New Residue Arithmetic Based Barrett Algorithms: Modular Polynomial Computations

Hari Krishna Garg
Electrical & Computer Engineering Department
National University of Singapore
eleghk@nus.edu.sg

Hanshen Xiao
Department of Mathematics
Tsinghua University
hsxiao@mit.edu

Abstract— **We derive a new computational algorithm for Barrett technique for modular polynomial multiplication, termed BA-P. Residue arithmetic is applied to BA-P to obtain a new Barrett algorithm for modular polynomial multiplication (BA-MPM). The work is focused on an algorithm that carries out computation using modular arithmetic without conversion to large degree polynomials. There are several parts to this work. First, we set up a new BA-P using polynomials other than u^{α}. Second, residue arithmetic based BA-MPM is described. A complete mathematical framework is described including proofs for the results. Third, we present a computational procedure for BA-MPM. Fourth, the BA-MPM is used as a basis for algorithms for modular polynomial exponentiation (MPE). Applications are in areas of signal security and cryptography.**

Keywords—Barrett Algorithm (BA), BA for Polynomials (BA-P), Modular Polynomial Multiplication (MPM), Montgomery Multiplication (MM), Residue Polynomial Systems (RPS), Chinese Remainder Theorem for Polynomials (CRT-P), BA-P based on MPM (BA-MPM), Modular Polynomial Exponentiation (MPE), Base Extension for Polynomials (BEX-P).

I. INTRODUCTION

CRYPTOGRAPHY TECHNIQUES play an important role in the security of electronic systems. Instances of such cryptography techniques include RSA (Rivest-Shamir-Adelman), Rabin, Diffie Hellman and El Gamal. These techniques deal with arithmetic defined in a large size finite fields GF(p) and/or GF(p^N), where p is a prime integer and N is an integer. A large size field may be realized by setting $p = 2$ (binary arithmetic) and N to be a large value. Here, we also deal with finite fields with large values of N, say 500 to 5,000. The elements in GF(2^N) are expressed as polynomials over GF(2) of degree up to $N-1$. A challenge is to perform the following computations efficiently:

1. Multiplication of two elements in GF(2^N):
   $C(u) = A(u) \cdot B(u) \pmod{P(u)}$; $\deg(P(u)) = N$; and
2. Modular exponentiation in GF(2^N):
   $C(u) = A(u)^E \pmod{P(u)}$; $\deg(P(u)) = N$.

Here, $P(u)$ is an irreducible (or primitive) polynomial in GF(2). This eliminates use of Chinese remainder theorem for polynomials (CRT-P) to compute $C(u)$. Computations in 1 and 2 without mod $P(u)$ are simpler while mod $P(u)$ computation is challenging. Also modular polynomial exponentiation (MPE) is computed via repeated use of modular polynomial multiplication (MPM). Hence, an efficient algorithm must be used for MPM. In many situations, N is large.

Residue arithmetic is used to express a large size ring as a direct product of a number of smaller size rings. Residue number systems have been applied in Barrett algorithm (BA) and Montgomery multiplication (MM), to compute modular operations in large size integer rings. However, there is a distinct gap when it comes to Residue Polynomial Systems (RPS) based BA and MM.

Contributions of the work are as follows. The primary objective is to compute MPM and MPE efficiently for applications in signal security and cryptography. We first describe a new BA for modular polynomial multiplication (BA-P) for computing the quotient $C(u)$ associated with $X(u)$ when it is divided by $P(u)$. It is assumed that $N = \deg(P(u))$ is a large integer. Second, a residue arithmetic based BA-P, termed BA-MPM, is described for modular polynomial multiplication. Third, a computationally efficient procedure for the new BA-MPM is described. Fourth, the new BA-MPM is used as a basis for MPE. The results are general and valid for all fields such as GF(p^N), rational, real, and complex numbers.

There is an abundance of research on MM and BA [1]-[13]. Polynomial versions of MM and BA can be found in [14]-[22]. A digit-serial multiplication in GF(2^N) based on Barrett modular reduction is presented in [15]. A version of digit-serial multiplication algorithm is described in [16]. Other aspects are explored in [19]. Further details of BA and MM are available in [23]-[28].

However, there is no paper on using residue arithmetic to compute MPM and MPE via BA. It is this particular aspect that we deal with in this paper. The algorithm described here begins with reformulating BA such that the new BA-P stays within residue arithmetic.

The organization of this paper is as follows. Section II provides mathematical preliminaries on arithmetic, BA-P, RPS, CRT-P, and base extension for polynomials (BEX-P). The new BA-P is described in Section III. The computational steps for RPS based BA-P algorithm are presented in Section IV. Examples are presented to illustrate the algorithm. In Section V, we describe an algorithm for MPE that uses the new BA-MPM. Section VI is on conclusions.

