The Chinese Remainder Theorem

Theorem. Let $n_1, \ldots, n_r$ be r positive integers relatively prime in pairs. (That is, $\gcd(n_i, n_j) = 1$ whenever $1 \leq i < j \leq r$.)
Let $a_1, \ldots, a_r$ be any r integers. Then the r congruences

$$x \equiv a_i \pmod{n_i}$$

for $i = 1, \ldots, r$ have common solutions. Any two common solutions are congruent modulo

$$n = n_1 \cdots n_r$$

.

The proof gives an algorithm for computing the common solution.

Proof: For $j = 1, \ldots, r$, the number n/n_j is an integer and

$$\gcd(n/n_j, n_j) = 1,$$

so there is an integer b_j such that

$$(n/n_j)b_j \equiv 1 \pmod{n_j}.$$

Clearly, $(n/n_j)b_j \equiv 0 \pmod{n_i}$ if $i \neq j$. Let

$$x_0 = \sum_{j=1}^{r} (n/n_j)b_j a_j.$$

Then

$$x_0 = \sum_{j=1}^{r} (n/n_j)b_j a_j = \sum_{j=1}^{r} \delta_{ij} a_j \equiv a_i \pmod{n_i}.$$

Thus there is a common solution x_0.

If x_1 is another common solution, then $n_i \mid (x_0 - x_1)$ for each i, so $n \mid (x_0 - x_1)$ because the moduli are relatively prime in pairs.

Example: Solve the system of congruences

$$\begin{aligned} x &\equiv 1 \pmod 7 \\ x &\equiv 3 \pmod{10} \\ x &\equiv 8 \pmod{13}. \end{aligned}$$

Note that the hypotheses of the Chinese remainder theorem are satisfied in this example because any two of the moduli 7, 10, 13 are relatively prime.

We have $n_1 = 7$, $n_2 = 10$, $n_3 = 13$, $a_1 = 1$, $a_2 = 3$, $a_3 = 8$ and $n = 910$. Then $n/n_1 = 10 \cdot 13 \equiv 4 \pmod 7$. The extended Euclidean algorithm gives $b_1 \equiv 4^{-1} \equiv 2 \pmod 7$. Likewise, $b_2 \equiv 1^{-1} \equiv 1 \pmod{10}$ and $b_3 \equiv 5^{-1} \equiv 8 \pmod{13}$. Then

$x \equiv 130{\cdot}2{\cdot}1{+}91{\cdot}1{\cdot}3{+}70{\cdot}8{\cdot}8 \equiv 463 \pmod{910}$.

Solving $x^2 \equiv a \pmod{n}$

We have said nothing (so far) about whether one can solve $x^2 \equiv a \pmod{n}$ when n is a composite number.

We have also said nothing about *how* to solve it if it has a solution.

There are probabilistic polynomial time algorithms (Tonelli and Cipolla) to compute square roots of QR's mod p, where p is prime. They work well for numbers of hundreds of digits, but are too complicated to present here.

Here is a simple algorithm that finds square roots of QR's modulo any prime $p \equiv 3 \pmod 4$, that is, it works for half of the primes.

If $p \equiv 3 \pmod 4$, then the solutions to $x^2 \equiv a \pmod p$ are $x_1 \equiv a^{(p+1)/4} \pmod p$ and $x_2 = p - x_1$.

To see that this works, note that

$$x_1^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv a \pmod p$$

since $a^{(p-1)/2} \equiv +1 \pmod p$ by Euler's Criterion and the fact that a is a QR mod p.

(When a is a quadratic nonresidue modulo p, with $p \equiv 3$ (mod 4), $-a$ is a quadratic residue modulo p, and the formulas for x_1 and x_2 give the two square roots of $-a$ modulo p because

$$x_1^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv -a \pmod p$$

since $a^{(p-1)/2} \equiv -1 \pmod p$ by Euler's Criterion and the fact that a is a QNR mod p.)

