Toffoli gate count Optimized Space-Efficient Quantum Circuit for Binary Field Multiplication

Sunyeop Kim[1] Insung Kim[1] Seonggyeom Kim[1] and Seokhie Hong[1]

Institute of Cyber Security & Privacy (ICSP), Korea University, South Korea
kin3548@gmail.com, cmcom35@icloud.com, jeffgyeom@korea.ac.kr, shhong@korea.ac.kr

Abstract. Shor's algorithm solves Elliptic Curve Discrete Logarithm Problem(ECDLP) in polynomial time. To optimize Shor's algorithm for binary elliptic curve, reducing the cost for binary field multiplication is essential because it is most cost-critical basic arithmetic. In this paper, we propose Toffoli gate count optimized space-efficient quantum circuits for binary field ($\mathbb{F}_{2^n}$) multiplication. To do so, we take advantage of Karatsuba-like formula and show that its application can be provided without ancillary qubits and optimized them in terms of CNOT gate and depth. Based on the Karatsuba-like formula, we drive a space-efficient CRT-based multiplication with two types of out-of-place multiplication algorithm to reduce CNOT gate cost. Our quantum circuits do not use ancillary qubits and have extremely low TOF gates count $O(n2^{\log_2^* n})$ where $\log_2^*$ is a function named iterative logarithm that grows very slowly. Compared to recent Karatsuba-based space-efficient quantum circuit, our circuit requires only ($12 \sim 24\%$) of Toffoli gate count with comparable depth for cryptographic field sizes ($n = 233 \sim 571$). To the best of our knowledge, this is the first result that utilizes Karatsuba-like formula and CRT-based multiplication in quantum circuits.

Keywords: Quantum Computers, Karatsuba Multiplication, CRT, Binary Field, Toffoli gate

1 Introduction

Quantum algorithms have received a lot of attention from the cryptographic community due to their impact on the security reduction of cryptosystems. Shor's algorithm solves Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial time, and Grover's algorithm gives quadratic speed up in searching problems. However quantum computers at present have only limited number of qubits and further development must be needed to attack currently used cryptographic algorithms. To estimate how much quantum resources will be needed to attack the cryptographic algorithms, it is necessary to implement them in quantum circuits. Therefore implementing cryptographic algorithms in quantum circuits and estimating, optimizing the costs of those circuits have become a widespread field of research. In this paper, we focus on binary field multiplications that are commonly used as basic arithmetic in many cryptographic algorithms, especially ECDSA.

The primary ways to express binary field elements are normal basis representation and polynomial basis representation. The addition is the same in the two representations, whereas the multiplication should be implemented differently. The multiplication with normal basis can be implemented by a quantum circuit that has $O(n)$ depth and requires $O(n^2)$ Toffoli (TOF) gates without CNOT gate [2]. The quantum circuits for the polynomial basis are based on either Mastrovito multiplier or Karatsuba multiplication. The Mastrovito-based quantum circuit of [15] uses $O(n^2)$ TOF and CNOT gates, respectively. Regarding Karatsuba multiplication, the quantum circuit of [13] needs $O(n^{\log_2 3})$ qubits with the same gate complexity as that of classical Karatsuba multiplication.

For the space-efficient Karatsuba multiplication ($3n$ qubits without ancilla), ModMult of [11], which is based on the multiplication of polynomial over GF(2)(so called carry-less product) algorithm KMult, needs $O(n^{\log_2 3})$ TOF gates and $O(n^2)$ CNOT gates. ModMult is also used in solving ECDLP, i.e., the analysis of ECDSA [4] due to it's space efficiency and small number of TOF gates. Thereafter, the improved version of ModMult, ModMult_Imp, with a reduced number of CNOT gates was proposed in [19]. For depth optimized Karatsuba multiplication circuit, trade-off between number of qubits and Toffoli depth were proposed [12]. The number of Toffoli gate and qubits needed for the circuit are both $O(n^{\log_2 3})$ with toffoli depth 1.

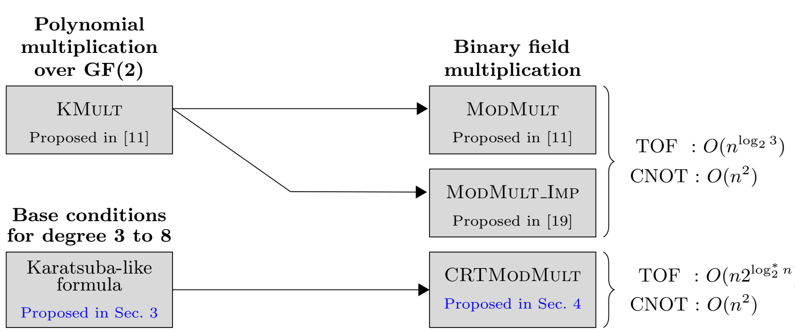


Fig. 1. Quantum Space-Efficient Karatsuba Multiplication

1.1 Our Contribution

1. **Karatsuba like formula in quantum circuit without ancilla.** We show that Karatsuba-like formula, which is a generalization of Karatsuba multiplication splitting polynomials into more than two terms [1, 17, 10, 7], can be

implemented in quantum circuit without ancilla. We also give optimization method in terms of CNOT gate count and depth, along with cost for 3-8 split Karatsuba-like formula.

2. **Toffoli-count optimized space-efficient quantum circuit for binary field multiplication.** Based on the Karatsuba-like formula, we propose TOF-count optimized space-efficient quantum circuit for binary field multiplication CRTModMult. CRTModMult provides TOF gate-optimized multiplication circuit by utilizing the CRT-based multiplication of [9]. Also we used two types of out-of-place multiplication algorithm to reduce CNOT gate cost. This approaches extremely reduces the required TOF gates into $O(n2^{\log_2^* n})$ from $O(n^{\log_2 3})$ with the asymptotically same number of CNOT gates $O(n^2)$ compared to ModMult_Imp. Fig. 1 demonstrates the comparison between the approaches of [4, 19] and ours. To the best of our knowledge, this is the first result that utilizes Karatsuba-like formula and CRT-based multiplication in quantum circuits.
3. **Comparing with previous space-efficient circuit.** We implemented CRTModMult in quantum programming tool Qiskit and obtained the resource analysis. The required TOF gates is reduced to 12 $\sim$ 24% of ModMult_Imp for cryptographic field sizes ($n = 233 \sim 571$). Table 1 gives the comparison of asymptotic costs. Our circuit also has a lower depth than ModMult_Imp, despite 3 $\sim$ 4 times many CNOT gates. Since our focus is optimizing the number of TOF gate without ancilla, which is opposed to [12] we will not compare our circuit with [12]. Considering the high cost of TOF gate, which consists of 7 T gates and 8 Clifford gates [3], CRTModMult is promising to be used in quantum computing for binary elliptic curves.