II. MATHEMATICAL PRELIMINARIES

Polynomial Arithmetic. Given $X(u)$ and $A_1(u)$ with coefficients in a field **F**, consider dividing $X(u)$ by $A_1(u)$ to write

$$X(u) = Q_1(u) \cdot A_1(u) + R_1(u). \tag{1}$$

Here, $Q_1(u)$ is quotient and $R_1(u)$ is remainder. Also, $\deg(R_1(u)) < \deg(A_1(u))$ with $\deg(Q_1(u)) = \deg(X(u)) - \deg(A_1(u))$. We write (1) as

$$X(u) \equiv R_1(u) \bmod A_1(u). \tag{2}$$

Dividing both sides of (1) by $A_1(u)$, we get,

$$X(u) / A_1(u) = Q_1(u) + R_1(u) / A_1(u). \tag{3}$$

The last term on the right when expressed as a sum of powers of u will only contain negative powers. We also write

$$Q_1(u) = \lfloor X(u) / A_1(u) \rfloor, \tag{4}$$

$\lfloor Y(u) \rfloor$ being the floor function of $Y(u)$. $Q_1(u)$ and $R_1(u)$ are unique. This process can be repeated between $Q_1(u)$ and $A_2(u)$. Thus

$$Q_1(u) = Q_2(u) \cdot A_2(u) + R_2(u), \tag{5}$$

$0 \le \deg(R_2(u)) < \deg(A_2(u))$. A generalization of (5) leads to

$$X(u) = Q_a(u) \cdot [(A_a(u) \cdots A_1(u)] + [R_a(u) \cdot \{A_{a-1}(u) \cdots A_1(u)\} + \cdots + R_2(u) \cdot A_1(u) + R_1(u)]. \tag{6}$$

This expression is useful in computations involving RPS and BEX-P.

Barrett Algorithm for Polynomials (BA-P). Given $A(u)$, $B(u)$, and $P(u)$ in GF(p), deg($A(u)$), deg($B(u)$) < deg($P(u)$) = N, MPM computes:

$$C(u) = A(u) \cdot B(u) \bmod P(u). \quad (7)$$

Let $X(u) = A(u) \cdot B(u)$. BA-P computes quotient $Q(u)$ such that $X(u) = Q(u) \cdot P(u) + C(u)$; deg($C(u)$) < N. Then $C(u)$ is computed as $C(u) = X(u) - Q(u) \cdot P(u)$. Given $X(u)$ and $P(u)$ BA-P expresses $Q(u)$ as

$$Q(u) = \lfloor X(u) / P(u) \rfloor, \quad (8)$$

In the current versions of BA-P [14]-[22], (8) is computed as

$$Q(u) = \lfloor X(u) / u^a \cdot \Omega(u) / u^{a+b} \rfloor \approx \lfloor \lfloor X(u) / u^a \rfloor \cdot \lfloor \Omega(u) \rfloor / u^b \rfloor, \quad (9)$$

where $\lfloor \Omega(u) \rfloor$ is pre-computed as $\mu(u) = \lfloor \Omega(u) \rfloor = \lfloor u^{a+b} / P(u) \rfloor$. Scalars a and b are chosen such that $Q(u)$ in (9) is same as $Q(u)$ in (8) [15, 16]. BA-P consists of steps:

0: Pre-compute $\mu(u) = \lfloor u^{a+b} / P(u) \rfloor$;

Compute:

1: $X(u) = A(u) \cdot B(u)$; **2**: $D(u) = \lfloor X(u) / u^a \rfloor$; **3**: $E(u) = D(u) \cdot \mu(u)$; **4**: $Q(u) = \lfloor E(u) / u^b \rfloor$; **5**: $C(u) = X(u) - Q(u) \cdot P(u)$.

Residue Polynomial System (RPS) [23, 29]. A RPS defined mod $M(u)$ is a ring defined by n co-prime polynomials $M_1(u)$, $M_2(u)$, ..., $M_n(u)$ with elements in field **F**. The elements in RPS are polynomials of degree up to $L - 1$, L = deg($M(u)$), where

$$M(u) = \prod_{i=1}^{n} M_i(u). \quad (10)$$

A polynomial $X(u)$ in the RPS is represented as n residues,

$$X(u) \leftrightarrow \mathbf{X}(\boldsymbol{u}) \leftrightarrow [X_1(u)\, X_2(u) \cdots X_n(u)], \quad (11)$$

where $X_i(u) \equiv X(u)$ (mod $M_i(u)$), i = 1, 2, ..., n.

Chinese remainder theorem for polynomials (CRT-P) [2, 3, 23, 29]. Given $\mathbf{X}(\boldsymbol{u})$, $X(u)$, deg($X(u)$) < L, is computed via CRT-P, as

$$X(u) \equiv \sum_{i=1}^{n} T_i(u) \cdot X_i(u) \left(\bmod M_i(u)\right) \cdot \left(\frac{M(u)}{M_i(u)} \right) . \quad (12)$$

Polynomials $T_i(u)$, deg($T_i(u)$) < deg($M_i(u)$), are computed a-priori via $T_i(u) \cdot \left(\frac{M(u)}{M_i(u)} \right) \equiv 1 \left(\bmod M_i(u) \right)$, i = 1, 2, ..., n. CRT-P computation of $X(u)$ involves large degree polynomials.

