

HW #3: Public Key Cryptography and Hash Functions

CS 392/681: Computer Security
Fall 2006

[100pts] DUE 10/01/2006 (midnight)

Problem 1 [5+5+7.5+7.5=25pts]

- (1) We discussed the Euclidian algorithm to compute gcd of two given numbers. Use the algorithm to compute (show all steps involved)
 - a. $\text{gcd}(34,276)$
 - b. $\text{gcd}(24,789)$

- (2) We also discussed the Extended Euclidian algorithm to find out modular inverses. Use the algorithm to compute (show all steps involved)
 - a. $5^{-1} \pmod{26}$
 - b. $111111 \pmod{11111111111111}$

- (3) We know that $a^{-1} \pmod{n}$ exists if $\text{gcd}(a,n)=1$. Let n be a product of two **large** (512-bit long) primes p, q , i.e. $n=pq$ (in other words, n is RSA modulus). n is public, while p, q are kept private. Given n , can you find a number $a < n$, such that $a^{-1} \pmod{n}$ does not exist? Explain why or why not?

- (4) Let n be a product of two large primes p, q , i.e. $n=pq$. If x and y are two numbers relatively prime to n such that $x = y \pmod{n}$, and g is relatively prime to n , then
 - a. Is $g^x = g^y \pmod{n}$? Why or why not? What if $x = y \pmod{\Phi(n)}$?

Problem 2 [25pts]

The reading assignment in the last lecture was on “Chinese Remainder Theroem (CRT)”. Use CRT to solve for the following set of modular equations (show all steps involved):

$$x = 2 \pmod{3}$$

$$x = 5 \pmod{7}$$

$$x = 1 \pmod{2}$$

Understand (don't simply copy-paste!) and explain how CRT can be used to speed-up RSA decryption. How fast is “RSA decryption with CRT” in comparison to “RSA decryption without CRT”?

Problem 3 [30pts]

In this exercise, you will get some hands on experience with RSA encryption/decryption.

Download RSA code from:

<http://www.funet.fi/pub/crypt/cryptography/asymmetric/rsa/rsaref2.tar.gz>. Get familiarized with RSA key generation, encryption, and decryption functions and the sample code.

1. execute the key generation function to generate the public key (e,n) and private key d
 - o compute the execution time
2. choose any message $M < n$ and execute the encryption function to encrypt M using (e,n) , and obtain the ciphertext C ;
 - o compute the execution time
3. execute the decryption function to decrypt C using d , and to obtain M back.
 - o compute the execution time

Repeat each of the above an appropriate number of times (as you did in AES exercise of HW#2, but with fewer operations, as RSA is much slower than AES) and give the average execution time for each of the three functions. List the type and speed of the processor, and the memory (RAM) of the machine you execute the code on.

Also, answer what is the most costly computation in the key generation? What algorithm is being used for this computation in the source code provided? Repeat this computation an appropriate number of times and output the average processing time for it.

Problem 4 [20pts]

Assume that the IP address assignment is performed using the following function: $IP: H \rightarrow \{0,1\}^{32}$, which on input of a host h on the internet, outputs its IP address as a 32-bit long number uniformly distributed at random. How many hosts would we need on the internet such that at least two have of them have the same IP address with a probability greater than 0.8? Show all steps involved. Does this represent a good way for IP address assignment? Why or why not?