], Daniel J. Bernstein[2,3], Iggy van Hoof[4] and Tanja Lange[4]

[1] Chalmers University of Technology, Gothenburg, Sweden gustavo@cryptme.in
[2] University of Illinois at Chicago, Chicago, USA djb@cr.yp.to
[3] Ruhr University Bochum, Bochum, Germany
[4] Eindhoven University of Technology, Eindhoven, The Netherlands
i.v.hoof@tue.nl,tanja@hyperelliptic.org

Abstract. This paper analyzes and optimizes quantum circuits for computing discrete logarithms on binary elliptic curves, including reversible circuits for fixed-base-point scalar multiplication and the full stack of relevant subroutines. The main optimization target is the size of the quantum computer, i.e., the number of logical qubits required, as this appears to be the main obstacle to implementing Shor's polynomial-time discrete-logarithm algorithm. The secondary optimization target is the number of logical Toffoli gates.

For an elliptic curve over a field of 2^n elements, this paper reduces the number of qubits to $7n + \lfloor \log_2(n) \rfloor + 9$. At the same time this paper reduces the number of Toffoli gates to $48n^3 + 8n^{\log_2(3)+1} + 352n^2 \log_2(n) + 512n^2 + O(n^{\log_2(3)})$ with double-and-add scalar multiplication, and a logarithmic factor smaller with fixed-window scalar multiplication. The number of CNOT gates is also $O(n^3)$. Exact gate counts are given for various sizes of elliptic curves currently used for cryptography.

Keywords: Quantum cryptanalysis · elliptic curves · quantum resource estimation · quantum gates · Shor's algorithm

Shor on ECDLP (CHES)

- **Point Addition?**
  - 해당 논문에서는 Point Addition을 위한 **다양한 Binary Filed Arithmetic의 양자 구현**에 대해 설명하고 있음

5 Basic arithmetic

In this section we discuss reversible in-place algorithms for the basic arithmetic of binary polynomials modulo a field polynomial $m(z)$, i.e. elements of $\mathbb{F}_{2^n}$.

5.1 Addition and binary shift

5.2 Multiplication

For multiplication we use a space-efficient Karatsuba algorithm by Van Hoof [Hoo20] which uses $O(n^2)$ CNOT gates, $O(n^{\log_2 3})$ Toffoli gates and $3n$ total qubits: $2n$ qubits for the input, f, g, and n separate qubits for the output, h. The algorithm is detailed in Appendix A.

공저자 Van Hoof의 양자 카라추바 곱셈을 사용

5.3 Squaring

6 Inversion and division in binary finite fields

6.1 Inversion using extended GCD

6.2 Inversion using FLT

Shor on ECDLP (CHES)

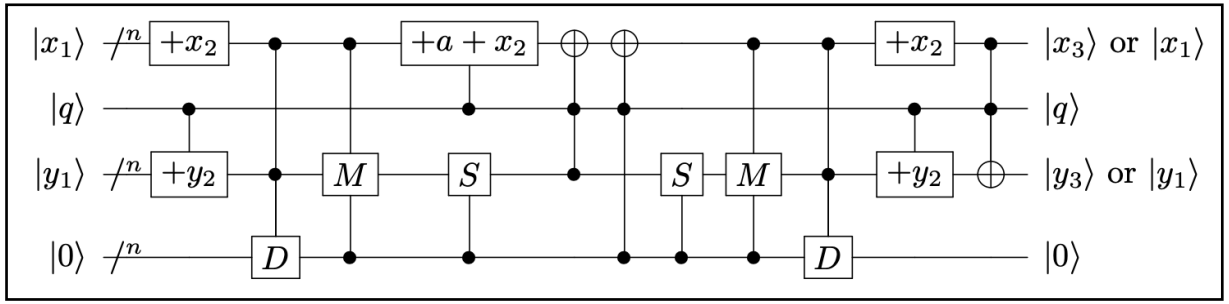
- Inversion 구현: 해당 논문에서는 **GCD** $(a \cdot s + b \cdot t = \gcd(a,\ b))$, **FLT** $(f^{2^n-2} = f^{-1} \bmod p(x))$ 기반의 Inversion을 구현 및 비교 (GCD 사용)
  - **GCD : 적은 큐빗, 많은 Toffoli gates**
  - **FLT : 많은 큐빗, 적은 Toffoli gates**