1.2 Organization

Section 2 introduces the relevant contents, basic quantum circuits and previous results of quantum Karatsuba multiplication. In Section 3, we introduce Karatsuba-like formula in quantum circuit. Our proposed modular multiplication CRTModMult is given in Section 4. The implement results of CRTModMult in qiskit are demonstrated in Section 6. We conclude the paper in Section 7.

Table 1. Comparison of Asymptotic Costs with Previous Works

n	TOF				CNOT				Qubit Count			
	Ours	[19]	[11]	[13]	Ours	[19]	[11]	[13]	Ours	[19]	[11]	[13]
16	64	81	81	81	974	655	725	376	48	48	48	113
127	737	2185	2185	2185	49040	20300	21028	13046	381	381	381	2433
233	1441	6307	6307	6307	154892	60453	63655	-	699	699	699	-
256	1590	6561	6561	6561	184948	63689	66107	57008	768	768	768	7043
283	1784	10241	10241	10241	224246	87929	89620	-	849	849	849	-
571	3813	31139	31139	31139	862604	267771	270940	-	1713	1713	1713	-
	$O(n2^{\log_2^* n})$	$O(n^{\log_2 3})$			$O(n^2)$			$O(n^{\log_2 3})$	$3n$			$O(n^{\log_2 3})$

2 Preliminary

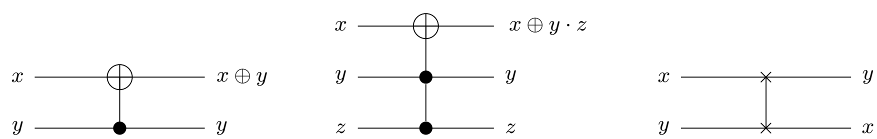
This chapter defines the basic quantum circuits and previous quantum karatsuba multiplication circuit ModMult.

2.1 Quantum Gate

We use CNOT, TOF, and SWAP gates throughout this paper.

$$\begin{aligned}\text{CNOT gate} &: (x, y) \to (x \oplus y, y)\\ \text{TOF gate} &: (x, y, z) \to (x \oplus y \cdot z, y, z)\\ \text{SWAP gate} &: (x, y) \to (y, x)\end{aligned}$$

In algorithms we write the above gates as $\text{CNOT}(x, y) \to x$, $\text{TOF}(x, y, z) \to x$, and $\text{SWAP}(x, y)$. Each gate can be illustrated as (a), (b), and (c) of Fig. 2, respectively. Also for quantum circuit C, We define $C^\dagger$ as the inverse of it.



(a) CNOT gate (b) TOF gate (c) SWAP gate

Fig. 2. Quantum Gates

2.2 Basic Setting

All polynomials considered in this paper are in $\mathbb{F}_2[x]$ and $\deg(f)$ denote the degree of polynomial $f(x)$. A polynomial $f(x) = f_0 + f_1x + \cdots + f_{n-1}x^{n-1}$ of degree at most $n-1$ is called an n-term polynomial and we represent it as an n-qubit array $(f_0, f_1, \cdots, f_{n-1})$. $M(n)$ denotes the number of TOF (or AND) gates required to multiply two arbitrary n-term polynomials. k $(0 < k \leq n)$ coefficients of $x^{2n-1+k}, \ldots, x^{2n-2}$ of the multiplication $f(x)g(x)$ are called remainder coefficients and $\lambda(k)$ denotes the number of TOF (or AND) gates required to calculate them. Minimum value of $M(n)$ is open problem for $n > 8$. Binary field $\mathbb{F}_{2^n}$ can be identified with quotient ring $\mathbb{F}_2[x]/\langle p(x)\rangle$ which is called polynomial basis representation, where $p(x)$ is an irreducible polynomial of degree n.

2.3 Related Works on Space-efficient Karatsuba Multiplication

For given input polynomials $f(x), g(x)$ of size n and $h(x)$ size of $2n$, output of Karatsuba algorithm is $h + fg$. For $k = \lceil \frac{n}{2} \rceil$, Karatsuba multiplication splits each polynomial as follows : $f(x) = f_0 + f_1x^k$, $g(x) = g_0 + g_1x^k$ and $h(x) = h_0 + h_1x^k + h_2x^{2k} + h_3x^3$.

Letting $\alpha = f_0g_0$, $\beta = f_1g_1$, $\gamma = (f_0 + f_1)(g_0 + g_1)$, $h + fg$ can be rewrited as follows.

$$\begin{aligned} h + fg &= h + \alpha + (\alpha + \beta + \gamma)x^k + \beta x^{2k} \\ &= h + (1 + x^k)\alpha + x^k\gamma + x^k(1 + x^k)\beta \end{aligned}$$

Using the fact that α, β, γ are results of k or $n - k$ term polynomial multiplications and multiplication of constant polynomial can be done inplace, recursive polynomial multiplication algorithm 'KMULT' was given in [11]. KMULT algorithm can be extend to binary field multiplication algorithm 'MODMULT' along with constant polynomial multiplication on fixed modulus. MODMULT has $O(n^{\log_2 3})$ TOF gate complexity and $O(n^2)$ CNOT gate cost, which is far efficient than school book multiplication. Afterwards, MODMULT_IMP, the improved version of MODMULT which changed the order of α, β, γ in computation step, was given recently [19]. Compared to MODMULT, MODMULT_IMP has same number of TOF gate but CNOT gate cost and depth were reduced. Therefore MODMULT_IMP will be our main target but the number of gates in the paper appears to be underestimated. So we will separately implement MODMULT_IMP in Qiskit in that comparisons can be made under the same conditions.