Base Extension for Polynomials (BEX-P). Consider $\mathbf{X}(\boldsymbol{u})$, residues of $X(u)$ in (11). BEX-P consists in computing t additional residues of $X(u)$, $X_j(u) \equiv X(u)$ (mod $M_j(u)$), $j = n + 1, ..., n + t$, in a RPS defined mod $M_{\mathrm{I}}(u)$, $M_{\mathrm{I}}(u) = \prod_{j=n+1}^{n+t} M_j(u)$, where gcd($M(u)$, $M_{\mathrm{I}}(u)$) = 1. BEX-P is intense computationally. Using (12), we compute it as [22]:

$$X_j(u) \equiv \left\{ \sum_{i=1}^{n} T_i(u) \cdot X_i(u) \bmod M_j(u) \cdot \frac{M(u)}{M_i(u)} \right\} \bmod M_j(u),$$

$$j = n + 1, ..., n + t. \quad (13)$$

III. A New Barrett Algorithm for Polynomials

A RPS based Montgomery multiplication algorithm has been described in [22]. However, there is no such algorithm for the BA-P. We cite [15, 16, 19] and the references therein. They have used modulo polynomials of the type u^a. Clearly, this doesn't lend itself to RPS. Here, we first revisit the computation of $Q(u)$ in (8). We now introduce two polynomials $G(u)$ and $H(u)$, not necessarily of the form u^a, and approximate $Q(u)$ in (8) as

$$Q(u) = \lfloor X(u) / G(u) \cdot \Omega(u) / H(u) \rfloor$$
$$\approx \lfloor \lfloor X(u) / G(u) \rfloor \cdot \mu(u) / H(u) \rfloor. \quad (14)$$

$\mu(u)$ is pre-computed as $\mu(u) = \lfloor \Omega(u) \rfloor = \lfloor G(u) \cdot H(u) / P(u) \rfloor$. Since $X(u) = A(u) \cdot B(u)$, deg($A(u)$), deg($B(u)$) < N, deg($X(u)$) ≤ 2 · N – 2.

Now we derive conditions on $G(u)$ and $H(u)$ for approximation of $Q(u)$ in (14) to be equal to $Q(u)$ in (8). $\lfloor V(u) \rfloor$ is polynomial part of $V(u)$ consisting of terms with positive powers of u. Consider dividing $V(u)$ by $S(u)$ to write $V(u) = Q(u) \cdot S(u) + R(u)$. Then we have

$$V(u) / S(u) = Q(u) + R(u) / S(u)$$
$$= \lfloor V(u) / S(u) \rfloor + \delta(u),\ \deg(\delta(u)) \le -1. \quad (15)$$

Applying (15) to (14), we get

$$Q(u) = \left\lfloor \frac{\left(\left\lfloor \frac{X(u)}{G(u)} \right\rfloor + \delta(u) \right) \cdot \left(\left\lfloor \frac{G(u) \cdot H(u)}{P(u)} \right\rfloor + \phi(u) \right)}{H(u)} \right\rfloor \quad (16)$$

$$= \lfloor A + B \rfloor,$$

where $A = \dfrac{\left\lfloor \frac{X(u)}{G(u)} \right\rfloor \cdot \mu(u)}{H(u)}$, and

$$B = \frac{\left\lfloor \frac{X(u)}{G(u)} \right\rfloor \cdot \phi(u) + \left\lfloor \frac{G(u) \cdot H(u)}{P(u)} \right\rfloor \cdot \delta(u) + \delta(u) \cdot \phi(u)}{H(u)}.$$

Here, deg($\delta(u)$), deg($\phi(u)$) ≤ –1. We wish the second term in the above summation to have degree less than 0. To achieve that,

(A) $\deg\left(\left\lfloor \frac{X(u)}{G(u)} \right\rfloor \right) \le \deg(H(u))$;

(B) $\deg\left(\left\lfloor \frac{G(u) \cdot H(u)}{P(u)} \right\rfloor \right) \le \deg(H(u))$.

We assume these conditions to be satisfied. Thus (16) becomes:

$$Q(u) = \left\lfloor \frac{\left\lfloor \frac{X(u)}{G(u)} \right\rfloor \cdot \mu(u)}{H(u)} + \Delta(u) \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{X(u)}{G(u)} \right\rfloor \cdot \mu(u)}{H(u)} \right\rfloor, \quad (17)$$

as deg($\Delta(u)$) ≤ –1. We note that deg($\mu(u)$) = deg($G(u)$) + deg($H(u)$) – deg($P(u)$). This analysis leads to the following theorem:

Theorem 1. Let $A(u)$, $B(u)$ and $P(u)$ be given such that 0 ≤ deg($A(u)$), deg($B(u)$) < deg($P(u)$) = N. For the computation $X(u)$ mod $P(u)$, $X(u) = A(u) \cdot B(u)$, if $G(u)$ and $H(u)$, α = deg($G(u)$) and β = deg($H(u)$), satisfy the conditions

$$\deg(X(u)) \le 2N - 2 \le \alpha + \beta;\ \alpha \le \deg(P(u)) = N, \quad (18)$$

then $Q(u)$ in (17) is same as the quotient $\lfloor X(u) / P(u) \rfloor$.

A generalization of $G(u)$ and $H(u)$ from polynomials of the type u^a is crucial. A choice of degrees that satisfy (18) is α = β = N. $G(u)$ and $H(u)$ can be identical. This analysis leads to the following new BA-P:

A New Barrett Algorithm for $A(u) \cdot B(u)$ mod $P(u)$ (BA-P)

Input: $A(u)$, $B(u)$, $P(u)$, $G(u)$, $H(u)$; 0 ≤ deg($A(u)$), deg($B(u)$) < N, N = deg($P(u)$), α = deg($G(u)$), β = deg($H(u)$).

