SAN097-2732C
SAND--97-2732C
CONF-980536--

Modular Redundant Number Systems

With the increased use of public key cryptography, faster modular multiplication has become an important cryptographic issue. Almost all public key cryptography, including most elliptic curve systems, use modular multiplication. Modular multiplication, particularly for the large public key modulii, is very slow. Increasing the speed of modular multiplication is almost synonymous with increasing the speed of public key cryptography.

There are two parts to modular multiplication: multiplication and modular reduction. Though there are fast methods for multiplying and fast methods for doing modular reduction, they do not mix well. Most fast techniques require integers to be in a special form. These special forms are not related and converting from one form to another is more costly than using the standard techniques. To this date it has been better to use the fast modular reduction technique coupled with standard multiplication. Standard modular reduction is much more costly than standard multiplication. Fast modular reduction (Montgomery's method) reduces the reduction cost to approximately that of a standard multiply.

Of the fast multiplication techniques, the redundant number system technique (RNS) is one of the most popular. It is simple, converting a large convolution (multiply) into many smaller independent ones. Not only do redundant number systems increase speed, but the independent parts allow for parallelization. As stated earlier, the main drawback is the costly conversion. The cost of converting the integers between every multiply and modular reduction far outweighs the savings. If RNS is to be used at all for modular multiplication, integers must remain in RNS form.

Unfortunately Montgomery's method does not seem possible in RNS form. The reduction formula includes both reduction and division by a given constant. RNS form implies working modulo another constant. Depending on the relationship between these two constants; reduction OR division may be possible, but not both. This paper describes a new technique using ideas from both Montgomery's method and RNS. It avoids the formula problem and allows fast reduction and multiplication. Since RNS form is used throughout, it also allows the entire process to be parallelized.

Redundant Number Systems and Fast Reduction Formula

The following is a very brief explanation of RNS and the fast reduction formula, the basis for Montgomery reduction.

RNS: Redundant number systems are based on the Chinese remainder theorem. In this paper a redundant number system (RNS) refers to the modulii in the system: $\{q_1, q_2, \ldots, q_u\}$ with GCD(q_i,q_j)=1 for all i≠j. Working in the RNS implies working mod Q where Q is the product of all the components in the RNS: $Q = \prod_{i=1}^{u} q_i$. An integer X in RNS form is written as the following. Let $x_i \equiv X \bmod q_i$ $(0 \le x_i < q_i)$. Then:

$$X \sim [x_1, x_2, \cdots, x_u].$$

Operations in the RNS are done component-wise, independent of the other components. For example, to compute XY+Z, simply compute $x_i y_i + z_i \bmod q_i$ for each of the q_i in the RNS. Operations done in the RNS mirror operations done mod Q, so any operations done mod Q can be done in RNS form (in the example above XY+Z is computed mod Q). Notice that if the results, intermediate as well as final, are greater than zero and less than Q, the RNS results will equal the integer results.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

There are many techniques to reconstruct the RNS number. A new method, the RNS interweaving process, is described in this paper. It differs from other techniques in that $X \bmod N$ can be computed without ever leaving the RNS/mod N world.

Fast Reduction Formula: This formula reduces integers Z modulo M, where $Z<4M^2$. A value R (the reduction R-value) for which division and modular reduction are easy is chosen. 'Easy' reduction and division replace the 'hard' reduction (mod M). The only restrictions on the R-value is that it must be relatively prime to M and R>4M. Letting $\overline{M} \equiv -M^{-1} \bmod R$, the reduction formula (RF) acting on an integer $Z<4M^2$ is:

$$RF(Z) = \frac{Z + M(Z\overline{M} \bmod R)}{R}$$

The results of this formula have two important properties when compared to M:

$$RF(Z) \equiv ZR^{-1} \bmod M$$
$$< 2M$$

Notice that a copy of R^{-1} was added to the reduced results. To compensate for this, integers are usually stored with a copy of R. In other words, instead of storing Z mod M, ZR mod M is stored. This way when two numbers are multiplied and reduced, one copy of R remains, leaving it in the same form:

$$RF(XR*YR) \equiv XYR \bmod M.$$

Problem

A problem occurs when trying to combine these two techniques. Working in an RNS implies working mod Q. The reduction formula mod Q looks like:

$$\left(Z + M\left(Z\overline{M} \bmod R\right)\right)R^{-1} \bmod Q$$

This formula contains both a reduction by R and an inverse of R. If GCD(R,Q)=1 the inverse exists but reduction by R is costly. If GCD(R,Q)≠1 then inverse of R does not exist. This seems to imply that this formula is not possible in an RNS.

