Problem Set 5

Due on Wednesday, November 13, 2013.

Instructions

Work the problems below, prepare your answers in electronic form, and submit your solutions using the submit script on the Zoo. Remember to specify "5" for the problem number argument to `submit`.

Problem 1 requires some computation. You may use a calculator or computer for the calculations, but you **must** show your work.

Problem 3 is a small programming problem that must be solved on the Zoo since it uses a pre-compiled library that I am furnishing.

Problem 1: Quadratic Residues and the Legendre Symbol

(a) Let $p = 19$, $q = 37$, and $n = p \times q = 703$. For each $i, j \in \{-1, 1\}$, find a number $x_{i,j} \in \mathbf{Z}_n^*$ such that

$$\left(\frac{x_{i,j}}{p}\right) = i \quad \text{and} \quad \left(\frac{x_{i,j}}{q}\right) = j.$$

(b) Use the Euler criterion to justify your answers to part (a).

Problem 2: ElGamal Authentication

Once Happy understood ElGamal signatures, he was excited to use them for authentication. He wants to send an authenticated message m to Bob so that Bob can verify that m came from him.

Here's his idea. Assume that Happy has an ElGamal signing key (g, p, x) and Bob has the corresponding verification key (g, p, a). We denote the signing algorithm using that key pair by S and the verification algorithm by V.

	Happy		Bob
1.		$\xleftarrow{r}$	Choose random string r.
2.	Compute $s = S_A(r)$	$\xrightarrow{s,m}$	Check $V_A(r, s)$. Accept m as coming from Happy if check succeeds.

(a) Mallory wants to get Bob to accept a message m' of his choosing. Describe in detail how he can do this using a man-in-the-middle attack.

(b) Suggest a way to fix this protocol to thwart Happy's attack. Your suggestion should not use any more rounds of communication nor assume any other encryption system or secret keys. [Hint: Think about using a secure hash function H to somehow "bind" m to the signature.]

Problem 3: Extending Hash Functions

Happy threw together a hash function h : 32-bits $\rightarrow$ 16-bits, which he implemented by a C function `hash32()`. Adapting Method 2 from slide 28 of Lecture 14, Happy defined a new hash function H : 64-bits $\rightarrow$ 16-bits and implemented it by a C function `hash64`. Since he didn't know how to find colliding pairs for h, he thought that H would also be collision-free.

Clever Clem was able to find lots of colliding pairs for H. He didn't want to tell Happy how he did it, but he presented Happy with a file `H-collisions` of colliding pairs for H, each line of which consists of two 64-bit whitespace-separated hex numbers.

Your job is to write a program `breakHash.c` that applies the ideas presented on slide 29 of Lecture 14 to find a corresponding colliding pair for h. Your program should take as a command line argument the name of a file containing pairs of collisions for H. You should read each line, determine whether case 1 or case 2 applies, and find the corresponding colliding pair for h. You should then write a line to standard output consisting of 5 numbers: the original colliding pair for H, the case number that pertains (1 or 2), and the colliding pair for h described by that case. Colliding pairs should be written in hex with the `0x`-prefix (as in the input file). The case number should be written as a single digit. In case 2, if both $m_1 \neq m_1'$ and $m_2 \neq m_2'$, then print the first colliding pair for h.

Three files for this assignment have been placed in the Zoo directory `/c/cs467/assignments/ps5/`: a header file `ps5.h` containing prototypes for `hash32` and `hash64`, a library `libps5.a` containing linux executables of those two functions, and the data file `H-collisions`.