A Math Library for Cryptography

Number : 199606131115

Extended Euclidean Algorithm

We want to have a bigger than b, so we swap the integers if that is not the case.

We initialize the algorithm with a = 1*s + 0*t and b=0*s + 1*t. Then we iterate while the gcd is not equal to zero by applying the extended Euclidean algorithm (we compute the new gcd, the new s and the new t each time using the quotient q and the old values).

We can then return the result (we re-swap s and t if we swapped a and b before) with the gcd, s and t.

Euler’s Phi Function (Totient)

The Euler’s totient function counts for a given n, all the positive integers that relatively prime to n. If n and an integer i are relatively prime it means that their GCD is equal to 1.

We added a little GCD recursive function which computes the GCD of two numbers a and b faster than the EEA. this function is basically the classic Euclidean Algorithm.

Then, we just loop on all the integers before n and if the GCD of i and n is 1 then we add upgrade by 1 our counter. We just have to return the counter in the end.

Modular Inverse

First, we compute the GCD of n and m by using the Extended Euclidean Algorithm (we make sure to send positive values to the algorithm).

If the GCD is not 1, then it is not invertible and we return 0.

Otherwise, we just have to return the s value from the Extended Euclidean Algorithm because we have 1=s*n+t*m=s*n (mod m). (Here, we also make sure to return a positive value).

Fermat Primality Test

We follow the algorithm and we test all values a between 2 and n/3. We compute each time a^(n-1) (mod n).

If the number n is a Fermat Prime, it should be equal to 1 for all the values computed (with a between 2 and n/3). If the value computed is not equal to 1, then we return the current value of a, which correspond to the lowest Fermat Witness.

Hash Collision Probability

As it is said in Lecture #11, the probability of a collision is : $1 - \frac{B-1}{B}\frac{B-2}{B}...\frac{B-(n-1)}{B}$. Here, B is the size or the number of different output values and n is the number of samples.

We just initialize a variable probability to 1, then we loop on the number of samples and we multiply the preceding probability with the one relative to the current number of samples. In the end, we return the 1 minus the probability as in the formula we have just written above. If do not perform this computation we have the probability that every output values are different.