Fortunately there is a way around the problem: Use a variable redundant number system. In a variable RNS the components change as the algorithm progresses. Initially all of R is in the RNS, making the modular reduction portion of the formula fast. R is gradually removed from the system, making the inverse of R exist.

This solution introduces another problem. Some of the components of the redundant number system are lost as the algorithm progresses. The resulting RNS is large enough to hold the reduced integer but too small to do another modular multiply. So before the next modular multiplication, lost parts of the RNS must be rebuilt. In other words: Given Z in an RNS, find Z mod N where N is some other integer not necessarily in the RNS. As mentioned earlier this is costly. The conventional way to do this is to compute Z from the RNS, then reduce it mod N.

Fortunately there is also a better way do this. The following technique avoids the conversion to a full integer. Instead, it interweaves the independent values of the RNS together to generate dependent values modulo the RNS components. All operations in the interweaving process are modulo the RNS components. These values are then used in a mod N formula to compute the integer modulo N. This process can also be used to implement standard Montgomery multiplication/reduction in a RNS by interweaving the results and computing Z modulo the power of two in R. However the interweaving process would have to be done at each step of the algorithm. This is probably too many times to make it an efficient modular multiplication algorithm. As it is used at between modular multiplies, the RNS-interweaving process will be described first.

DISCLAIMER

RNS-Interweaving Process

The RNS-interweaving process extracts partial data from the RNS. Given an integer X in an RNS, the interweaving process computes X mod N, where N may not be a part of the RNS. The process avoids any multiple precision arithmetic (assuming the RNS and N are single precision), working only modulo the components of the RNS and N. Instead of computing the whole number, the integer is woven together inside the RNS. The actual integer is computed from a formula on the woven RNS values. This formula is completely in the ring of integers and equals the integer reduced mod Q (the product of the RNS modulii). There is no need to compute the full integer, although the full integer can be easily computed from the woven RNS values.

Concept

The interweaving process starts with a very simple observation. Let $\{q_1,q_2\}$ be a RNS with $Q=q_1q_2$ and an integer X<Q represented in Q: $X\sim[x_1,x_2]$. If $X=x_1+X_2q_1$, then $X_2=\left\lfloor\frac{X}{q_1}\right\rfloor<q_2$.

Using the remaining congruence, X_2 can be solved for:

$$x_1+X_2q_1\equiv x_2 \bmod q_2$$
$$X_2\equiv(x_2-x_1)q^{-1} \bmod q_2$$

Since $X_2<q_2$ this formula determines X_2 uniquely. Now that X_2 is known, X can be written in terms of results mod q_1 and q_2: $X=x_1+q_1\left((x_2-x_1)q_1^{-1} \bmod q_2\right)$. The RNS-interweaving process extends this idea out for more than two modulii.

Extension

The first step in the RNS-interweaving process is to break down the integer into dependent components. The full integer, or the integer modulo another integer, is constructed from these parts. The full integer ($0\le(X=X_1)<Q$) can be reconstructed from one element of the RNS (say x_1) and the quotient $X_2=\left\lfloor\frac{X_1}{q_1}\right\rfloor$: $X_1=x_1+q_1X_2$. Since X_1 is less than Q, X_2 must be less than Q/q_1. So X_2 can be uniquely represented in the RNS with q_1 removed. Likewise, X_2 can be reconstructed with one element of its representation (say x_2) and the quotient $X_3=\left\lfloor\frac{X_2}{q_2}\right\rfloor$. Using the same arguments, X_3 must be less than Q/q_1q_2, so it can be represented uniquely in the RNS with q_1 and q_2 removed. If there are u-components in the RNS, the final quotient $X_u=\left\lfloor\frac{X_{u-1}}{q_{u-1}}\right\rfloor$ will be less than q_u. The following lemma shows that these quotient values can be represented by smaller and smaller RNS and shows what the representation of the quotient values are.