Output: $Q(u)$ (quotient when $A(u) \cdot B(u)$ is divided by $P(u)$)

Step 0. Pre-compute $\mu(u) = \lfloor G(u) \cdot H(u) / P(u) \rfloor$,

deg($\mu(u)$) = α + β – N (one-time)

Compute

Step 1. $X(u) = A(u) \cdot B(u)$, $\deg(X(u)) \le 2 \cdot N - 2$ (ordinary mult)
Step 2. $D(u) = \lfloor X(u) / G(u) \rfloor$, $\deg(D(u)) \le 2 \cdot N - \alpha - 2$ (quotient)
Step 3. $E(u) = D(u) \cdot \mu(u)$, $\deg(E(u)) \le N + \beta - 2$ (ordinary mult)
Step 4. $Q(u) = \lfloor E(u) / H(u) \rfloor$, $\deg(Q(u)) \le N - 2$ (quotient)
Once $Q(u)$ is computed, remainder $X(u)$ mod $P(u)$ is computed as
Step 5. $C(u) = X(u) - Q(u) \cdot P(u)$, $\deg(C(u)) \le N - 1$. (ordinary mult)
The conditions in (18) required for $G(u)$ and $H(u)$ are general and open door to a range of possibilities for different computational steps.

Example 1. Assume that the computation is defined in GF(2). Let $N = 6$, $P(u) = u^6 + u + 1$. We can choose $G(u) = u^6 + 1$, $H(u) = u^6 + 1$. Then $\mu(u) = u^6 + u + 1$. Let $X(u) = u^{10} + u^9 + u^8 + u^4 + u^2 + 1$. We have $D(u) = u^4 + u^3 + u^2$, $E(u) = u^{10} + u^9 + u^8 + u^5 + u^2$. $Q(u) = u^4 + u^3 + u^2$, $Q(u) \cdot P(u) = u^{10} + u^9 + u^8 + u^5 + u^2$, $C(u) = u^5 + u^4 + 1$.

Example 2. Let the computation be defined in GF(2) with $G(u) = H(u)$ and $\mu(u) = P(u)$. Then, $G(u)^2 = P(u)^2 + R(u)$, $\deg(R(u)) < N$. If $G(u) = \sum_{i=0}^{N} G_i u^i$ and $P(u) = \sum_{i=0}^{N} P_i u^i$, then $G(u)^2 = \sum_{i=0}^{2N} G_i u^{2i}$ and $P(u)^2 = \sum_{i=0}^{2N} P_i u^{2i}$. A trivial solution to $G(u)^2 = P(u)^2 + R(u)$ is $G(u) = P(u)$, $R(u) = 0$. Other possibilities are $R(u) = \sum_{i=0}^{(N-1)/2} R_i u^{2i}$, where $R_i \in$ GF(2) and can take any values. This results in $G_i = P_i + R_i$, $i = 0, ..., (N - 1) / 2$; $G_i = P_i$, $i = (N + 1) / 2, ..., N$.

Analysis in Example 2, though applicable to GF(2) only, can be used to identify other desirable forms for $G(u)$ and $H(u)$ for a given $P(u)$ including those that require no pre-computation [19]. Analysis in [19] focuses only on $G(u) = u^N$ and $P_i = 0$ for $i = \lfloor N / 2 \rfloor, ..., N - 1$. In general, $G(u) \cdot H(u) = \mu(u) \cdot P(u) + R(u)$, $\deg(R(u)) < N$. Hence, for a given $\mu(u)$ and arbitrary $R(u)$, $\deg(R(u)) < N$, numerous possibilities exist for $G(u)$ and $H(u)$. We can choose $G(u)$ and $H(u)$ to simplify computations in steps 2 and 4 as shown in section IV. Next section establishes RPS based BA-P. The results are general and valid for GF(p^a), p a prime, and rational, real, and complex numbers.

IV. RPS Based Barrett Algorithm for Polynomials

We turn to a RPS based BA-P for computation in Steps 1-5. All polynomials need to be expressed as residues in RPS defined mod $M(u)$. Step 0 involves a one-time computation, so it is easy to map to RPS. Steps 1, 3, and 5 involve ordinary polynomial multiplication. Hence they are also straightforward to map to RPS. Steps 2 and 4 require computation of quotient up on division by $G(u)$ and $H(u)$, respectively. As seen in Appendix, in order to compute these two steps in residue arithmetic, both $G(u)$ and $H(u)$ must be a factor of $M(u)$. Also, since BA-P is to be used recursively for carrying out MPE, we use the same RPS in all Steps 1-5. In addition, for residues to correspond to the actual polynomials in Steps 1-5, $\deg(M(u))$ must exceed the maximum value of degree of polynomial at each step.

Given $P(u)$, $G(u)$, and $H(u)$ that satisfy Theorem 1, it is easy to calculate the smallest degree of $M(u)$ that is larger than the largest degree polynomial in steps 1-5. We note that $\gcd(G(u), H(u)) = 1$ is not required. We propose to compute the quotients as required in steps 2 and 4 using the algorithm described in Appendix. Since this algorithm requires modulo inverses, it works only when the various moduli polynomials are relatively co-prime. Also, the result of $\lfloor X(u) / G(u) \rfloor$ is known in terms of residues for the moduli that constitute $M(u) / G(u)$. Thus, for the quotient residues to be computed in step 2, we require $\gcd(G(u), M(u) / G(u)) = 1$. Similarly, it is also required that $\gcd(H(u), M(u) / H(u)) = 1$ for quotient residues in step 4. Based on this discussion and Theorem 1, we have the following additional conditions for the RPS based on $M(u)$ to be used in BA-P:
1. largest degree of polynomials in Steps 1-5 < $\deg(M(u)) = L$
2. $G(u) \mid M(u)$ 3. $H(u) \mid M(u)$
4. $\gcd(G(u), M(u) / G(u)) = 1$ 5. $\gcd(H(u), M(u) / H(u)) = 1$.

