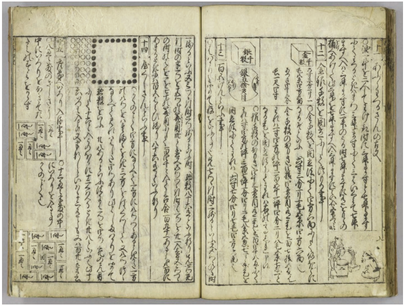
Chinese Remainder Theorem

Çetin Kaya Koç

koc@cs.ucsb.edu



The Chinese Remainder Theorem

- Some cryptographic algorithms work with two (such as RSA) or more moduli (such as secret-sharing)
- The Chinese Remainder Theorem (CRT) and underlying algorithm allows to work with multiple moduli
- The general idea is to compute a large integer X knowing only its remainders modulo a small set of integers (called moduli)
- The principles of this method was established sometime in the 3rd and 5th century in China
- A Chinese mathematician Sun Tzu or Sunzi is known to be the author of *The Mathematical Classic of Sunzi*, which contains the earliest known example of the algorithm
- Thus, it is named as the Chinese Remainder Theorem.

The Chinese Remainder Theorem

Theorem

Given k pairwise relatively prime moduli n_i for $i = 1, 2, \ldots, k$, a number $x \in [0, m-1]$ with $m = n_1 \cdot n_2 \cdots n_k$ is uniquely representable using the remainders r_i for $i = 1, 2, \ldots, k$ such that $r_i = x \pmod{n_i}$.

Given the remainders $r_1, r_2, \ldots, r_k$, we can compute x using

$$x = \sum_{i=1}^{k} r_i \cdot c_i \cdot m_i \pmod{m}$$

where $m_i = m/n_i$ and $c_i = m_i^{-1} \pmod{n_i}$

- The computation of x using the linear summation formula above is also called the Chinese Remainder Algorithm (CRA)

A CRT Example

- Let the moduli set be $\{5, 7, 9\}$
- These moduli are pairwise relatively prime: $\gcd(5, 7) = \gcd(5, 9) = \gcd(7, 9) = 1$
- Each modulus does not need to be prime, but they need to be pairwise relatively prime
- If they are all prime, they will be pairwise relatively prime too
- We have $n_1 = 5$, $n_2 = 7$, $n_3 = 9$, and thus $m = 5 \cdot 7 \cdot 9 = 315$
- All integers in the range $[0, 314]$ are uniquely representable using this moduli set

A CRT Example

- Let $x = 200$, then we have

$$\begin{array}{lllllllll} r_1 & = & 200 \bmod 5 & r_2 & = & 200 \bmod 7 & r_3 & = & 200 \bmod 9 \\ r_1 & = & 0 & r_2 & = & 4 & r_3 & = & 2 \end{array}$$

- The remainder set $(0, 4, 2)$ with respect to the moduli set $(5, 7, 9)$ uniquely represents the integer 200
- Given the integer x and the moduli set, the remainders can be computed using $r_i = x \pmod{n_i}$ for $i = 1, 2, \ldots, k$
- Given the remainders and the moduli set, the integer x can be computed using the standard Chinese Remainder Algorithm, given above, represented as

$$\mathrm{CRT}(0, 4, 2; 5, 7, 9) = 200$$

A CRT Example

- Compute $x = \text{CRT}(0,4,2;5,7,9)$

$$m = n_1 \cdot n_2 \cdot n_3 = 5 \cdot 7 \cdot 9 = 315$$

$$m_1 = m/n_1 = 315/5 = 7 \cdot 9 = 63$$
$$m_2 = m/n_2 = 315/7 = 5 \cdot 9 = 45$$
$$m_3 = m/n_3 = 315/9 = 5 \cdot 7 = 35$$

$$c_1 = m_1^{-1} = 63^{-1} = 3^{-1} = 2 \pmod 5$$
$$c_2 = m_2^{-1} = 45^{-1} = 3^{-1} = 5 \pmod 7$$
$$c_2 = m_3^{-1} = 35^{-1} = 8^{-1} = 8 \pmod 9$$

$$\begin{aligned} x &= r_1 \cdot c_1 \cdot m_1 + r_2 \cdot c_2 \cdot m_2 + r_3 \cdot c_3 \cdot m_3 \pmod m \\ &= 0 \cdot 2 \cdot 63 + 4 \cdot 5 \cdot 45 + 2 \cdot 8 \cdot 35 = 1460 \pmod{315} \\ &= 200 \pmod{315} \end{aligned}$$