Lemma: Let Q_i be an RNS with $\{q_i,q_{i+1},\ldots,q_u\}$ and X_i be an integer represented in Q_i:

$X_i\sim[x_{i,i},x_{i,i+1},\cdots,x_{i,u}]$. Then $X_i=x_{i,i}+X_{i+1}q_i$ implies that $X_{i+1}<Q_{i+1}$ and $X_{i+1}\equiv(x_{i,j}-x_{i,i})q_i^{-1} \bmod q_j$ for j=i+1,i+2,...,u. This implies that X_{i+1} is uniquely represented in the RNS Q_{i+1} by $X_{i+1}\sim[x_{i+1,i+1},x_{i+1,i+2},\cdots,x_{i+1,u}]$ where $x_{i+1,j}\equiv(x_{i,j}-x_{i,i})q_i^{-1} \bmod q_j$.

Proof: (1) X_i represented in Q_i implies that $X_i<Q_i$. So

$$x_{i,i} + X_{i+1} q_i < Q_i$$
$$X_{i+1} < \frac{Q_i}{q_i} .$$
$$X_{i+1} < Q_{i+1}$$

(2) $X_{i+1}<Q_{i+1}$ implies that X_{i+1} can be uniquely represented in the Q_{i+1} RNS. Since $x_{i,i} + X_{i+1} q_i = X_i \equiv x_{i,j} \bmod q_j$ for j=i+1, i+2,...,u:

$$x_{i,i} + X_{i+1} q_i \equiv x_{i,j} \bmod q_j$$
$$X_{i+1} q_i \equiv x_{i,j} - x_{i,i} \bmod q_j \quad .$$
$$X_{i+1} \equiv \left(x_{i,j} - x_{i,i}\right) q_i^{-1} \bmod q_j$$

So $X_{i+1} \sim \left[x_{i+1,i+1}, x_{i+1,i+2}, \cdots, x_{i+1,u}\right]$ with $x_{i+1,j} \equiv \left(x_{i,j} - x_{i,i}\right) q_i^{-1} \bmod q_j$.

Algorithm

The reconstruction formula uses the $x_{i,i}$-values only. The other values in the process are only needed to go to the next step and can be overwritten. The algorithm is as follows:

1. Begin with $X_1 \sim \left[x_{1,1}, x_{1,2}, \cdots, x_{1,u}\right]$ in the RNS $Q_1 \sim \{q_1, q_2, \ldots, q_u\}$.
2. For i=2,3,...,u compute X_i in it's RNS form:
   a. For j=i+1,i+2,...,u compute $x_{i,j}$:
      $x_{i,j} \equiv \left(x_{i-1,j} - x_{i-1,i-1}\right) q_{i-1}^{-1} \bmod q_j$
3. RESULTS: $x_{i,i}$ for i=1,2,...,u

Formula

Since $X_j = x_{j,j} + q_j X_{j+1}$, the process can be backed up, giving the following equation for X:

$$X = x_{1,1} + q_1(x_{2,2} + q_2(\ldots q_{u-2}(x_{u-1,u-1} + q_{u-1}(x_{u,u}))\ldots))$$

Notice that the formula above is a simple equality (no modular reductions) with only integer addition and multiplication as operands. This means that finding X mod N does not require anything but mod N and mod q_i operations. If N and $\{q_i\}$ are all single precision then computing X mod N can be done in single precision, even if X is a multiple precision integer.