n	GCD				FLT			
	TOF	CNOT	qubits	depth	TOF	CNOT	qubits	depth
8	3,641	1,516	67	4113	243	2,212	56	1314
16	10,403	5,072	124	12,145	1,053	10,814	144	5968
127	277,195	227,902	903	378,843	50,255	502,870	1,778	203,500
163	442,161	375,738	1,156	612,331	83,353	906,170	1,956	451,408
233	827,977	743,136	1,646	1,172,733	132,783	1,486,464	3,029	640,266
283	1,202,987	1,088,400	1,997	1,708,863	236,279	2,708,404	3,962	1,434,686
571	4,461,673	4,266,438	4,014	6,494,306	814,617	10,941,536	9,136	6,151,999

< GCD, FLT 기반 Inversion 연산 비교 >

Shor on ECDLP (CHES)

- 아래 **Binary Field 연산들을 조합**하여 **Point addition 양자 회로 구현**
  - **Squaring (S)→ LUP 분해** 사용 (in-place)
  - **Multiplication (M) → 양자 카라추바 곱셈** 사용 (Van Hoof의 기법은 **전반적으로 성능이 좋음**)
  - **Inversion (D) →** 적은 큐비트를 사용하는 **GCD 버전 사용**
- **최적화가 달성되는 곳은, 양자 구현 비용이 높은 Binary 곱셈 및 역원**



< Point addition 양자 회로 >

	n	#Qubit	#Toffoli	#CNOT	Depth
Point addition	233	1,647	1,669,299	1,615,287	2,406,230
	283	1,998	2,428,369	2,359,187	3,503,964
	571	**4,015**	**8,987,401**	**9,081,061**	**13,238,554**

< Point addition 구현 결과 >

Shor on ECDLP (IEEE ACCESS)

- **Putranto et al., IEEE ACCESS (2023):**
  - **CHES(2020)에서 사용된 양자 곱셈 개선**
    - **LUP를 사용하는 상수 곱셈 reverse 대신 logical swap 사용 → Depth와 CNOT gate 수 감소**
    - **전체적으로 단계별 계산 순서를 변경**
  - **Fermat 의 소정리 기반의 Inversion선택**
    - $f^{2^n-2} = f^{-1} \bmod p(x)$ → 큐비트를 더 사용하지만 **Depth 및 양자 게이트 감소**
  - **CHES는?, 유클리드 알고리즘을 사용한 역 연산 (GCD)**
    - $a \cdot s + b \cdot t = \gcd(a,\ b)$ → **큐비트를 적게 사용**하며 Depth 및 양자 게이트 증가

n	Ours qubits	BAN qubits	Ours (1 PA) Toffoli	Ours (1 PA) CNOT	Ours (1 PA) depth	BAN (1 PA) Toffoli	BAN (1 PA) CNOT	BAN (1 PA) depth	Ours ($2n+2$ PA) Toffoli	BAN ($2n+2$ PA) Toffoli
8	74	68	348	2,734	210	7,360	3,522	8,562	6,264	132,480
16	194	125	1,344	15,924	623	21,016	11,686	25,205	45,696	714,544
127	2,320	904	52,773	1,352,497	7,010	559,141	497,957	776,234	13,509,888	143,140,096
163	2,805	1,157	96,299	2,137,063	14,632	893,585	827,623	1,262,280	31,586,072	293,095,880
233	4,228	1,647	152,067	4,480,745	46,702	1,669,299	1,615,287	2,406,230	71,167,356	781,231,932
283	5,694	1,998	267,115	7,376,571	34,792	2,427,369	2,359,187	3,503,964	151,721,320	1,378,745,592
571	13,167	4,015	935,883	32,888,178	93,142	8,987,401	9,081,061	13,238,554	1,070,650,152	10,281,586,744

Shor on ECDLP (IEEE ACCESS)