3 Karatsuba-like formula

Karatsuba multiplication can be generalized by splitting polynomials more than two terms. This is called Karatsuba-like formula and it has been first studied for [1] and have been improved over the years [17, 10, 7]. Because of high searching complexity, only up to 8-split Karatsuba-like formula were studied. In this paper, we focus on Karatsuba-like formula which can be represented in symmetric bilinear form [10]. Symmetric bilinear form consists of

1. a top layer consisting only of XOR gates with n bit inputs A, B
2. symmetric multiplication (AND) layer that computes following form
$$\sum_{i \in S} a_i \cdot \sum_{i \in S} b_i$$
where
$$A = [a_0, ..., a_{n-1}],\ B = [b_0, ..., b_{n-1}],\ S \subseteq \{0, ..., n-1\}$$
3. a bottom layer that uses only XOR gates with l bit output C

Symmetric bilinear form can be represented with $k \times n$ matrix T which corresponds to top layer and $k \times l$ matrix R which corresponds to bottom layer.

$$C = R \cdot [(T \cdot A) \circ (T \cdot B)]$$

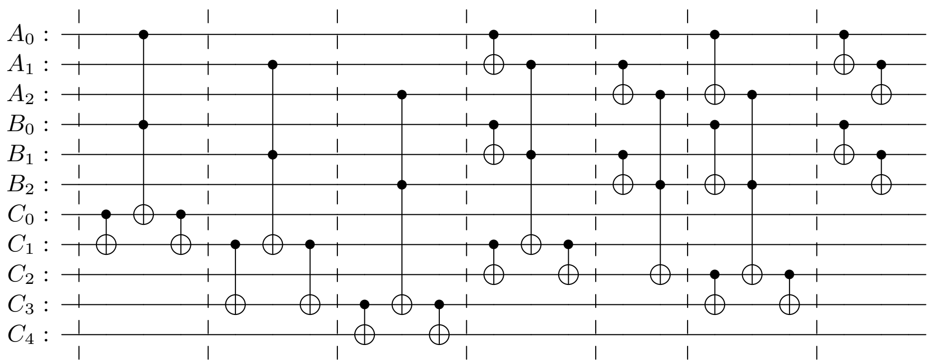


Fig. 3. 3-split Karatsuba-Like Formula in Quantum Circuit

For example, karatsuba multiplication is symmetric bilinear form with following T, R matrix.

$$T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, R = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Each row of T matrix can be converted to CNOT gates in A and B registers. For i th column of R matrix, '1' can be seen as CNOT operation controlled target bit of i th TOF gate. So we can construct the symmetric bilinear form in quantum circuit without ancilla. we define $SBF_{T,R}(a, b, c)$ as quantum circuit which symmetric bilinear formula with T, R matrix. Note that symmetric bilinear form of n-split Karatsuba-like formula has parameter $l = (2n-1)$ and k corresponds the number of AND (or TOF) gate needed for polynomial multiplication. Fig 3 is an example of 3-split Karatsuba-like formula in quantum circuit with following T, R matrix.

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

3.1 Optimizing Karatsuba-Like Formula in Quantum Circuit

Optimizing Cost of *T* Matrix. Optimizing T matrix on quantum circuit can be seen as problem similar to straight-line program but it is different in that only intermediate values stored in qubit at the moment can be reused in this case. Since this is a difficult problem likewise, we heuristically reduced the number of CNOTs needed based of greedy algorithm.

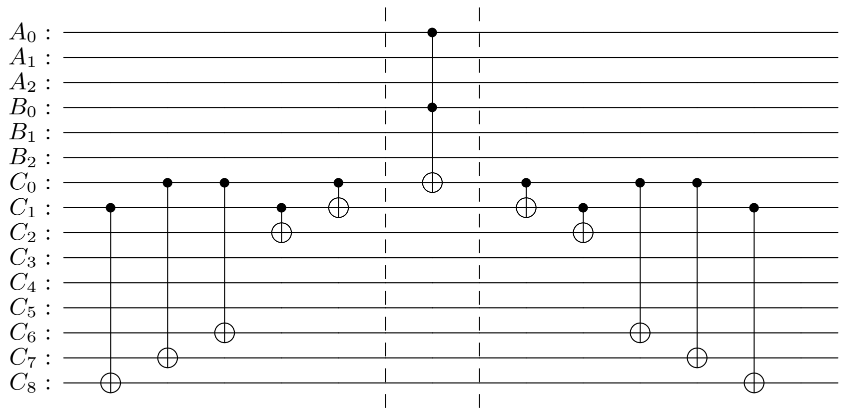


Fig. 4. Example of Reducing the Depth of R Matrix

Optimizing Depth of *R* Matrix. With respect to R matrix, we focus on reducing the depth and explain it by example. Assume that some column of R matrix is [1,1,1,0,0,0,1,1,1]. If C_0 is only used as the control qubit in Fig 4, depth will be 11, considering † operation in front of the TOF gate. But if C_1, where value of C_0 is already copied, is also used as a control bit, depth is reduced to 7. We reduced depth of every Karatsuba-like formula likewise.

3.2 Cost of (3 $\sim$ 8)-split Karatsuba-like formula with optimization

We examined $3 \sim 8$ Karatsuba-like formula in [17, 10, 7] and selected ones which has lowest CNOT gate cost when above optimizations are applied. Table 2 gives complexity of 3 - 8 degree polynomial multiplication circiut based on Karatsuba like formula and KMult.

Table 2. number of TOF, CNOT gate and depth for (3 $\sim$ 8)-split Karatsuba-like formula

Term	Karatsuba-like formula			KMult		
	TOF	CNOT	depth	TOF	CNOT	depth
3	6	20	13	7	28	21
4	13	48	33	17	88	52
5	13	80	53	17	88	52
6	17	124	77	21	116	63
7	22	180	95	25	152	77
8	26	310	157	27	176	84

4 CRT-Based Modular Multiplication

CRT-based modular multiplication uses the CRT formula (Theorem 1) to make high-order multiplication circuit form low-degree circuit[9]. Because of huge number of XOR gates, CRT-based multiplication does not get much attention in classical circuit. However it would be a good choice in quantum circuits where cost of TOF gate is much higher than CNOT gate.

Theorem 1. *Let $m_1(x), ..., m_t(x)$ be pairwise co-prime polynomials and define $m(x) = \prod_{n=1}^{t} m_i(x)$, $h_i(x) = (\frac{m(x)}{m_i(x)})(\frac{m(x)}{m_i(x)})^{-1} \bmod m_i(x))$. Then following equation holds for every polynomial $r(x)$ which satisfies $deg(r(x)) < deg(m(x))$.*

$$r(x) = \sum_{i=1}^{t} r_i(x)h_i(x) \bmod m(x),$$

where $r_i(x) = r(x) \bmod m_i(x)$

We will follow the method in [9], which separately calculate remainder coefficients

$$c_{2n-1-w}, c_{2n-2-w}, \ldots, c_{2n-1}$$

and merge with CRT result.(known as modulo $(x - \infty)^w$ construction). We also add modular $p(x)$ reduction, which is linear operation that does not change TOF gate complexity, to original algorithm. Based on parameters $m_1(x), ..., m_t(x)$, $m(x) = \prod_{n=1}^{t} m_i(x)$ and $w = 2n - 1 - deg(m)$, CRT-based modular multiplication is performed by following steps. Since TOF(or AND) gates are used only in the modular multiplication in step 2 and 4, $M(n) \le \sum_{i=1}^{t} M(deg(m_i)) + \lambda(w)$ holds.