Example: Q~{1999, 107, 71,31}; X~[306,86,13,22]. Compute X mod 2, 5, and 97

INVERSES: 1999^{-1}: 22 mod 107, 13 mod 71, 29 mod 31
107^{-1}: 2 mod 71, 20 mod 31
71^{-1}: 7 mod 31.

	mod 1999	mod 107	mod 71	mod 31
X_1	306={92,22,27}	86	13	22
X_2		(86-92)22≡82	(13-22)13≡25	(22-27)29≡10
X_3			(25-11)2≡28	(10-20)20≡17
X_4				(17-28)7≡16

X=306+1999(82+107(28+71(16)))
X≡0+1(0+1(0+1(0)))≡0 mod 2

$X \equiv 1+4(2+2(3+1(1))) \equiv 1 \bmod 5$
$X \equiv 15+59(82+10(28+71(16))) \equiv 3 \bmod 97$

M-RNS

Basic Idea:

This technique solves the problem of RNS/reduction formula incompatibility by going around it. Make the RNS components vary as the algorithm progresses. This way the reduction R-value can start out as part of the RNS (making reduction by R easy) and later be removed (so inverses of R exist). Overflow will not occur as the size of the number shrinks along with the RNS as the algorithm progresses.

Before describing the technique, the reduction formula should be recalled in its modular form. Let Z be the integer being reduced, M the modulus, R the reduction formula value, $\overline{M} = -M^{-1} \bmod R$ and Q the remaining portion of the RNS:

$$\left(Z + M\left(Z\overline{M} \bmod R\right)\right)R^{-1} \bmod Q .$$

This modular redundant number system consists of a set of relatively prime integers, some in R, the rest in Q. As the algorithm progresses components in R will gradually be lost. In the following (over) simplified version, R and Q consist of just one component. Let X and Y be the two integers to be multiplied. The algorithm does modular multiplication in the following three steps:

1. Multiply in the RNS: Z=[XY mod R, XY mod Q].
2. Compute $T = Z\overline{M} \bmod R$.
3. Finish the reduction formula: $(Z+MT)R^{-1} \bmod Q$ (note that the results will be congruent to $ZR^{-1} \bmod M$).

As mentioned before, R and Q consist of many relatively prime components. Step (1) is done first for each element the RNS. Steps (2) and (3) are repeated for each component of R. If $R = \prod_{i=1}^{u} q_i$ and $Q = \prod_{i=u+1}^{u+v} q_i$, then at iteration i the reduction in step (2) is done modulo q_i:

$$t_i \equiv z_i \overline{M} \bmod q_i$$

With this part known, the rest of the formula can be computed in all the remaining components except q_i. So for j=i+1,i+2,...,u,u+1,...,u+v:

$$z_j \equiv \left(z_j + M_j t_i\right) q_i^{-1} \bmod q_j$$

The resulting number (if converted back to a standard integer at this point) is equal to:

$$\frac{Z + M\left(Z\overline{M} \bmod \prod_{j=1}^{i} q_j\right)}{\prod_{j=1}^{i} q_j}$$

If X,Y<2M, R>4M and Q>2M, the result will be less than $\prod_{j=i+1}^{u+v} q_j$. In words this means that the formula can be computed iteratively without overflow. These results will be proved after the algorithm is explained.

Components:

M: The modulus

R: $R \sim \{q_1, q_2, \cdots, q_u\}$ and $R = \prod_{i=1}^{u} q_i$, the product of relatively prime modulii, all relatively prime to M and greater than 4M (this bound allows input of integers up to $4M^2$ in size to be reduced to less than 2M).

Q: $Q \sim \{q_{u+1}, q_{u+2}, \cdots, q_{u+v}\}$ and $Q = \prod_{i=u+1}^{u+v} q_i$, another product of relatively prime modulii, also relatively prime to R and greater than 2M (this bound allows the RNS consisting only of Q to store the end result).

$\overline{M}$: $\overline{M} \equiv -\left(M^{-1}\right) \bmod R$, found with the GCD algorithm;

RNS form:

$\overline{M} \sim [\overline{m}_1, \overline{m}_2, \cdots, \overline{m}_u]$ (since these values are only needed modulo the R components);

$M \sim [m_1, m_2, \cdots, m_u, m_{u+1}, \cdots, m_{u+v}]$

RNS form and multiplied by R mod M ($0 \le X, Y < 2M$):

$X \sim [x_1, x_2, \cdots, x_u, x_{u+1}, \cdots, x_{u+v}]$

$Y \sim [y_1, y_2, \cdots, y_u, y_{u+1}, \cdots, y_{u+v}]$

Algorithm

Multiply in the full RNS: $XY = Z_0 \sim \left[z_{0,i} \equiv x_i y_i \bmod q_i\right]_{i=1,2,\ldots,u+v}$.