A number of possibilities become apparent. We can select $G(u)$ and $H(u)$ first that satisfy the above conditions. Then $M(u)$ is constructed such that $\mathrm{lcm}(G(u), H(u)) \mid M(u)$. Finally, if needed further residues are included in $M(u)$ to satisfy the first condition. Also, it is possible to select $M(u)$ first and then $G(u)$ and $H(u)$ in terms of factors of $M(u)$. For instance, if $\alpha = \beta = N$, then $L > 2 \cdot N - 2$.

This description of RPS based BA-P via quotient residues brings out another aspect. Computations in steps 1, 3 and 5 are performed mod $M(u)$. After Step 2, quotient $\lfloor X(u) / G(u) \rfloor$ is available in residues for $M(u) / G(u)$. Hence, we need BEX-P to expand quotient residues back to mod $M(u)$. Similarly, we need BEX-P to expand quotient residues of $\lfloor E(u) / H(u) \rfloor$ computed mod $M(u) / H(u)$ to mod $M(u)$. Such a BEX-P algorithm is described in Section II.

A RPS based new Barrett Algorithm for Polynomials (BA-MPM)
Given: $M(u)$, $G(u)$, $H(u)$. Let $M(u)$ have n factors.
In step 2a, first a factors of $M(u)$ give $G(u)$; and in step 4a, first b factors of $M(u)$ give $H(u)$. There is no loss in generality.
Input: Residues of $A(u)$ and $B(u)$, $(A_i(u), B_i(u) \equiv (A(u), B(u))$ (mod $M_i(u)$), $i = 1, ..., n$.
Pre-computational Step:
Step 0. Compute $\mu_i(u)$, $i = 1, ..., n$, $\mu(u) = \lfloor G(u) \cdot H(u) / P(u) \rfloor$.
Computational Steps:
Step 1. Modulo mult. $X_i(u) \equiv A_i(u) \cdot B_i(u)$, $i = 1, \ldots, n$.
Step 2a. Quotient. Quotient residues $D_i(u)$, $i = a + 1, \ldots, n$, from residues $X_i(u)$, $i = 1, ..., n$, and moduli $G_i(u)$, $i = 1, ..., a$.
Step 2b. BEX-P. Use BEX-P on $D_i(u)$, $i = a + 1, ..., n$, to get a residues $D_i(u)$, $i = 1, ..., a$.
Step 3. Modulo mult. $E_i(u) \equiv D_i(u) \cdot \mu_i(u)$, $i = 1, ..., n$.
Step 4a. Quotient. Quotient residues $Q_i(u)$, $i = b + 1, ..., n$, from residues $E_i(u)$, $i = 1, ..., n$, and moduli $H_i(u)$, $i = 1, ..., b$.
Step 4b. BEX-P. Use BEX-P on $Q_i(u)$, $i = b + 1, ..., n$, to get b residues $Q_i(u)$, $i = 1, ..., b$.
Step 5. Remainder. $C_i(u) \equiv X_i(u) - Q_i(u) \cdot P_i(u)$, $i = 1, \ldots, n$.
Provided that $L - \beta > N - 1$ (a rather trivial condition at this stage), Steps 4b and 5 may also be swapped. In that case, we have:
Step 5. Remainder. $C_i(u) \equiv X_i(u) - Q_i(u) \cdot P_i(u)$, $i = b + 1, \ldots, n$.
Step 6. BEX-P. Use BEX-P on residues $C_i(u)$, $i = b + 1, ..., n$, to get b residues $C_i(u)$, $i = 1, ..., b$.

Example 3. Let the computation be defined in GF(2). Let $N = 2^P - 1$, and $\alpha = \beta = N$. In this case, $\deg(M(u)) > 2 \cdot N - 2$. We choose $M(u) = (u^N - 1) \cdot [(u^{N+2} - 1) / (u - 1)]$ with $G(u) = H(u) = u^N - 1$. Also, $\gcd(u^a - 1, u^b - 1) = u^{\gcd(a,\ b)} - 1$. In our case, N and $N + 2$ are two consecutive odd integers, hence $\gcd(N, N + 2) = 1$. Thus, $u^N - 1$ and $[(u^{N+2} - 1) / (u - 1)]$ are co-prime. The factorization of $u^N - 1$ in GF(2) using GF(2^P) is well-known with each factor having degree P or less. Factorization of $u^{N+2} - 1$ can be obtained from the factorization of $u^A - 1$, $A = 2^{2 \cdot P} - 1$ as $2^{2 \cdot P} - 1 = (2^P - 1) \cdot (2^P + 1)$. Consequently, the factorization of $u^{N+2} - 1$ in GF(2) using GF($2^{2 \cdot P}$) is obtained with factors having degree $2 \cdot P$ or less. The computation $A(u) \cdot B(u)$ mod $P(u)$ with $N = 2^P - 1$ is performed using arithmetic where half the moduli have degree P or less and the other half have degree $2 \cdot P$ or less. These techniques can be extended for N such that $N \mid (2^P - 1)$. These computations correspond to circular convolutions.
Example 4. We pursue previous example. Let $P = 10$, and $N = \alpha = \beta = 1{,}023$. Thus, N takes a large value. In this case, $u^{1{,}023} - 1$ has factors

