Cryptography - Programming Assignment.

Marios Aspris 19880415599,
Anders Hansson 199401202297

December 2017

1 Introduction

In this report, we specify how we have solved the programming assignments. We used 19880415599 (Marios personal number) as key to the assignments.

2 Problem 1 - A crypto library

2.1 Extended Euclidean Algorithm

In our implementation of the extended euclidean algorithm(EEA) receives two numbers as input a, b such that $a > b$. It returns the greatest common divisor(GCD) and coefficients c_a, c_b in Bézout's identity for a and b. It starts by checking for the special case when $a = b$ and if so, returns $GCD(a, b) = a, c_a = 1, c_b = 0$. Then it checks for another special case if $a < b$ and then just swaps the variables. Then by iteratively removing remainders until the remainder equals zero, the coefficients c_a, c_b and $GCD(a, b)$ are computed.

2.2 Euler's Φ Function

The Euler Phi function of a number n is, the number of numbers a between 1 <= $a < n$ that have a greatest common divisor of 1. Therefore I have created a helper method that finds the gcd between two numbers, and then I loop from the number till number n-1, and if the condition gcd(n, x) equals to 1, then I increment a number i by 1, which will be the final result, the answer to the euler ϕ.

2.3 Modular Inverse

This function inputs two integers n, m and returns the modular inverse n^{-1} in mod m.

We start by checking if a modular inverse exists. The inverse exists if the $GCD(a, b) = 1$, if the inverse does not exist, then the function returns 0. Since the modular inverse equals one of coefficients in Bézout's identity, we use our EEA function to compute the GCD and the modular inverse. If the inverse c is negative we just reverse it, $c < 0, c + m \pmod{m}$.

2.4 Fermat Primality Test

To use the Fermat primality requires to check for $a^{p-1} \equiv 1 \ (mod \ p)$, where p is the number we want to determine if it is prime or, not and a is an arbitrary number smaller than p. I loop through all the numbers for a, and I break the loop when I find the smallest number that proves compositeness of the number p. If the number p is prime 0 is returned. I do the calculations with modulo exponentiation, because for large numbers, powers of numbers will be impossible. I do modulo exponentiation by taking the number a and performing modulo exponentiation. For the next iteration, the result from the previous modulo exponentiation will be multiplied by the number a, and taking the modulo operation again. This is repeated for the numbers we want to raise the power to, in this case p-1.

2.5 Hash Collision Probability

The Hash Collision Probability is calculated as a combinatorial probability. Specifically , the probability is calculated as
$prob = 1 - \frac{H!}{(H-n)!H^n}$. Now H is the size of the problem, and n is the sample that we want to calculate the probability for. The factorial is expensive computation, therefore is calculated as $Pr = \frac{H-n+1}{H} . \frac{H-n+2}{H} . \frac{H-n+3}{H} ... \frac{H-n+n}{H}$.
This is how I solve it in a for loop, which is fast and accurate.

3 Problem 2 - Decrypting CBC with simple XOR

In the CBC encryption a message is encrypted as $c_i = E(k, m_i \oplus c_{i-1}) = k \oplus (m_0 \oplus c_{i-1})$, where $\oplus$ stands for the mod(2) operation or XOR, m_0 is the known message in the first block, IV is the initial randomized vector, and k is the key to encrypt this. The decryption step is $m_i = D(k, c_i, c_{i-1}) = k \oplus (c_i \oplus c_{i-1})$. Therefore since the first block of the message is given, and the IV is the first 12 byte block with c_0 the next 12 byte block, we can recover the encryption key k as, $k = (IV \oplus m_0) \oplus c_0$. After recovering the key, we used it to decrypt the rest of the plain message blocks, starting from the last 12 bytes of the vector, as described above.

The final message decrypted was:
198804155994Light thinks it travels faster than anything but it is wrong. No matter how fast light travels, it finds the darkness has always got there first, and is waiting for it. - Terry Pratchett000000
We can see also in padding which is a type 0 pad block.