C&O 487: Assignment #2

Due date: February 6 (Friday)

1. **Chosen-plaintext attack on a Feistel cipher** (2+3+5+5 marks)
Notation: In this question, bitstrings of length 4 are represented in hexadecimal notation. That is $0000 \leftrightarrow$ `0`, $0001 \leftrightarrow$ `1`, $0010 \leftrightarrow$ `2`, $0011 \leftrightarrow$ `3`, $0100 \leftrightarrow$ `4`, $0101 \leftrightarrow$ `5`, $0110 \leftrightarrow$ `6`, $0111 \leftrightarrow$ `7`, $1000 \leftrightarrow$ `8`, $1001 \leftrightarrow$ `9`, $1010 \leftrightarrow$ `a`, $1011 \leftrightarrow$ `b`, $1100 \leftrightarrow$ `c`, $1101 \leftrightarrow$ `d`, $1110 \leftrightarrow$ `e`, $1111 \leftrightarrow$ `f`.

Consider the Feistel cipher F defined as follows.

(A) $n = 24$.

(B) $h = 3$.

(C) The key is a randomly chosen function $S_k : \{0,1\}^3 \to \{0,1\}^3$.

(D) Each subkey is S_k itself.

(E) The component function $f : \{0,1\}^{24} \to \{0,1\}^{24}$ that is used in each round of encryption is the following. (This is similar to the component function for the NDS cipher presented in class. You should find it useful to draw a diagram to illustrate this function, like the NDS diagram handed out in class.) To calculate $f(z)$ where $z \in \{0,1\}^{24}$, do:

(i) Divide z into 3 bytes: $z = (z^1, z^2, z^3)$.

(ii) Divide each byte z^j into two nibbles: $z^j = (n_1^j, n_2^j)$ (for $1 \le j \le 3$).

(iii) Apply $S_0 : \{0,1\}^4 \to \{0,1\}^4$ to each n_1^j to get p_1^j (for $1 \le j \le 3$). S_0 is defined as follows:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S_0(x)$	1	9	2	9	a	7	f	e	6	b	c	8	4	3	2	0

(iv) Apply $S_1 : \{0,1\}^4 \to \{0,1\}^4$ to each n_2^j to get p_2^j (for $1 \le j \le 3$). S_1 is defined as follows:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S_1(x)$	6	b	b	a	a	c	c	d	d	1	1	2	2	4	5	7

(v) Let z^* be the 3-bit string obtained by taking the first bit of each byte of z, and compute the 3-bit string $t = S_k(z^*)$.

(vi) For each $j \in \{1,2,3\}$ do the following:
If the jth bit of t is 1, then swap p_1^j and p_2^j.

(vii) Apply the following permutation to the 6 transformed nibbles p_1^1, p_2^1, p_1^2, p_2^2, p_1^3, p_2^3:
Swap p_1^1 and p_1^2, swap p_2^1 and p_2^3, swap p_2^2 and p_1^3.

(viii) The resulting 24-bit string $(p_1^1, p_2^1, p_1^2, p_2^2, p_1^3, p_2^3)$ is the output $f(z)$.

(To make sure you understand the description of this encryption function, you may wish to verify that if the secret key S_k is the following:

y	0	1	2	3	4	5	6	7
$S_k(y)$	2	5	7	6	6	6	3	4

then the plaintext m = `001122334455` gets encrypted to c = `5e629957d73f`. The intermediate blocks are m_0 = `001122`, m_1 = `334455`, m_2 = `ac8688`, m_3 = `5e6299`. m_4 = `57d73f`.)

(a) What is the size of the key space? (Or, equivalently, what is the value of the parameter l?)

(b) Encrypt the plaintext $m =$ `001122334455` with the secret key

y	0	1	2	3	4	5	6	7
$S_k(y)$	2	2	3	3	4	4	5	5

Hand in the ciphertext (and clearly labelled intermediate calculations if you want part marks).

(c) Suppose now that the secret key is unknown. You have access to an encryption oracle (see the course web site). Determine $S_k(0)$. Hand in a *brief* description of your attack, and the list of plaintext/ciphertext pairs you obtained from the encryption oracle.

(d) Decrypt the ciphertext $c =$ `0123456789ab`. Here, the secret key is unknown (the same key as in (c)). You have access to an encryption oracle (see the course web site).
(Hint: decryption is very easy with the encryption oracle!)