Therefore, $\text{CRT}(0,4,2;5,7,9) = 200$

Another CRT Example

- Compute $x = \text{CRT}(2,1,1;7,9,11)$

$m = n_1 \cdot n_2 \cdot n_3 = 7 \cdot 9 \cdot 11 = 693$

$m_1 = m/n_1 = 693/7 = 9 \cdot 11 = 99$
$m_2 = m/n_2 = 693/9 = 7 \cdot 11 = 77$
$m_3 = m/n_3 = 693/11 = 7 \cdot 9 = 63$

$c_1 = m_1^{-1} = 99^{-1} = 1^{-1} = 1 \pmod{7}$
$c_2 = m_2^{-1} = 77^{-1} = 5^{-1} = 2 \pmod{9}$
$c_2 = m_3^{-1} = 63^{-1} = 8^{-1} = 7 \pmod{11}$

$$\begin{aligned} x &= r_1 \cdot c_1 \cdot m_1 + r_2 \cdot c_2 \cdot m_2 + r_3 \cdot c_3 \cdot m_3 \pmod{N} \\ &= 2 \cdot 1 \cdot 99 + 1 \cdot 2 \cdot 77 + 1 \cdot 7 \cdot 63 = 793 \pmod{693} \\ &= 100 \pmod{693} \end{aligned}$$

Therefore, $\text{CRT}(2,1,1;7,9,11) = 100$

The Mixed Radix Conversion Algorithm

- The standard CRA uses the summation

$$x = \sum_{i=1}^{k} r_i \cdot c_i \cdot m_i \pmod{m}$$

- The CRA requires multi-precision arithmetic at each step, as each product term in the summation grows beyond $m = n_1 \cdot n_2 \cdots n_k$
- There exists another algorithm, called the Mixed Radix Conversion (MRC) Algorithm, which computes x more efficiently
- The MRC algorithm is particularly useful when each modulus fits into the word size of the computer
- The MRC avoids multi-precision arithmetic until the last phase

The Step 1 of the MRC Algorithm

- Step 1: Compute and save the inverses c_{ij} for $1 \leq i < j \leq k$

$$c_{ij} = n_j^{-1} \pmod{n_i}$$

- This is accomplished using the extended Euclidean algorithm for the Fermat's method if the modulus is prime
- In case each modulus fits into the word size of the computer, any of the inverses would also fit into the same size
- Step 1 can be performed using the single-precision arithmetic

The Step 2 of the MRC Algorithm

- Step 2: Given the remainders $(r_1, r_2, \ldots, r_k)$ of X with respect to the moduli $(n_1, n_2, \ldots, n_k)$ and its first column as

$$r_{i1} = r_i \quad \text{for} \quad i = 1, 2, \ldots, k$$

compute the entries of the lower triangular matrix

$$\begin{array}{lllll} r_{11} & & & & \\ r_{21} & r_{22} & & & \\ r_{31} & r_{32} & r_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ r_{k1} & r_{k2} & r_{k3} & \cdots & r_{kk} \end{array}$$

- The computations in the ith row are performed mod n_i

The Step 2 of the MRC Algorithm

- The 2nd column is computed using the 1st column and the inverses

$$r_{i2} \quad = \quad (r_{i1} - r_{11}) \cdot c_{i1} \mod n_i \quad \text{for} \quad i = 2, 3, \ldots, k$$

- The 3rd column is computed using the 2nd column and the inverses

$$r_{i3} \quad = \quad (r_{i2} - r_{22}) \cdot c_{i2} \mod n_i \quad \text{for} \quad i = 3, 4, \ldots, k$$

- The 4th column is computed using the 3rd column and the inverses

$$r_{i4} \quad = \quad (r_{i3} - r_{33}) \cdot c_{i3} \mod n_i \quad \text{for} \quad i = 4, 5, \ldots, k$$