Now I will tell you how to solve $x^2 \equiv a \pmod{n}$ when $n = pq$ is the product of two primes $p \equiv q \equiv 3 \pmod{4}$, an important special case.

Separately solve $y^2 \equiv a \pmod{p}$, with solutions y_1 and y_2, and $z^2 \equiv a \pmod{q}$, with solutions z_1 and z_2. Then use the CRT four times to solve the four systems

$$x \equiv y_i \pmod{p} \qquad x \equiv z_j \pmod{q}$$

for $i = 1, 2; j = 1, 2$. This will produce *four* different roots to $x^2 \equiv a \pmod{n}$.

Example. Find all four square roots of 11 modulo 133.

Factor $133 = 7 \cdot 19$. We must first solve $x^2 \equiv 11 \pmod{p}$ for $p = 7$ and for $p = 19$.

$11 \bmod 7 = 4$, which happens to be 2^2. So the solution to $x^2 \equiv 11 \pmod 7$ is $x \equiv \pm 2 \pmod 7$, or $x \equiv 2$ or $5 \pmod 7$.

$11 \bmod 19 = 11$, so we use exponentiation:

$$x \equiv 11^{(19+1)/4} = 11^5 \equiv 7 \pmod{19}.$$

So the solution to $x^2 \equiv 11 \pmod{19}$ is $x \equiv \pm 7 \pmod{19}$, or $x \equiv 7$ or $12 \pmod{19}$.

We have to solve the four CRT problems:

$$\begin{aligned} x_1 &\equiv 2 \pmod 7 \\ x_1 &\equiv 7 \pmod{19}. \end{aligned}$$

$$\begin{aligned} x_2 &\equiv 2 \pmod 7 \\ x_2 &\equiv 12 \pmod{19}. \end{aligned}$$

$$\begin{aligned} x_3 &\equiv 5 \pmod 7 \\ x_3 &\equiv 7 \pmod{19}. \end{aligned}$$

$$\begin{aligned} x_4 &\equiv 5 \pmod 7 \\ x_4 &\equiv 12 \pmod{19}. \end{aligned}$$

We begin the CRT by solving $19x + 7y = 1$ by the extended Euclidean algorithm.

It gives $19(3) + 7(-8) = 1$. We have found both b_1 and b_2 in the CRT by one extended Euclidean algorithm.

In all four CRT problems we have $n_1 = 7$, $n_2 = 19$, $b_1 = 3$ and $b_2 \equiv -8 \equiv 11 \pmod{19}$.

In the first CRT, we have $a_1 = 2$ and $a_2 = 7$. The solution is

$$x_1 = 19 \cdot 3 \cdot 2 + 7 \cdot 11 \cdot 7 = 653 \equiv 121 \pmod{133}.$$

We also get $x_4 = 133 - x_1 = 133 - 121 = 12$.

In the second CRT, we have $a_1 = 2$ and $a_2 = 12$. The solution is

$$x_2 = 19 \cdot 3 \cdot 2 + 7 \cdot 11 \cdot 12 = 1038 \equiv 107 \pmod{133}.$$

We also get $x_3 = 133 - x_2 = 133 - 107 = 26$.

The four square roots of 11 modulo 133 are 121, 107, 26, 12.

An application of finding square roots modulo n is the Rabin-Blum Oblivious Transfer or Coin Flipping Protocol. In it, Alice reveals a secret to Bob with probability 0.5.

In the Oblivious Transfer version, Alice doesn't know whether Bob got the secret or not (and this outcome must be acceptable to both participants).

In the Coin Tossing version, Bob tells Alice whether he got the secret. He wins the coin toss if he did get it; loses otherwise.

Alice's secret is the factorization of a number $n = pq$ which is the product of two large primes $p \equiv q \equiv 3 \pmod 4$.

1. Alice sends n to Bob.

2. Bob picks a random x in $\sqrt{n} < x < n$ with $\gcd(x, n) = 1$. Bob computes $a = x^2 \bmod n$ and sends a to Alice.