2. **Meet-in-the-middle attack on Double-DES** (7+3 marks)
In this question, you have to assume that the total amount of storage space available is 1,000 Terabytes ($\approx 2^{53}$ bits).

(a) Describe a known-plaintext meet-in-the-middle attack on Double-DES that recovers the secret key. You should justify the correctness of the algorithm, estimate the expected number of DES operations the algorithm takes, and justify that the algorithm requires less than 1,000 Terabytes of storage. (A *DES operation* is a single-DES encryption or a single-DES decryption.) Of course, you should design your algorithm to be as fast as possible.

(b) Suppose that you have access to a network of 1,000 workstations, each workstation capable of performing 5,000,000 DES operations per second. Estimate the expected calendar time your algorithm takes when run on this network. Compare this to the estimated calendar time that exhaustive key search on Double-DES would take when run on the same network.

3. **Hash functions derived from block ciphers** (3+7 marks)
Let E denote a block cipher where plaintext blocks, ciphertext blocks, and keys are each 128 bits in length. Consider the following hash function $H : \{0,1\}^* \to \{0,1\}^{128}$. To compute the hash of some message $M \in \{0,1\}^*$, one does the following:

(i) Append just enough 0's to the end of M so that the length of the resulting bitstring $\overline{M}$ is a multiple of 128.

(ii) Let $\overline{M} = M_1 M_2 \ldots M_l$ where each M_i is a bitstring of length 128.

(iii) Let h_0 be the all-zeros string of bitlength 128.

(iv) Compute $h_i = E_{M_i}(h_{i-1})$ for $1 \le i \le l$.

(v) Define $H(M) = h_l$.

(a) Show that H is not collision resistant. (Hint: This is trivial!)

(b) Is H preimage resistant? Justify your answer.
(Hint: The following variant of the birthday paradox may be useful. Suppose that we have two urns, one containing n black balls numbered $1, 2, \ldots, n$, and the other containing n white balls numbered $1, 2, \ldots, n$. Suppose that t black balls are drawn, with replacement, from the first urn and their numbers listed. Next, t white balls are drawn, with replacement, from the second urn and their numbers listed. If $t \approx \sqrt{n}$, then with probability at least $\frac{1}{2}$ at least one black ball drawn has the same number as a white ball drawn.)

4. **Hash functions** (5 marks)
   Suppose that $H : \{0,1\}^* \to \{0,1\}^n$ is a collision-resistant hash function. Define a new hash function $G : \{0,1\}^* \to \{0,1\}^n$ by $G(x) = H(H(x))$. Prove that G is also collision resistant.

5. **Review of elementary number theory** (0 marks)
   In preparation for the upcoming lectures on public-key cryptography, please review the chapters *Integers* and *Congruences* of your Math 135 Classical Algebra text. I will assume in class that you know the basic properties of divisibility, greatest common divisors, prime numbers, Fermat's little theorem, linear congruences, the integers modulo n, and the Chinese Remainder Theorem. I will also assume in class that you can find greatest common divisors using the Euclidean algorithm, can find inverses of integers modulo n using the extended Euclidean algorithm, and can solve linear systems of congruences.

   Exercises (not to be handed in):

   (a) Use the Euclidean algorithm to compute $\gcd(12684, 16044)$.
   (b) Use the extended Euclidean algorithm to find integers x and y such that $1234x + 4321y = 1$.
   (c) Use the extended Euclidean algorithm to compute $47^{-1} \bmod 167$. (That is find the integer x, such that $47x \equiv 1 \pmod{167}$ and $0 \le x \le 166$.)
   (d) Solve the congruence $47x \equiv 17 \pmod{167}$.
   (e) Solve the following simultaneous congrences:

   $$\begin{aligned} x &\equiv 46 \pmod{51} \\ x &\equiv 27 \pmod{52}. \end{aligned}$$

   (f) State Fermat's little theorem.

Please note that assignments are not weighted equally. The total marks received on assignments will be added together at the end of the course.

You are welcome to collaborate on assignments with your colleagues. However, solutions must be written up by yourself. If you do collaborate, please acknowledge your collaborators in the write-up for each problem. If you obtain a solution with help from a book, paper, wikipedia, a web site, solutions from previous offerings of the course, *or any other source*, please acknowledge your source. You are not allowed to solicit help from online bulletin boards, chat groups, or newsgroups.

The assignment is due at the *beginning* of class on February 6. Late assignments will not be accepted except in *very* special circumstances.