For i=1,2,...,u

Compute $t_i = z_{i-1,i}\overline{M} \bmod q_i$

For j=i+1, i+2, ..., u+v, compute $z_{i,j} = q_i^{-1}\left(z_{i-1,j} + Mt_i\right) \bmod q_j$

At the end of step i, the RNS consists of $\{q_{i+1}, q_{i+2}, \cdots, q_u, q_{u+1}, \cdots, q_{u+v}\}$. At step u,

$Z_u \equiv ZR^{-1} \bmod M$, and $Z_u < 2M$. The final results are only represented by the Q portion of the RNS. Full representation is needed to multiply again. See the 'RNS-Interweaving Process' earlier in this paper.

Inequalities and Size

In the following discussion on inequalities and size, the reconstructed integer Z (uniquely determined by the Chinese remainder theorem) is used instead of the RNS version of Z. This greatly simplifies the proofs.

Size

The intermediate and final results, Z_i, are computed in two steps:

1. Compute $t_i = z_{i-1,i}\overline{M} \bmod q_i$.
2. Compute $Z_i = \left(Z_{i-1} + Mt_i\right)q_i^{-1}$ in the remaining RNS.

There are no size concerns in the first step, but there are two overflow concerns in the second step. First, does Z_i overflow and second, does $(Z_{i-1}+Mt_i)$ overflow. To insure that Z_i does not overflow, Z_i must be less than the remaining RNS:

$$Z_i < Q\prod_{j=i+1}^{u} q_j .$$

There is a little more leeway in the size of $(Z_{i-1}+Mt_i)$. Even though the value modulo qi is not actually computed, it is predetermined to be zero. Thus $(Z_{i-1}+Mt_i)$ will not overflow as long as the it is less than the preceding RNS:

$$Z_{i-1} + Mt_i < Q\prod_{j=i}^{u} q_j .$$

The following lemmas show that intermediate values will not overflow, and that the final value is less than 2M.

Lemma: In the algorithm above, $Z_i < M\left(1+\prod_{j=i+1}^{u} q_j\right) < Q\prod_{j=i+1}^{u} q_j$, and $Z_u < 2M$.

Proof: First we know that

$$M\left(1+\prod_{j=i+1}^{u} q_j\right) < \frac{Q}{2}\left(1+\prod_{j=i+1}^{u} q_j\right) = Q\prod_{j=i+1}^{u} q_j - \frac{Q}{2}\left(\prod_{j=i+1}^{u} q_j - 1\right)$$

$$< Q\prod_{j=i+1}^{u} q_j$$

So all that's left is to show is $Z_i < M\left(1+\prod_{j=i+1}^{u} q_j\right)$.

By induction.

1. $Z_0 = XY < 4M^2 < MR < M\left(1+\prod_{j=1}^{u} q_j\right)$.

2. Assume this holds for i=1,2,...,k-1. Then

$$Z_k = q_k^{-1}\left(Z_{k-1} + M\left(Z_{k-1}\overline{m} \bmod q_k\right)\right) < q_k^{-1}\left(M\left(\prod_{j=k}^{u} q_j + 1\right) + M\left(q_k - 1\right)\right) = M\left(\prod_{j=k+1}^{u} q_j + 1\right)$$

So $Z_k < M\left(\prod_{j=k+1}^{u} q_j + 1\right)$.

3. Therefore $Z_i < M\left(\prod_{j=i+1}^{u} q_j + 1\right)$ holds for i=1,2,...,u. In particular, for i=u,

$Z_u < M(1+1) = 2M$.