of degree 10 or less. Similarly, $u^{1,025} - 1$ has factors of degree 20 or less. Let $M(u) = (u^{1,023} - 1) \cdot [(u^{1,025} - 1) / (u - 1)]$ with $G(u) = H(u) = u^{1,023} - 1$. Here, we assume that $\deg(A(u)) = \deg(B(u)) = N - 1 = 1{,}022$. If $A(u)$ and $B(u)$ have degree less than 1,022, then we pad them with 0's and treat them as polynomials of degree 1,022. Thus, $A(u) \cdot B(u) \bmod P(u)$, $N = 1{,}023$, is computed using arithmetic where half the moduli have degree up to 10 and half have degree up to 20.

Example 5. Consider $A(u) \cdot B(u) \bmod P(u)$ in the field of real or complex numbers. We let $G(u) = H(u) = u^N - 1$ and $M(u) = u^{2\cdot N} - 1 = (u^N - 1) \cdot (u^N + 1)$. Let ω be the $2 \cdot N$ root of unity. In this case, computations use DFT and IDFT via FFT. For instance, $X(u)$ in step 1 can be computed using a size $2 \cdot N$ DFT. In step 2a, let $R(u)$ denote the remainder $X(u) \bmod G(u)$. Then $R(u)$ is computed as a size N IDFT of the even DFT coefficients of $X(u)$. We write

$$D(u) = [X(u) - R(u)] / (u^N - 1).$$

Substituting odd powers of ω, we get,

$$D(\omega^{2\cdot k+1}) = -0.5 \cdot [X(\omega^{2\cdot k+1}) - R(\omega^{2\cdot k+1})],\ k = 0, ..., N - 1.$$

Computation of $R(\omega^{2\cdot k+1})$ is same as a size N DFT of sequence $R_i \cdot \omega^i$, $i = 0, ..., N - 1$. $D(\omega^{2\cdot k+1})$ is same as a size N DFT of sequence $D_i \cdot \omega^i$, $i = 0, ..., N - 1$. The computation of $D(u)$ in step 2a is performed by first taking size N IDFT of $D(\omega^{2\cdot k+1})$, $k = 0, ..., N - 1$, obtaining the sequence $D_i \cdot \omega^i$, $i = 0, ..., N - 1$, and then constructing D_i, $i = 0, ..., N - 1$. The BEX-P in step 2b consists in taking size N DFT of $D(u)$. Step 4 is similar to step 2. A total of 2 IDFT and 2 DFT, each of size N, are needed in steps 2 and 4. Steps 0, 1, 3, and 5 are straightforward.

V. Further Analysis

We describe an algorithm for MPE, called BA-MPE, that uses the new RPS based BA-MPM. Let BA-MPM that computes $C(u) = A(u) \cdot B(u) \bmod P(u)$ be denoted by BA-MPM(A, B).

BA-MPE. Here, $\underline{\mathbf{1}}$ denotes vector of all 1s.

Input: Residue vector $\mathbf{A}(\boldsymbol{u})$ for $A(u)$, $P(u)$, and E, $E = \sum_{i=0}^{k} e_i \cdot 2^i$.

Output: Residue vector $\mathbf{C}(\boldsymbol{u})$ for $C(u)$, $C(u) = A(u)^E \bmod P(u)$.

1. If $e_0 = 1$ $\mathbf{C}(\boldsymbol{u}) \leftarrow \mathbf{A}(\boldsymbol{u})$ else $\mathbf{C}(\boldsymbol{u}) \leftarrow \underline{\mathbf{1}}$
2. For $j = 1$ to k do
   - $\mathbf{A}(\boldsymbol{u}) \leftarrow$ BA-MPM(A, A)
   - If $e_j = 1$ then
     - $\mathbf{C}(\boldsymbol{u}) \leftarrow$ BA-MPM(C, A)
   - end If

   end For.

VI. Conclusions

In this work, new Barrett algorithms are described for computing $A(u) \cdot B(u) \bmod P(u)$ and $A(u)^E \bmod P(u)$, $P(u)$ being an irreducible polynomial of degree N. A residue polynomial system based new Barrett algorithm is described that uses only residue arithmetic thus avoiding large degree polynomial multiplication that may be computationally intensive. All the algorithms as described here are a first. The previously known Barrett algorithms use powers of u to scale the various computations.

VII. Acknowledgment

The work of H.K. Garg is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office at the NUS- ZJU Sensor-Enhanced Social Media (SeSaMe) Centre. The work of Hanshen Xiao is supported by Initiative Scientific Research Pro- gram of Tsinghua University under Grant No. 2015THZ0 & 20161080166.

APPENDIX: Computing Quotient Residues in RPS

Problem. Given residues $X_i(u)$ of $X(u)$, $X_i(u) \equiv X(u) \bmod M_i(u)$, $i = 1, \ldots, n$, $M(u) = M_{\mathrm{I}}(u) \cdot M_{\mathrm{II}}(u)$, $M_{\mathrm{I}}(u) = \prod_{i=1}^{a} M_i(u)$, $M_{\mathrm{II}}(u) = \prod_{i=a+1}^{n} M_i(u)$, $\gcd(M_{\mathrm{I}}(u), M_{\mathrm{II}}(u)) = 1$, compute residues of quotient $Q(u)$ when $X(u)$ is divided by $M_{\mathrm{I}}(u)$, $\deg(Q(u)) < \deg(M_{\mathrm{II}}(u))$.