- **Binary 곱셈 개선**
  - **LUP를 사용하는 상수 곱셈 역 연산 사용 X → Depth와 CNOT gate 수 감소**

    → 대신 비용이 들지 않는 logical swap
  - **전체적으로 단계별 계산 순서를 변경**

Degree	schoolbook	This Work			Previous Work [17]		
	TOF	TOF	CNOT	Depth	TOF	CNOT	Depth
2	4	3	10	3	3	9	9
4	16	9	32	9	9	44	32
8	64	27	102	82	27	220	124
16	256	81	376	226	81	678	365
32	1,024	243	1,194	246	243	2,238	1,110
64	4,096	729	3,973	754	729	6,896	3,129
127	16,129	2,183	12,659	3,151	2,185	20,632	8,769
128	16,384	2,187	13,223	3,311	2,187	21,272	9,142
163	26,569	4,355	22,080	6,976	4,387	37,168	17,906
233	54,289	6,307	36,238	25,225	6,323	63,655	29,530
256	65,536	6,561	36,005	17,080	6,561	64,706	26,725
283	80,089	10,241	54,099	23,225	10,273	89,620	41,548
571	326,041	31,139	166,982	45,979	31,171	270,940	121,821
1024	1,048,576	59,049	487,345	66,470	59,049	591,942	234,053

Shor on ECDLP (IEEE ACCESS)

- **FLT 기반 inversion**
  - **CHES – 각 반복마다 제곱연산에 역 연산을 적용하여 연산 해제 → 큐비트 수 아낌**
  - **IEEE ACCESS**
    - **CHES 논문의 FLT 기반 inversion 내의 계산 순서를 변경**
    - **제곱연산 역 연산 사용 X**

      → 더 많은 큐비트를 사용하지만 depth와 gate 수 감소

n	Toffoli Count			CNOT Count			Qubit Count			Overall Depth		
	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*
8	108	256	108	1012	268	1064	57	89	41	13	13	12
16	486	1536	486	6534	750	6792	161	257	113	24	24	23
127	24013	177419	24013	663541	8920	668272	2287	3684	1525	143	143	142
163	39195	239121	39195	946681	48499	984652	2772	4339	1631	178	178	270
233	63070	542890	63070	2010283	47303	2034266	4195	6525	2564	249	249	270
283	112651	880979	112651	3343751	143444	3473260	5661	8774	3397	301	301	528
571	404807	4238533	404807	15138845	341985	15445474	9408	20557	7995	592	592	1042

Shor on ECDLP (IEEE ACCESS)

- **FLT 기반 inversion**
  - **LAR* : schoolbook 곱셈 사용한 동일한 방식의 FLT inversion**
  - **BAN* : 개선한 곱셈을 적용한 CHES 기반 FLT inversion**

  - **LAR*에 비해 Toffoli, Qubit 측면에서 최적화**

    → 개선한 곱셈을 사용하여 depth는 유지하고 낮은 qubit을 사용할 수 있음
  - **BAN*에 비해 depth, CNOT gate 수 측면에서 최적화**

    → 개선한 FLT inversion을 통해 낮은 depth를 달성할 수 있음

n	Toffoli Count			CNOT Count			Qubit Count			Overall Depth		
	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*
8	108	256	108	1012	268	1064	57	89	41	13	13	12
16	486	1536	486	6534	750	6792	161	257	113	24	24	23
127	24013	177419	24013	663541	8920	668272	2287	3684	1525	143	143	142
163	39195	239121	39195	946681	48499	984652	2772	4339	1631	178	178	270
233	63070	542890	63070	2010283	47303	2034266	4195	6525	2564	249	249	270
283	112651	880979	112651	3343751	143444	3473260	5661	8774	3397	301	301	528
571	404807	4238533	404807	15138845	341985	15445474	9408	20557	7995	592	592	1042

Shor on ECDLP (RSA Conference)

- **Taguchi.et al., RSA Conference (2023) :**
  - **CHES(2020)의 point addition** 사용
  - **Squaring으로 LUP 사용** (in-place 구현)
  - **Kim et al. [KKKH22] 곱셈기** 사용
    - Karatsuba-like formula 기반 CRT 곱셈 기법 사용
      → Van Hoof 의 곱셈기보다 **Toffoli gate** 및 **depth** 를 개선

  - Itoh-Tsujii addition chain이 아닌 **arbitrary addition chain 사용**
    - **지수를 작은 수의 합으로 나타내고, 이러한 작은 수들의 거듭제곱을 계산하는 방식**
    - **큐비트** 수를 아끼기 위해 **거듭제곱을 적게** 사용하는 임의의 **addition chain** 사용
      ex) $\{1, 2, 4, 8, 16, 32, 64, 128, 160, 162\} \rightarrow \{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$