3. Knowing p and q, Alice computes the four solutions to $x^2 \equiv a \pmod n$. They are x, $n - x$, y and $n - y$, for some y. These are just four numbers to Alice. She doesn't know which ones are x and $n - x$. She chooses one of the four numbers at random and sends it to Bob.

4. If Bob receives x or $n-x$, he learns nothing. But, if Bob receives y or $n - y$, he can factor n by computing $\gcd(x + y, n) = p$ or q.

Why can Bob factor n if he gets y or $n - y$?

Theorem. If $n = pq$ is the product of two distinct primes, and if $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y \pmod{n}$, then $\gcd(x + y, n) = p$ or q.

Proof: We are given that n divides $(x+y)(x-y)$ but not $(x+y)$ or $(x-y)$. Hence, one of p, q must divide $(x + y)$ and the other must divide $(x - y)$.

It is easy to modify the Oblivious Transfer protocol to let Alice give Bob the content of an arbitrary file with probability 0.5. Alice's secret is the content of the file.

Alice enciphers the file using AES with secret key K. She gives the ciphertext of the file to Bob.

Alice chooses two large primes $p \equiv q \equiv 3 \pmod 4$, sets $n = pq$ and chooses $0 < e < n$ with $\gcd(e, (p-1)(q-1)) = 1$. This sets up an RSA public key cipher with public key n and e. Alice enciphers K as $C = K^e \bmod n$. Alice gives Bob C and e.

Then Alice and Bob do the Oblivious Transfer protocol, Alice sending n to Bob in Step 1.

If Bob learns the factorization of $n = pq$ in Step 4, then Bob finds d with $ed \equiv 1 \pmod{(p-1)(q-1)}$ by extended Euclid. He finds $K = C^d \bmod n$, and deciphers the file using K as the AES key.

Zero-Knowledge Proofs

This protocol is closely related to the oblivious transfer protocol. The difference is that Alice wants to convince Bob that she knows the factors of $n = pq$, but does not want to reveal the factors to Bob.

Alice (the prover) convinces Bob (the verifier) that she knows the prime factorization of a large composite number n, but does not give Bob any hint which would help him find the factors of n. Bob learns nothing about the factorization of n during the protocol that he could not have deduced on his own without Alice's help.

Roughly speaking, Bob gives Alice some quadratic residues modulo n and Alice replies with their square roots. The difficulty with this simple approach is that when Alice replies to Bob with a square root, there is a 50% chance that she will reveal the factorization of n to Bob, as in the oblivious transfer protocol.

Here is a good way to do the zero-knowledge proof protocol:

Alice knows n, p and q. Bob knows n but not p or q.

1. Alice chooses a in $\sqrt{n} < a < n$ and computes $b = a^2 \bmod n$.

2. At the same time, Bob chooses c in $\sqrt{n} < c < n$ and computes $d = c^2 \bmod n$.

3. Alice sends b to Bob and Bob sends d to Alice.

4. Alice receives d and solves $x^2 \equiv bd \pmod{n}$. (Note that this is possible because bd is a QR and she can compute its square root because she knows the factors of n.) Let x_1 be one solution of this congruence.

5. At the same time, Bob tosses a fair coin and gets Heads or Tails each with probability 0.5. Bob sends H or T to Alice.

6. If Alice receives H, she sends a to Bob. If Alice receives T, she sends x_1 to Bob.

7. If Bob sent H to Alice, then he receives a from Alice and checks that $a^2 \equiv b \pmod{n}$. If Bob sent T to Alice, then he receives x_1 from Alice and checks that $x_1^2 \equiv bd \pmod{n}$.

Alice and Bob repeat steps 1 through 7 many (20 or 30) times.

If the check in step 7 is always okay, then Bob accepts that Alice knows the factorization of n.

But if Alice ever fails even one test, then Bob concludes that Alice is lying.

Why does this protocol work?

Why does Bob not learn the factors of n?