The Step 2 of the MRC Algorithm

- The jth column is computed using the $(j-1)$th column and the inverses

$$r_{ij} \quad = \quad (r_{i,j-1} - r_{j-1,j-1}) \cdot c_{i,j-1} \mod n_i \quad \text{for} \quad i = j, j+1, \ldots, k$$

- All computations in Step 2 are in single-precision arithmetic
- As an example, for $k = 5$ we compute

$$\begin{array}{llllll}
\mathbf{r_{11}} = r_1 & & & & & \mod n_1 \\
r_{21} = r_2 & \mathbf{r_{22}} = (r_{21} - r_{11})c_{21} & & & & \mod n_2 \\
r_{31} = r_3 & r_{32} = (r_{31} - r_{11})c_{31} & \mathbf{r_{33}} = (r_{32} - r_{22})c_{32} & & & \mod n_3 \\
r_{41} = r_4 & r_{42} = (r_{41} - r_{11})c_{41} & r_{43} = (r_{42} - r_{22})c_{42} & \mathbf{r_{44}} = (r_{43} - r_{33})c_{43} & & \mod n_4 \\
r_{51} = r_5 & r_{52} = (r_{51} - r_{11})c_{41} & r_{53} = (r_{52} - r_{22})c_{52} & r_{54} = (r_{53} - r_{33})c_{53} & \mathbf{r_{55}} = (r_{54} - r_{44})c_{54} & \mod n_5
\end{array}$$

The Step 3 of the MRC Algorithm

- Step 3: The integer x is then computed using the diagonal entries as

$$c = \mathbf{r_{11}} + \mathbf{r_{22}} \cdot n_1 + \mathbf{r_{33}} \cdot n_1 \cdot n_2 + \cdots + \mathbf{r_{kk}} \cdot n_1 \cdot n_2 \cdots n_{k-1}$$

- This step requires multiprecision arithmetic due to the product terms $n_1 \cdot n_2 \cdots n_i$ for $i = 1, 2, \ldots, k-1$ in the summation to obtain x
- Step 3 is the only step in the MRC which requires multiprecision arithmetic
- The MRC has some other advantages: Two numbers can be compared in size if their MRC coefficients $(r_{11}, r_{22}, \ldots, r_{kk})$ are known
- The MRC is essentially a radix representation of the number x, however, more than one radix is used (thus: the term, mixed-radix)

An Example of the MRC Algorithm

- As example, let us take the remainders $(r_1, r_2, r_3) = (2, 1, 1)$ with respect to the moduli $(n_1, n_2, n_3) = (7, 9, 11)$ and compute x
- Step 1: First we compute and save the inverses c_{21}, c_{31}, c_{32}

$$\begin{array}{lllll} c_{21} & = & n_1^{-1} \pmod{n_2} & 7^{-1} \pmod{9} & \rightarrow 4 \\ c_{31} & = & n_1^{-1} \pmod{n_3} & 7^{-1} \pmod{11} & \rightarrow 8 \\ c_{32} & = & n_2^{-1} \pmod{n_3} & 9^{-1} \pmod{11} & \rightarrow 5 \end{array}$$

An Example of the MRC Algorithm

- Step 2: The first column of the lower triangular matrix is the given set of remainders $(2, 1, 1)$ from which we compute the rest of the columns:

$$\begin{array}{llll} \mathbf{2} & & & \\ 1 & (1-2)\cdot 4 \pmod 9 \rightarrow \mathbf{5} & & \\ 1 & (1-2)\cdot 8 \pmod{11} \rightarrow 3 & (3-5)\cdot 5 \pmod{11} \rightarrow \mathbf{1} & \end{array}$$

An Example of the MRC Algorithm

- Step 3: To compute x, we perform the summation

$$\begin{aligned} x &= r_{11} + r_{22} \cdot n_1 + r_{33} \cdot n_1 \cdot n_2 \\ &= \mathbf{2} + \mathbf{5} \cdot 7 + \mathbf{1} \cdot 7 \cdot 9 \\ &= 100 \end{aligned}$$

- Until the Step 3, all computations are in single-precision, assuming each modulus is a single-precision integer