— 거듭제곱
— 덧셈

  - FLT 기반의 두 가지 Inversion 구현
    - **Basic algorithm (Depth 중점)**
      → IEEE Access (2023)와 동일
    - **Extended algorithm (Qubit 중점)**
      → Inversion 내부 연산 중, **리버스 연산**을 활용하여 **ancilla 큐비트 초기화**

Shor on ECDLP (RSA Conference)

- **addition chain?**

  - **Fermat의 소정리 기반의 역 연산**
    - $f^{2^n-2} = f^{-1} \bmod p(x)$
    - $f^{2^n-2} = \left(f^{2^{n-1}-1}\right)^2$

$$\langle 2^{2^{k-1}} - 1\rangle \times \langle 2^{2^{k-1}} - 1\rangle^{2^{2^{k-1}}} = \langle 2^{2^k} - 1\rangle,$$
$$\langle 2^{\alpha} - 1\rangle^{2^{\beta}} \times \langle 2^{\beta} - 1\rangle = \langle 2^{\alpha+\beta} - 1\rangle,$$
$$\langle 2^{n-1} - 1\rangle^2 = \langle 2^n - 2\rangle.$$

    - **N=163**
    - $f^{2^{n-1}-1} = f^{2^{162}-1}$
    - $162 = 128 + 32 + 2 = 2^7 + 2^5 + 2^1$
    - $f^{2^{2^1}-1}, f^{2^{2^2}-1}, \ldots, f^{2^{2^7}-1}, f^{2^{2^7+2^5+}-1}, f^{2^{2^7+2^5+2^1}-1} = f^{2^{162}-1}$

    - **162에 대한 Addition chain = $\{1, 2^1, 2^2, \ldots . 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}$**

Shor on ECDLP (RSA Conference)

- **addition chain?**
  - **Fermat의 소정리 기반의 역 연산**
    - **162 에 대한 Addition chain = $\{1, 2^1, 2^2, \ldots . 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}$**
      - **Length =9, double terms 7 , add terms 2**
    - **162 에 대한 다른 Addition chain = $\{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$ =$\{1, 2^1, 2^2, 2^3, 2^4, 2^5/2^5 + 1, 2^5 + 2^5 + 1\ , 2^5 + 2^5 + 2^5 + 1, 2^5 + 2^5 + 1 + 2^5 + 2^5 + 2^5 + 1 = 162\}$**
      - **Length =9, double terms 5 , add terms 4**

Table 1: d, m, ℓ of Itoh and Tsujii's addition chains

n	163	233	283	571
d	7	7	8	9
m	2	3	3	4
ℓ	9	10	11	13

Table 2: d, m, ℓ of our choice of addition chains

n	163	233	283	571
d	5	4	3	4
m	4	6	8	8
ℓ	9	10	11	12

Shor on ECDLP (RSA Conference)

- **Shor algorithm 자원 추정 비교**
  - **Basic algorithm**
    - 가장 낮은 **depth** 달성
    - **PWLK22-FLT(IEEE ACCESS)**보다 **큐비트** 수 감소
  - **Extended algorithm**
    - Basic algorithm보다 낮은 **큐비트** 수
    - **BBHL21-GCD(CHES)**보다 큐비트 수는 높지만 매우 낮은 **depth** 달성

	n	#Qubit	#Toffoli	#CNOT	Depth
Bagnes et al. (2021)	283	1,998	1,359,458,584	1,644,682,056	1,672,269,248
	571	**4,015**	**10,156,396,536**	**13,091,280,488**	**12,963,368,704**
Putranto et al. (2023)	283	6,227	49,121,208	6,494,863,592	977,046,336
	571	**14,276**	**246,235,704**	**59,611,633,224**	**8,283,571,296**
Taguchi et al.(2023) / Basic	283	4,812	49,121,208	6,491,648,712	977,034,976
	571	**10,850**	**228,787,416**	**55,651,292,840**	**8,000,884,320**
Taguchi et al. (2023) / extended	283	3,963	49,121,208	6,506,182,696	981,029,152
	571	**8,566**	**228,787,416**	**55,778,093,800**	**8,053,979,648**

Q & A