Lemma: Intermediate values, $Z_{i-1} + Mt_i < Q\prod_{j=i}^{u} q_j$ for i=1,2,...,u

Proof: From above we know that $Z_{i-1} < M\left(1+\prod_{j=i}^{u} q_j\right)$. So

$$Z_{i-1} + Mt_i < M\left(1 + \prod_{j=i}^{u} q_j\right) + M(q_i - 1) = M\left(q_i + \prod_{j=i}^{u} q_j\right)$$
$$< \tfrac{1}{2}Q\left(q_i + \prod_{j=i}^{u} q_j\right) = Q\prod_{j=i}^{u} q_j - \tfrac{1}{2}Qq_i\left(\prod_{j=i+1}^{u} q_j - 1\right)$$
$$< Q\prod_{j=i}^{u} q_j$$

CONGRUENCE

The following lemma shows that the final results satisfy: $Z_u \equiv ZR^{-1} \bmod M$.

Lemma: $Z_i \equiv Z\left(\prod_{j=1}^{i} q_j\right)^{-1} \bmod M$, and $Z_u \equiv ZR^{-1} \bmod M$.

Proof: By induction.

1. $Z_0 = Z \equiv Z\left(\prod_{j=1}^{0} q_j\right)^{-1} \bmod M$.

2. Assume this holds for i=1,2,...,k-1. Then

$$Z_k = q_k^{-1}\left(Z_{k-1} + M\left(Z_{k-1}\overline{m} \bmod q_k\right)\right)$$
$$= q_k^{-1}\left(Z_{k-1} + M\overline{M}Z_{k-1} - wMq_k\right)$$
$$= q_k^{-1}\left(Z_{k-1} + \left(q_k q_k^{-1} - 1\right)Z_{k-1} - wMq_k\right)$$
$$\equiv \left(Z\left(\prod_{j=1}^{k-1} q_j\right)^{-1}\right) q_k^{-1} \bmod M$$
$$\equiv Z\left(\prod_{j=1}^{k} q_j\right)^{-1} \bmod M$$

3. Therefore this holds for i=1,2,...,u. And for i=u: $Z_u \equiv Z\left(\prod_{j=1}^{u} q_j\right)^{-1} \equiv ZR^{-1} \bmod M$.

Choice of the RNS Components:

One of the main concerns for efficiency is choice of the RNS modulii. Fast modular reduction in the system and simplification of the operations in the reduction formula are the major concerns. Keeping the modulii single precision is probably a good idea. The largest size modulus for a 32-bit processor is 32,400 bits (use prime powers less than or equal to 2^{16}), which is large enough for most current uses.

Another choice is modulii of the form 2^n-1 are good choices. Modular reduction simplifies to mask, shift and add (since $2^n \equiv 1 \bmod (2^n - 1)$) and components can be chosen such that the inverses of subsequent elements in the RNS have very low density. This simplifies multiplication by these inverses to a few number of shifts and adds.

Low Density Inverses for 2^m-1 mod 2^n-1

In an RNS with modulii of the form 2^n-1, low density inverses for subsequent RNS components should improve the performance of the algorithm. If m′ is the inverse of m mod n, then the inverse of 2^m-1 mod 2^n-1 is (see [1]):

$$\left(2^m-1\right)^{-1} \equiv \sum_{k=0}^{m'-1} 2^{km} \bmod\left(2^n-1\right).$$

So a $(2^m-1)^{-1}$ mod 2^n-1 has a density of m′. Choosing m-values for which m′ modulo the previously chosen n-values gives inverses with low density.

Let $\{q_1, q_2, \cdots, q_{u+v}\}$ be the RNS, with $q_i = 2^{n_i} - 1$. For efficiency the inverse of q_j mod q_i should have low density for all j<i. If $m_{i,j} \equiv n_j^{-1} \bmod n_i$ then the inverse for q_j mod q_i is given by the following (see reference [1]):

$$q_j^{-1} \equiv \sum_{k=0}^{m_{i,j}-1} 2^{kn_j} \bmod q_i .$$

Finding an efficient RNS set reduces to: find a set of pair-wise relatively prime integers $\{n_1, n_2, \cdots, n_{u+v}\}$ such that the inverse of n_j mod n_i (for all j<i) is less than a small bound. Care should be taken in choosing this bound. Too small a bound will restrict the choices for the RNS too greatly while too large a bound will slow down the multiplication by inverses.

Report Number (14) SAND--97-2732C
CONF-980536--

Publ. Date (11) 19980531
Sponsor Code (18) DOE/DP, XF
UC Category (19) UC-700, DOE/ER

DOE