We revisit polynomial arithmetic described in Section II and use it to get an algorithm for computing residues of $Q(u)$. Consider (1) when $X(u)$, $A_1(u)$ and $R_1(u)$ are known. We compute $Q_1(u)$ as

$$Q_1(u) = (X(u) - R_1(u)) \cdot A_1(u)^{-1}. \tag{A1}$$

When $Q_1(u)$, $A_2(u)$ and $R_2(u)$ are known, we compute $Q_2(u)$ as

$$Q_2(u) = (Q_1(u) - R_2(u)) \cdot A_2(u)^{-1}. \tag{A2}$$

This is carried out recursively to finally compute $Q_a(u)$ as

$$Q_a(u) = (Q_{a-1}(u) - R_a(u)) \cdot A_a(u)^{-1}. \tag{A3}$$

The representation of $X(u)$ in (6) is valid. It is reproduced below:

$$X(u) = Q_a(u) \cdot [A_a(u) \cdots A_1(u)] + [R_a(u) \cdot (A_{a-1}(u) \cdots A_1(u)) + \ldots + R_2(u) \cdot A_1(u) + R_1(u)]. \tag{A4}$$

We apply the arithmetic in (A1)-(A4) to RPS defined mod $M(u)$. Given moduli $M_i(u)$ and residues $X_i(u)$, $i = 1, \ldots, n$, we set

$$A_i(u) = M_i(u). \tag{A5}$$

Thus, $R_1(u) = X_1(u)$. This leads to,

$$Q_1(u) = (X(u) - X_1(u)) \cdot M_1^{-1}(u) \tag{A6}$$

Since $X(u)$ is expressed in terms of its residues and $M_1^{-1}(u)$ exists only mod $M_i(u)$, $i = 2, ..., n$, we compute residues of $Q_1(u)$ in (A6) by taking mod $M_i(u)$, $i = 2, ..., n$, of both sides. Thus,

$$Q_{1,i}(u) \equiv (X_i(u) - X_1(u)) \cdot M_1(u)^{-1} \pmod{M_i(u)},\ i = 2, ..., n.$$

As $\deg(Q_1(u)) = \deg(X(u)) - \deg(M_1(u)) < \deg(M(u)) - \deg(M_1(u)) = \sum_{i=2}^{n} \deg\big(M_i(u)\big)$, $Q_1(u)$ is uniquely expressed by its residues $Q_{1,i}(u)$, $i = 2, ..., n$. After the 1st iteration in (A1),

$$R_2(u) \equiv Q_2(u) \pmod{M_2(u)} = Q_{1,2}(u). \tag{A7}$$

Expressing (A7) in residue form, we compute residues of $Q_2(u)$ by taking mod $M_i(u)$, $i = 3, ..., n$, of both sides. This results in

$$Q_{2,i}(u) \equiv (Q_{1,i}(u) - Q_{1,2}(u)) \cdot M_2(u)^{-1} \bmod M_i(u),\ i = 3, ..., n.$$

Again, $\deg(Q_2(u)) < \sum_{i=3}^{n} \deg\big(M_i(u)\big)$. Thus, $Q_2(u)$ is expressed in terms of its residues $Q_{2,i}(u)$, $i = 3, ..., n$.

This process is iterated a times to compute residues of $Q_k(u)$ or $Q_{k,i}(u)$, $i = k + 1, ..., n$, $k = 1, ..., a$. At the end, (A4) becomes

$$X(u) = Q_a(u) \cdot (M_a(u) \cdots M_1(u)) + [R_a(u) \cdot (M_{a-1} \cdots M_1(u)) + \ldots + R_2(u) \cdot M_1(u) + R_1(u)].$$

Thus, $Q_a(u)$ is the quotient obtained by dividing $X(u)$ by $M_{\mathrm{I}}(u)$ expressed in terms of its residues $Q_{a,i}(u)$, $i = a + 1, \ldots, n$.

An algorithm to compute quotient polynomials

Input: RPS defined mod $M(u)$; $M(u) = M_{\mathrm{I}}(u) \cdot M_{\mathrm{II}}(u)$; residues of $X(u)$, $X_i(u) \equiv X(u) \pmod{M_i(u)}$, $i = 1, ..., n$.

Output: Residues of $Q(u) \bmod M_i(u)$, $i = a + 1, \ldots, n$, $Q(u)$ being quotient when $X(u)$ is divided by $M_{\mathrm{I}}(u)$.

Initialization:

$$Q_{0,i}(u) = X_i(u),\ i = 1, \ldots, n.$$

Computational Step: For $k = 1, \ldots, a$,

$$Q_{k,i}(u) \equiv (Q_{k-1,i}(u) - Q_{k-1,k}(u)) \cdot M_k(u)^{-1} \pmod{M_i(u)},\quad i = k + 1, ..., n.$$

Output: $Q_i(u) = Q_{a,i}(u)$, $i = a + 1, ..., n$.

In general, computations in each iteration can be performed in parallel. All the modular inverses can be pre-computed and stored.