1. **modular reduction**

   $$f_i(x) := f(x) \bmod m_i(x), g_i(x) := g(x) \bmod m_i(x)$$

   for $1 \le i \le t$
2. **modular multiplication**

   $$C_i(x) = f_i(x)g_i(x) \bmod m_i(x)$$

   for $1 \le i \le t$
3. **CRT**

   $$C'(x) = \sum_{i=1}^{t} (C_i(x)h_i(x) \bmod m(x)) \bmod p(x)$$
4. **modulo** $(x - \infty)^w$
   compute remainder coefficients

   $$c_{2n-2}, ..., c_{2n-1-w}$$

   and

   $$C(x) = C'(x) + \sum_{i=2n-1-w}^{2n-2} c_i((x^i) + (x^i \bmod m(x))) \bmod p(x)$$

4.1 Choice of m_i's for CRT-Based Multiplication

To reduce TOF(or AND) gate as much as possible, it is typical to choose $m_1(x), ..., m_t(x)$ as power of irreducible polynomials. Theorem 2 gives the number of irreducible polynomials.

Theorem 2. *Let $I(n)$ be number of irreducible polynomials of degree n over $\mathbb{F}_2$. Then*

$$I(n) \overset{(I)}{=} \frac{1}{n} \sum_{d|n} \mu(\frac{n}{d}) 2^d \overset{(II)}{=} \frac{2^n - O(2^{n/2})}{n} \approx \frac{2^n}{n}$$

where μ is Möbius function.

Proof. *Equality (I) is given in [18], [16]. So we will only prove equality (II).*

By Möbius inversion formula, $2^n = \sum_{d|n} d \cdot I(d)$. Therefore $I(n) \leq 2^n/n$ Also,

$$\begin{aligned} 2^n &= \sum_{d|n} d \cdot I(d) \\ &\leq nI(n) + 2^{n/2} + \sum_{d|n, d<n/2} d \cdot 2^d \\ &\leq nI(n) + 2^{n/2} + n \cdot 2^{n/3} \end{aligned}$$

Then,

$$\begin{aligned} I(n) &\geq \frac{(2^n - (1/2)2^{n/2} - n/3 \cdot 2^{n/3})}{n} \\ &= \frac{(2^n - O(2^{n/2}))}{n}. \end{aligned}$$

Based on theorem 2, it is enough to use powers of irreducible polynomials of degree up to n for multiplication of 2^n- term polynomials since $\sum_{i=1}^{n} k \times \frac{2^k}{k} \approx 2^{n+1}$. For cryptographic field size, it is sufficient to use irreducible polynomials up to degree 10 and $I(n)$ for $2 \leq n \leq 10$ are given in Table 3.

5 Quantum Circuit Implementation of CRT Modular Multiplication

In this chapter we introduce sub-algorithms, including computation of remainder coefficient and matrix multiplication, and propose our main circuit. CRT

Table 3. $I(n)$ for $2 \leq n \leq 10$

degree	1	2	3	4	5	6	7	8	9	10
$I(n)$	2	1	2	3	6	9	18	30	56	99

based multiplication includes constant polynomial multiplication and modular reductions with fixed modulus which can be seen as a linear operation (i.e., matrix multiplication). Therefore matrix multiplication circuits will be used as important sub-algorithms.

5.1 Computation of remainder Coefficients

In modulo $(x-\infty)^w$ step coefficients of $x^{2n-1+k}, ..., x^{2n-2}$ are calculated separately to reduce TOF gate cost. Let $f(x), g(x)$ be n-term polynomials and $h(x) := f(x)g(x)$ be $(2n-1)$-term polynomial whose i-th coefficient is f_i, g_i, and h_i , respectively. Define $s_i = f_i g_i$ and $s_{i,j} = (f_i + f_j)(g_i + g_j)$. Then it is straightforward to see that

$$h_{2n-2-t} = \sum_{i+j=2n-t-2, n>i>j} s_{i,j} + \sum_{i=2n-t-2}^{2n-2} s_i$$

Based on above formula algorithm 1 computes $h_{2n-2}, h_{2n-3}, ..., h_{2n-k-1}$ with cost of $k + (k^2/4)$ TOF gates and $3(k-1) + k^2$ CNOT gates.

Algorithm 1: HIGHDEG$_{k,n}$. computation of k high degree coefficients

Quantum input : Two binary n term polynomials f, g, t stored in arrays **F**,**G**, and **H**, respectively

Quantum output: **H** as $t(x) + h_{2n-k+1} + h_{2n-k+2}x^1 + ... + h_{2n-1}x^{k-1}$ where $h(x) := f(x)g(x) = h_0 + h_1x^1 + ... + h_{2n-1}x^{2n-1}$

for $i = n-1 \dots n-k$ **do**
    $\mathbf{H}[i+k-n] \leftarrow \mathrm{TOF}(\mathbf{H}[i], \mathbf{F}[i], \mathbf{G}[i])$
    $\mathbf{H}[i+k-n-1] \leftarrow \mathrm{CNOT}(\mathbf{H}[i+k-n-1], \mathbf{H}[i])$
for $i = n-1 \dots n - \lceil (k+1)/2 \rceil$ **do**
    for $j = i-1 \dots 2n-1-k-i$ **do**
        $\mathbf{F}[j] \leftarrow \mathrm{CNOT}(\mathbf{F}[j], \mathbf{F}[i])$
        $\mathbf{G}[j] \leftarrow \mathrm{CNOT}(\mathbf{G}[j], \mathbf{G}[i])$
        $\mathbf{H}[i+j-2n-2+k] \leftarrow \mathrm{TOF}(\mathbf{H}[i+j-2n-2+k], \mathbf{F}[j], \mathbf{G}[j])$
        $\mathbf{F}[j] \leftarrow \mathrm{CNOT}(\mathbf{F}[j], \mathbf{F}[i])$
        $\mathbf{G}[j] \leftarrow \mathrm{CNOT}(\mathbf{G}[j], \mathbf{G}[i])$

5.2 In-Place Multiplication.

PLU decomposition factors an invertible matrix M as a product of permutation matrix P, lower triangular matrix L and upper triangular matrix U; $M = PLU$. Such a decomposition allows in-place multiplication, which does not require any ancillary qubits. In the multiplication, P can be implemented with swap gates and U and L are implemented as a sequence of CNOT gates. '1' not on the diagonal in U and L is converted to a CNOT gate controlled by the column index on the row index. For U, the order of CNOT gates should be top row to bottom row whereas the order for L should be bottom row to top row. Algorithm 2 gives the in-place multiplication circuit using at most $n^2 - n$ CNOT gates.

