C&O 487: Assignment #3 Due date: February 27 (Friday)

---

1. **MAC schemes derived from block ciphers** (10 marks)
Let E denote the family of encryption functions for a block cipher where plaintext blocks, ciphertext blocks, and keys are each 80 bits in length. Let $H : \{0,1\}^* \to \{0,1\}^{80}$ be a hash function. Assume that plaintext messages m all have bitlengths that are multiples of 80; we write $m = (m_1, m_2, \ldots, m_t)$, where each m_i is an 80-bit block and t is the length of m in blocks.
Consider the following four MAC schemes each with 80-bit secret key k.

   (i) $\mathrm{MAC}_k(m) = c_t$, where $c_0 = 0$, and $c_i = c_{i-1} \oplus m_i \oplus E_k(m_i)$ for $1 \le i \le t$.

   (ii) The sender selects $c_0 \in_R \{0,1\}^{80}$ and computes $\mathrm{MAC}_k(c_0, m) = c_t$, where $c_1 = E_k(c_0 \oplus m_1)$, and $c_i = c_{i-1} \oplus E_k(m_{i-1} \oplus m_i)$ for $2 \le i \le t$. The sender transmits (m, c_0, c_t). The receiver recomputes $\mathrm{MAC}_k(c_0, m)$ and compares it with c_t.

   (iii) $\mathrm{MAC}_k(m) = E_k(H(m))$.

   (iv) The sender selects $c_0 \in_R \{0,1\}^{80}$ and computes $\mathrm{MAC}_k(c_0, m) = h$, where $c_i = E_k(m_i \oplus c_{i-1})$ for $1 \le i \le t$ and $h = H(c_t)$. The sender transmits (m, c_0, h). The receiver recomputes $\mathrm{MAC}_k(c_0, m)$ and compares it with h.

   Are any of these MAC schemes secure? (Justify your answer.)
   (It will help to review the definition of a secure MAC scheme.)

2. **Another MAC scheme derived from a block cipher** (10 marks)
Recall that AES is a block cipher where plaintext blocks, ciphertext blocks, and keys are each 128 bits in length. Let p be the prime $2^{128} - 173$. Consider the MAC scheme where secret keys are pairs (k, b) with $k \in_R \{0,1\}^{128}$ and $b \in_R [1, p-1]$. The tag $\mathrm{MAC}_{(k,b)}(m)$ on a message $m \in \{0,1\}^*$ is computed as follows: First, a 1 bit is appended to the right of m, followed by just enough 0 bits so that the bitlength of the extended message m' is a multiple of 128. Next, m' is divided into 128-bit blocks: m_1, $m_2 \ldots, m_t$, and $c_i = \mathrm{AES}_k(m_i)$ is computed for $1 \le i \le t$. Finally, the MAC tag of m is defined to be $\mathrm{MAC}_{(k,b)}(m) = (\sum_{i=1}^{t} c_i b^{i-1}) \bmod p$ (where the c_i are interpreted as integers).

   Suppose now than an attacker (who does not know the secret key (k, b) chosen by Alice) wishes to compute the MAC tag of an ℓ-block message $M = (M_1, M_2, \ldots, M_\ell)$ (each M_i has bitlength 128). The attacker can obtain the MAC tag of any messages of her choosing from Alice, except of course for the MAC tag of M itself. Show how the attacker can efficiently compute the MAC tag of M.

3. **RSA computations** (10 marks)
Alice chooses $n = 1073$ and $e = 715$ for use in the basic RSA public-key encryption scheme.

   (a) What is Alice's private key?

   (b) Encrypt the message $m = 3$ for Alice.

   (Please do not use Maple for this problem, except possibly to do the modular multiplications. The purpose of this problem is to make sure that you understand the basic RSA operations. Please show the main steps in your calculations.)

4. **Computing GCDs without long division** (15 marks)
One of the drawbacks of the Euclidean algorithm is that it requires long divisions which can be difficult to implement in software and hardware. The following algorithm for computing greatest common divisors only uses divisions by 2, an operation which can be easily implemented as a right-shift of the binary representation.

Algorithm: GCDeasy

Input: Positive integers a, b.

Output: $\gcd(a, b)$.

(i) $e \leftarrow 1$.

(ii) While a and b are both even do
$a \leftarrow a/2, \quad b \leftarrow b/2, \quad e \leftarrow 2e$.

(iii) While $a \neq 0$ do
While a is even do: $a \leftarrow a/2$.
While b is even do: $b \leftarrow b/2$.
If $a \geq b$ then
$a \leftarrow a - b$
else
$b \leftarrow b - a$

(iv) $d \leftarrow e \cdot b$.

(v) RETURN(d).

(a) Compute $\gcd(308, 440)$ using GCDeasy. (Show the main steps of your work.)

(b) Prove that GCDeasy *terminates*.

(c) Prove that GCDeasy is *correct* (that is, $d = \gcd(a, b)$). The following result from Math 135 will be useful: $\gcd(a, b) = \gcd(a, b - a)$ for all integers a, b.

(d) Determine the running time of GCDeasy in bit operations. The running time should be expressed in terms of k, where k is the bitlength of the larger of a and b.
Hint: Consider the binary representations of a and b.

(e) Is GCDeasy a polynomial-time algorithm? (Justify your answer.)

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 27. Late assignments will not be accepted except in *very* special circumstances.