References

[1] PL Montgomery, “Modular Multiplication without Trial Division,” Math. of Comp., vol 44, 1985, pp 519-521.

[2] PV Ananda Mohan, Residue Number Systems, Algorithms and Architectures, Springer Int. Series in Engg. & Comp. Sc., 2002.

[3] A Omondi & B Premkumar, Residue Number Systems, Theory & Implementation, Imperial College Press, Advances in Comp. Sc. & Engg., 2007.

[4] J-C Bajard & T Plantard, “RNS Bases and Conversions,” Proc. SPIE, vol 5559, 2004, pp 60-69.

[5] AP Shenoy & R Kumaresan, “Fast Base Extension Using a Redundant Modulus in RNS,” IEEE Trans. on Comp., vol 38, 1989, pp 292-297.

[6] J-C Bajard, L-S Didier & P Kornerup, “Modular Multiplication and Base Extensions in Residue Number Systems,” IEEE Symp. on Comp. Arithmetic, 2001, pp 59-65.

[7] S Kawamura, M Koike, F Sano & A Shimbo, “Cox-Rower Architecture for Fast Parallel Montgomery Multiplication,” EUROCRYPT 2000, Springer Lect. Notes in Comp. Sc., vol 1807, 2000, pp 523-538.

[8] F Gandino, F Lamberti, G Paravati, J-C Bajard & P Montuschi, “An Algorithmic and Architectural Study on Montgomery Exponentiation in RNS,” IEEE Trans. on Comp., vol 61, 2012, pp 1071-1083.

[9] KC Posch & R Posch, “Modulo Reduction in Residue Number Systems,” IEEE Trans. on Parallel & Distributed Systems, vol 6, 1995, pp 449-454.

[10] J Bajard, L-S Didier, & P Kornerup, “An RNS Montgomery Modular Multiplication Algorithm,” IEEE Trans. on Comp., vol 47, 1998, pp 766-776.

[11] H Nozaki, M Motoyama, A Shimbo, & S Kawamura, “Implementation of RSA Algorithm Based on RNS Montgomery Multiplication,” CHES 2001, vol 2162, 2001, pp 364-376.

[12] KA Gbolagade & SD Cotofana, “An O(n) Residue Number System to Mixed Radix Conversion technique,” ISCAS, 2009, pp 521-524.

[13] P Barrett, “Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor,” CRYPTO 1986, Lect. Notes in Comp. Sc., Springer, vol 263, 1987, pp 311–323.

[14] CK Koc & T Acar, “Montgomery Multiplication in GF(2k),” Design, Codes & Cryptography, vol 14, 1998, pp 57-69.

[15] J-F Dhem, “Efficient Modular Reduction Algorithm in Fq[x] and Its Application to Left to Right Modular Multiplication in F2[x],” Proc. Int. Work. on CHES, Lect. Notes in Comp. Sc., Springer-Verlag, 2003, pp 203–213.

[16] M Knezevic, F Vercauteren, & I Verbauwhede, “Speeding up Barrett and Montgomery Modular Multiplications,” http://homes.esat.kuleuven.be/~fvercaut/papers/bar_mont.pdf.

[17] F Angelini & M Bucci, “Reduction Algorithms for Polynomials,” http://accounts.unipg.it/~angelini/welcome_file/redalg.pdf.

[18] ED Win, A Bosselaers, S Vandenberghe, PD Gersem, & J Vandewalle, “A Fast Software Implementation for Arithmetic Operations in GF(2n),” ASIACRYPT’96, vol 1163, Lect. Notes in Comp. Sc., Springer, 2005, pp 65-76.

[19] M Knezevic, K Sakiyama, J Fan, & I Verbauwhede, “Modular Reduction in GF(2n) Without Pre-computation Phase,” Chapter, Arithmetic of Finite Fields, vol 5130, Lect. Notes in Comp. Sc., Springer, 2008, pp 77-87.

[20] J Bajard, L Imbert, & T Plantard, “Arithmetic Operations in the Polynomial Modular Number Systems,” IEEE Symp. on Comp. Arithmetic, 2005, pp 206-213.

[21] T Acar & D Shumow, “Modular Reduction without Pre-Computation for Special Moduli,” http://research.microsoft.com/pubs/120244/modmul_no_precomp.pdf.

[22] D Schinianakis, A Skavantzos, & T Stouraitis, “GF(2n) Montgomery Multiplication using Polynomial Residue Arithmetic,” ISCAS, 2012, pp 3033-3036.

[23] C Ding, D Pei, & A Salomaa, Chinese Remainder Theorem, Applications in Computing, Coding, and Cryptography, World Scientific, 1996.

[24] NS Szabo & RI Tanaka, Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill, NY, 1967.

[25] A Menezes, P van Oorschot, & S Vanstone, Handbook of Applied Cryptography, CRC Press, USA, 1996.

[26] LC Washington, Elliptic Curves: Number Theory & Cryptography, CRC Press, USA, 2008.

[27] D Hankerson, A Menezes & S Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag NY, 2004.

[28] Z Cao, R Wei, & X Lin, “A Fast Modular Reduction Method,” IACR Cryptology ePrint Archive, 2014.

[29] H Krishna, B Krishna, K-Y Lin & J-D Sun, Computational Number Theory and Digital Signal Processing: Fast Algorithms and Error Control Techniques, CRC Press, USA, 1994.