Algorithm 2: InMult($\mathbf{G}$; $M = PLU$) : in-place multiplication of an invertible $n \times n$ binary matrix M

Fixed input : PLU-decomposition (P, L, U) of M
Quantum input : an input $1 \times n$ binary vector g stored in an array $\mathbf{G}$
Quantum output: $\mathbf{G}$ stores the output Mg

```
for i = 0 ... n − 1 do
    for j = i + 1 ... n − 1 do
        if U[i, j] = 1 then
            G[i] ← CNOT(G[i], G[j])
for i = n − 1 ... 0 do
    for j = i − 1 ... 0 do
        if L[i, j] = 1 then
            G[i] ← CNOT(G[i], G[j])
for i = 0 ... n do
    for j = i + 1 ... n − 1 do
        if P[i, j] = 1 then
            SWAP(G[i], G[j])
            Swap the (i, j)-th columns of P
```

5.3 Out-of-place multiplications

Let M be an arbitrary $n_2 \times n_1$ matrix and $|A\rangle, |B\rangle$ be n_1, n_2-qubit vectors respectively. We present two types of out-of-place matrix multiplication circuit that computes $|A\rangle |B\rangle \rightarrow |A\rangle |B + MA\rangle$. Considering the matrix size and the possibility of combining two matrices, each algorithms will be used for appropriate situation.

Naïve Approach. Algorithm 3 gives out-of-place multiplication OutMultNaïve. This algorithm simply uses $|A\rangle$ as control bit of CNOT gate and $|MA\rangle$ is XORed to $|B\rangle$. CNOT gates needed are equal to the number of '1', which means that $(n_1 \times n_2)/2$ CNOT gates will be used in average case. Note that two OutMultNaïve can be merged to one OutMultNaïve. More explicitly, following equality holds.

$$\begin{aligned}\textsc{OutMultNaïve}(\mathbf{A}, \mathbf{B}; M_1) + \textsc{OutMultNaïve}(\mathbf{A}, \mathbf{B}; M_2) \\ = \textsc{OutMultNaïve}(\mathbf{A}, \mathbf{B}; M_1 + M_2)\end{aligned}$$

Algorithm 3: $\textsc{OutMultNaïve}(\mathbf{A}, \mathbf{B}; M)$: out-of-place matrix multiplication algorithm type 1

Fixed input : $n_2 \times n_1$ binary matrix M
Quantum input : n_1, n_2-qubit vectors $|A\rangle, |B\rangle$ stored in $\mathbf{A}$ and $\mathbf{B}$, respectively
Quantum output: $\mathbf{B}$ stores the output $|A\rangle |B + MA\rangle$

for $i = 0 \dots n_1 - 1$ **do**
  for $j = 0 \dots n_2 - 1$ **do**
    if $M[j, i] = 1$ **then**
      $\mathbf{B}[j] \leftarrow \mathrm{CNOT}(\mathbf{B}[j], \mathbf{A}[i])$

Algorithm 4: $\textsc{OutMultReuse}(\mathbf{A}, \mathbf{B}; M, M')$: out-of-place matrix multiplication algorithm type 2

Fixed input : $n_2 \times n_1$ binary matrices M, M'
Quantum input : n_1, n_2-qubit vectors $|A\rangle, |B\rangle$ stored in $\mathbf{A}$ and $\mathbf{B}$, respectively
Quantum output: $\mathbf{B}$ stores the output $|A\rangle |M'B + MA\rangle$

// S_{Col} and S_{CNOT} can be pre-computed by greedy approach with M.
// M' is determined by S_{Row} and S_{CNOT}.
for $Row \in mathcal R_M$ **do**
  for $(c, t) \in S_{\mathrm{CNOT}}(Row)$ **do**
    // (c, t) are control and target qubit indices of $\mathbf{C} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$.
    // c is the index on any parts of $\mathbf{C}$, whereas $t \geq n_1$ is the index on the $\mathbf{B}$ part of $\mathbf{C}$.
    $\mathbf{B}[t - n_1] \leftarrow \mathrm{CNOT}(\mathbf{C}[c], \mathbf{B}[t - n_1])$

Reusing Intermediate Values. $\textsc{OutMultReuse}$ uses both of $|A\rangle$ and $|B\rangle$ as control qubits. This can be seen as a linear straight-line program where intermediate values are continuously updated can be reused in the process of computation [6]. Every solutions of linear straight-line program can be converted to quantum circuit if intermediate variables are less than n_2. Since it is difficult to find the *shortest* solution of linear straight-line program, we used a greedy approach. This approach selects each row of M by the minimum implementation cost considering the intermediate values (input $|A\rangle$ and continuously updated $|B\rangle$). For $n_2 \times n_1$ binary matrix M, this greedy approach gives the optimal sequence of rows of M as

$$\mathcal{R}_M = \{Row_0, \dots, Row_{n_2-1}\}$$

and the sequence of control and target qubit indices of $|A\rangle \times |B\rangle$ for the CNOT gates to generate each column:

$$S_{\mathrm{CNOT}}(Row_i) = \{(c_0, t_0), \cdots, (c_{t-1}, t_{t-1})\}.$$

Algorithm 4 describes this out-of-place multiplication $\textsc{OutMultReuse}$. In contrast with $\textsc{OutMultNaïve}$, $\textsc{OutMultReuse}$ gives $|A\rangle |M'B + MA\rangle$, where M'

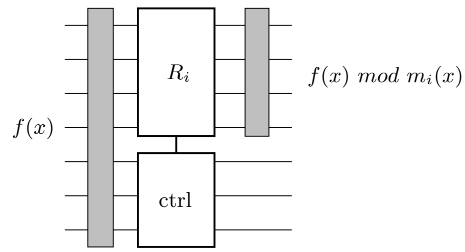


Fig. 5. Modular Reduction

is determined by S_{Row} and S_{CNOT}. However, one does not need to care of $|M'B\rangle$ because this part will be discarded by † operation during CRTModMult. Note that, unlike OutMultNaïve, it is hard to merge two OutMultNaïve to one OutMultNaïve or other simple circuits because of the $|M'B\rangle$ term.

5.4 Proposed Circuit

We propose binary field multiplication circuit named CRTModMult which follows the CRT based multiplication steps in chapter 4. Algorithm 5 in Appendix A gives CRTModMult. We define $d_i = deg(m_i(x))$ and assume $m_1(x) = x^{d_1}$ and $d_2 \leq d_3 \leq ... \leq d_t$ for efficiency of modular reduction.

Modular Reduction. Modular reduction for polynomial $m_i(x)$ can be represented as $(d-d_i)\times d_i$ reduction matrix R_i whose k th column is $x^{k+d_i}\ mod\ m(x)$. R_i is implemented using out-of-place matrix multiplication circuit and this step is depicted in Fig 5.

Modular Multiplication and CRT. Let T_i, R_i be matrix corresponding to symmetric bilinear form of d_i-split Karatsuba-like formula. Then multiplication over mod $m_i(x)$ can be written in symmetric bilinear form using T_i, R_i' matrix where R_i' is mod $m_i(x)$ reduced version of R_i. we define $MODMULT_i := SBF_{T_i,R_i'}$. For CRT operation, we should compute $(C_i(x)h_i(x) \bmod m(x)) \bmod p(x)$ which can be expressed as matrix form $H_i^p C_i$ where H_i^p is $n \times d_i$ matrix whose k th column is $(h_i(x)x^k\ mod\ m(x)) \bmod p(x)$. Choosing d_i linearly independent columns, H_i^p can be decomposed into three matrices:

$$H_i^p = S_i^p \begin{bmatrix} M_i^p \\ N_i^p \end{bmatrix},$$

where $n \times n$ permutation matrix S_i^p, $d_i \times d_i$ matrix M_i^p and $n - d_i \times d_i$ matrix N_i^p. M_i^p and N_i^p correspond to in-place and out-of-place matrix multiplication, respectively. † operation is needed before modular multiplication to remove the effects of the values from previous step and this is depicted in Table 4. Permutation matrix S_i^p can be implemented implicitly without SWAP gate cost by

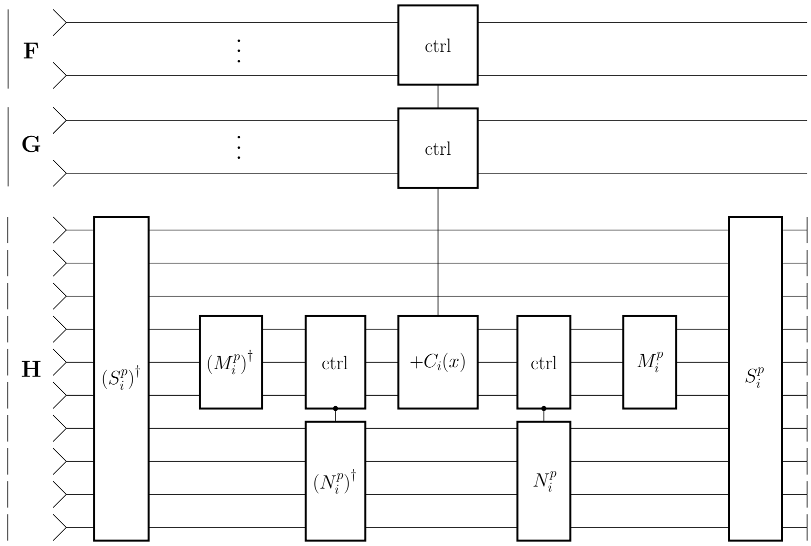


Fig. 6. One Step of Modular Multiplication and CRT

changing the positions of control bits and target bits and this step is written as Permutation($\mathbf{H}[0:n]; S_i^P$) in the algorithm. Note that ModMult algorithm also does not consider cost of permutation. One step of modular multiplication and CRT is depicted in Fig 6.

Applying Modulo $(x-\infty)^w$. This step can be expressed as matrix form

$$C = C' + H_\infty^p \begin{bmatrix} c_{2n-w} \\ | \\ c_{2n-1} \end{bmatrix},$$

where H_∞^p is $n \times w$ matrix whose k th column is $(((x^k)+(x^k \bmod m(x))) \bmod P(x)$. Like CRT operation, H_∞^p be decomposed into three matrices : $n \times n$ permutation

Table 4. Step-by-step calculation of Modular Multiplication and CRT step. N' denotes the matrix in OutMultReuse. $N' = I$ if OutMultNaïve is used

STEP	$\mathbf{H}[0:d_i]$	$\mathbf{H}[d_i:n]$
initial value	A	B
InMULT$^\dagger(M)$	$M^{-1}A$	B
OutMult$^\dagger(N)$	$M^{-1}A$	$(N')^{-1}(B+N(M^{-1}A))$
$+C_i(x)$	$M^{-1}A+C$	$(N')^{-1}(B+N(M^{-1}A))$
OutMult(N)	$M^{-1}A+C$	$B+NC$
InMULT(M)	$A+MC$	$B+NC$

matrix S_i^p , $w \times w$ matrix M_i and $(n-w) \times w$ matrix N_i.

$$H_\infty^p = S_\infty^p \begin{bmatrix} M_\infty^p \\ N_\infty^p \end{bmatrix}$$

Similar to CRT, we can construct M_∞^p and N_∞^p using in-place, out-of-place matrix multiplication respectively.

5.5 What Out-of-place multiplications to use?

Modular reduction Matrix R_i for modular reduction has size $n \times d_i$ where $d_i \approx log(n)$. Therefore it is less likely that rows of R_i collides, which means that using OutMultReuse will not be very efficient. Instead, we used OutMultNaïve to merge consecutive R_i and R_{i+1} to $R_i + R_{i+1}$ and remove duplicated CNOT gates.

CRT and $(x-\infty)^w$ Matrix N_i^p for CRT has size $(n-d_i) \times d_i$. Since it is hard to compute average cost of OutMultReuse, we will assume the worst case where all 2^{d_i} linear combination exists in rows of N_i^p. In such worst case, OutMultReuse will use $n + 2^{d_i} - d_i \approx 2n$ CNOT gates in total where 2×2^{d_i} CNOT gates for generating gray code, $n - d_i - 2^{d_i}$ CNOT gates are copying to others. Considering that average cost of OutMultNaïve is $((n-d_i) \times d_i)/2 \approx n \times (d_i/2)$, OutMultReuse will definitely outperform OutMultNaïve in most of case if $4 \leq d_i$. Therefore we will use OutMultReuse for N_i^p and N_∞^p likewise. One can think of the case where N_i^p and N_{i+1}^p are merged using OutMultNaïve, but this will be hard because of the permutation S_∞^p.

5.6 Asymptotic Cost for CRT-Based Mod Multiplication

This chapter proves asymptotic cost of CRTModMult algorithm : $O(n2^{log^* n})$ TOF gate and $O(n^2)$ CNOT gates. The number of CNOT gates depends on the choice of m_i, but we assume that every matrix is randomly selected (i.e., half of the entries are 1).

TOF Gate Count.

$$\begin{aligned} M(n) &\leq \sum_{i=1}^{t} M(deg(m_i)) + \lambda(w) \\ &\approx t \times M(\lfloor \log_2(n) \rfloor) \\ &\approx \frac{2n}{\log_2(n)} M(\lfloor \log_2(n) \rfloor) \\ \Rightarrow \quad \frac{M(n)}{n} &\leq 2 \times \frac{M(\lfloor \log_2(n) \rfloor)}{\log_2(n)} \end{aligned} \tag{1}$$

Applying the above inequality (1) iteratively, we get the bound $M(n) \leq O(n2^{\log_2^* n})$, where $\log_2^*$ is iterative logarithm.

CNOT Gate Count.

1. **modular reduction**

$$\sum_{i=1}^{t}(deg(m_i) \times (n - deg(m_i)) \approx (2n/\log_2(n) \times \log_2(n)) \times n = O(n^2)$$

2. **modular multiplication(induction applied)**

$$O(\sum_{i=1}^{t} deg(m_i)^2 + w^2) = O((2n-1)/\log_2(n) \times (\log_2(n))^2) = O(n\log_2(n))$$

3. **CRT** (assumed that OutMultWide is used)

$$(\sum_{i=1}^{t} deg(m_i))(2n-1-w)/2 \leq (2n \times 2n)/2 = O(n^2)$$

4. **modulo** $(x - \infty)^w$

$$2w(2n-1-w) + 2(w-1) + w^2 = O(n\log_2(n))$$

summing up, we get CNOT count $O(n^2)$.

6 Results

In this section we evaluate gate complexity and depth of of CRTModMult and compare it to ModMult_Imp, which is most recent space-efficient quantum binary field multiplication circuit [19]. Multiplication circuit in [12] is not comparable to our circuit because the optimization target is different : depth and TOF-gate, respectively. CRTModMult is implemented in Qiskit and built-in function of Qiskit is used to compute the complexity. The Result and irreducible polynomials we used are given in Table 5 and Table 7 in Appendix B, respectively. Table 5 shows that TOF gate grows almost linearly and uses only $(12 \sim 24)\%$ in cryptographic field size ($n = 233 \sim 571$) compared to the previous ModMult_Imp. In terms of CNOT gates, CRTModMult requires about 3×2^n, which is $(3 \sim 4)$ times more then ModMult_Imp. Despite the increased CNOT gates, CRTModMult has even low depth in cryptographic field size due to the depth optimization in R matrix.

For more precise comparison, we converted a TOF gate to 7 T-gates and 8 Clifford gates[3] and the result are in Table 6. For field size ($n = 233 \sim 571$), CRTModMult will outperform ModMult_Imp if the cost of T-gate is more than twice high as the cost of Clifford gates. Although the exact comparison between T-gate and Clifford gate costs is hard to examine because of its dependency on physical system and fault tolerant implementation, it is commonly recognized that T-gate has significantly higher cost than Clifford gates. Therefore, we think that CRTModMult will outperform ModMult_Imp.

Table 5. CNOT and TOF gate count and depth upper bounds for various instances of ModMult_Imp and CRTModMult

Degree	This Work CRTModMult			Previous Work ModMult_Imp		
	TOF	CNOT	Depth	TOF	CNOT	Depth
16	64	974	405	81	655	286
32	149	3604	1018	243	2153	855
64	337	13022	2595	729	6728	2468
127	737	49040	6953	2183	20300	7000
128	740	49632	6879	2187	20838	7071
163	992	76262	10210	4355	36439	13814
233	1441	154892	16383	6307	60453	19294
256	1590	184948	18504	6561	63689	20188
283	1784	224246	22050	10241	87929	31894
571	3813	862604	61771	31139	267771	95863
1024	6978	2740484	257684	59049	585331	180193

Table 6. T gate and Clifford gate count for various instances for various instances of ModMult_Imp and CRTModMult

Degree	This Work CRTModMult		Previous Work ModMult_Imp	
	T gate	Clifford gate	T gate	Clifford gate
16	448	1486	567	1303
32	1043	4796	1701	4097
64	2359	15718	5103	12560
127	5159	54936	15281	37764
128	5180	55552	15309	38334
163	6944	84198	30485	71279
233	10087	166420	44149	110909
256	11130	197668	45927	116177
283	12488	238518	71687	169857
571	26691	893108	217973	516883
1024	48846	2796308	413343	1057723

7 Conclusion

In this paper, we present TOF-count optimized space-efficient binary field multiplication circuit CRTModMult which provides extremely low TOF gate count $O(n2^{\log_2^*(n)})$. Our circuits are based on Karatsuba-like formulas and CRT-based multiplication that have not previously been applied to quantum circuits. The number of TOF gates are fixed for CRT-based multiplication whereas the number of CNOTs can be reduced. Therefore we optimized Karatsuba-like formula in terms of CNOT gate and depth. Also two out-of-place multiplication OutMultReuse,OutMultnaïve are used to reduce CNOT gate cost in CRT-based multiplication. For cryptographic field sizes, CRTModMult reduces $76 \sim 88\%$ of TOF gates compared to recent results. Considering high cost of TOF gate and comparable depth, CRTModMult can be used to enhance quantum cryptanalysis of ECDSA.

References

1. A. Weimerskirch, C.P.: "generalizations of the karatsuba algorithm for efficient implementations. IEEE Transactions on Computers **54**(3), 362–369 (2003)
2. Amento, B., Rötteler, M., Steinwandt, R.: Quantum binary field inversion: improved circuit depth via choice of basis representation. arXiv preprint arXiv:1209.5491 (2012)
3. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**(6), 818–830 (2013)
4. Banegas, G., Bernstein, D.J., Van Hoof, I., Lange, T.: Concrete quantum cryptanalysis of binary elliptic curves. Cryptology ePrint Archive (2020)
5. Banegas, G., Custódio, R., Panario, D.: A new class of irreducible pentanomials for polynomial-based multipliers in binary fields. Journal of Cryptographic Engineering **9**(4), 359–373 (2019)
6. Boyar, J., Matthews, P., Peralta, R.: On the shortest linear straight-line program for computing linear forms. In: International Symposium on Mathematical Foundations of Computer Science. pp. 168–179. Springer (2008)
7. Çalık, Ç., Dworkin, M., Dykas, N., Peralta, R.: Searching for best karatsuba recurrences. In: International Symposium on Experimental Algorithms. pp. 332–342. Springer (2019)
8. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of elliptic and hyperelliptic curve cryptography. CRC press (2005)
9. Fan, H., Hasan, M.A.: Comments on" five, six, and seven-term karatsuba-like formulae. IEEE Transactions on Computers **56**(5), 716–717 (2007)
10. Find, M.G., Peralta, R.: Better circuits for binary polynomial multiplication. IEEE Transactions on Computers **68**(4), 624–630 (2018)
11. van Hoof, I.: Space-efficient quantum multiplication polynomials for binary finite fields with sub-quadratoc toffoli gate count. Quantum Information and Computation **20**(9&10), 721–735 (2020)
12. Jang, K., Kim, W., Lim, S., Kang, Y., Yang, Y., Seo, H.: Optimized implementation of quantum binary field multiplication with toffoli depth one. In: International Conference on Information Security Applications. pp. 284–297. Springer (2022)
13. Kepley, S., Steinwandt, R.: Quantum circuits for f_{2^n}-multiplication with subquadratic gate count. Quantum Information Processing **14**(7), 2373–2386 (2015)
14. Kerry, C.F., Gallagher, P.D.: Digital signature standard (dss). FIPS PUB pp. 186–4 (2013)
15. Maslov, D., Mathew, J., Cheung, D., Pradhan, D.K.: An o (m2)-depth quantum algorithm for the elliptic curve discrete logarithm problem over gf (2m) a. Quantum Information & Computation **9**(7), 610–621 (2009)
16. Metropolis, N., Rota, G.C.: Witt vectors and the algebra of necklaces. Advances in Mathematics **50**(2), 95–125 (1983)
17. Montgomery, P.L.: Five, six, and seven-term karatsuba-like formulae. IEEE Transactions on Computers **54**(3), 362–369 (2005)
18. Moreau, C.: Sur les permutations circulaires distinctes. Nouvelles annales de mathématiques: journal des candidats aux écoles polytechnique et normale **11**, 309–314 (1872)

19. Putranto, D.S.C., Wardhani, R.W., Larasati, H.T., Kim, H.: Another concrete quantum cryptanalysis of binary elliptic curves. Cryptology ePrint Archive (2022)
20. Seroussi, G.: Table of low-weight binary irreducible polynomials. Citeseer (1998)

A The Proposed Algorithm

Algorithm 5: $\text{CRTModMult}(\mathbf{F}, \mathbf{G}, \mathbf{H}; m(x), w, p(x))$

Fixed input : $m(x) = \prod_{n=1}^{t} m_i(x), w = 2n - 1 - deg(m), p(x)$
Quantum input : Two binary n term polynomials f, g store in array $\mathbf{F}$ and $\mathbf{G}$ respectively of size n, A binary polynomial h stored in array $\mathbf{H}$ of size $2n - 1$
Quantum output: $\mathbf{H}$ as $h + fg \bmod p(x)$

for $i = 1..t$ **do**
  `// duplicated CNOT gates with next loop will be removed`
  $\text{OutMultWide}(\mathbf{F}[d_i : n], \mathbf{F}[0 : d_i]; R_i)$
  $\text{OutMultWide}(\mathbf{G}[d_i : n], \mathbf{G}[0 : d_i]; R_i)$
  $\text{Permutation}^\dagger(\mathbf{H}[0 : n]; S_i^p)$ `// implicit permutation`
  $\text{InMult}^\dagger(\mathbf{H}[0 : d_i]; M_i^p)$
  $\text{OutMultNarrow}^\dagger(\mathbf{H}[0 : d_i], \mathbf{H}[d_i : n]; N_i^p)$
  `//` $3 \sim 8$ `Karatsuba-like formula is used`
  if $d_i \leq 8$ **then**
    $T, R \leftarrow$ symmetric bilinear form of d_i-split karatsuba like formula
    $R_i \leftarrow$ mod m_i reduced matrix of R
    $SBF(T, R_i)(F[0 : d_i], G[0 : d_i], H[0 : d_i])$
  `// recursive call for` CRTModMult `for degree larger than 8`
  else
    $m'(x), w' \leftarrow$ predefined parameters for degree d_i multiplication
    $\text{CRTModMult}(\mathbf{F}[0 : d_i], \mathbf{G}[0 : d_i], \mathbf{H}[0 : d_i]; m'(x), w', m_i(x))$
  $\text{OutMultNarrow}(\mathbf{H}[0 : d_i], \mathbf{H}[d_i : n]; N_i^p))$
  $\text{InMult}(\mathbf{H}[0 : d_i]; M_i^p)$
  $\text{Permutation}(\mathbf{H}[0 : n]; S_i^p)$
  $\text{OutMultWide}(\mathbf{F}[d_i : n], \mathbf{F}[0 : d_i]; R_i)$
  $\text{OutMultWide}(\mathbf{G}[d_i : n], \mathbf{G}[0 : d_i]; R_i)$
`// Modulo` $(x - \infty)^w$ `step`
$\text{Permutation}^\dagger(\mathbf{H}[0 : n]; S_\infty^p)$
$\text{InMult}^\dagger(\mathbf{H}[0 : w]; M_\infty^p)$
$\text{OutMultNarrow}^\dagger(\mathbf{H}[0 : w], \mathbf{H}[w : n]; N_\infty^p)$
$\text{HighDeg}_{w,n}\ (\mathbf{F}[n - w : n], \mathbf{G}[n - w : n], \mathbf{H}[0 : n])$
$\text{OutMultNarrow}(\mathbf{H}[0 : w], \mathbf{H}[w : n]; N_\infty^p)$
$\text{InMult}(\mathbf{H}[0 : w]; M_\infty^p)$
$\text{Permutation}(\mathbf{H}[0 : n]; S_\infty^p)$

B Considered Irreducible Polynomials

Table 7. Considered Irreducible Polynomials

Degree	Irreducible Polynomial	Ref.
16	$x^{16} + x^5 + x^3 + x + 1$	[8]
32	$x^{32} + x^7 + x^3 + x^2 + 1$	[8]
64	$x^{64} + x^4 + x^3 + x + 1$	[8]
127	$x^{127} + x + 1$	[8]
128	$x^{128} + x^7 + x^2 + x + 1$	[8]
163	$x^{163} + x^7 + x^6 + x^3 + 1$	[14]
233	$x^{233} + x^{74} + 1$	[14]
256	$x^{256} + x^{10} + x^5 + x^2 + 1$	[8]
283	$x^{283} + x^{12} + x^7 + x^5 + 1$	[5]
571	$x^{571} + x^{10} + x^5 + x^2 + 1$	[14]
1024	$x^{1024} + x^{19} + x^6 + x^1 + 1$	[20]