Computer Science and Engineering, UCSD — Fall 10
CSE 107: Introduction to Modern Cryptography — **Instructor:** Mihir Bellare
Problem Set 5 — November 1, 2010

Problem Set 5

Due: Monday November 8, 2010, in class.

See course information section (on course web page) for instructions and rules on working on problem sets and turning them in.

Problem 1. [**40 points**] Let E: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a secure block cipher, where $k, l \geq 128$. Let $\mathcal{K}$ be the key-generation algorithm that returns a random k-bit key K. Let

$$\mathsf{Plaintexts} = \{ M \in \{0,1\}^l : 0 < |M| < l2^l \text{ and } |M| \bmod l = 0 \} .$$

Let $\mathcal{T}, \mathcal{V}$ be the following tagging and verification algorithms:

```
algorithm T_K(M)
    if M ∉ Plaintexts then return ⊥
    Break M into l bit blocks, M = M[1] ... M[n]
    M[n+1] ← ⟨n⟩
    C[0] ← 0^l
    for i = 1, ..., n+1 do
        C[i] ← E_K(C[i-1] ⊕ M[i])
    return C[n+1]
```

```
algorithm V_K(M, σ)
    if M ∉ Plaintexts then return 0
    if σ = T_K(M) then return 1
    else return 0
```

Above, $\langle n \rangle$ denotes the l-bit binary representation of the integer n.

Show that $\mathcal{MA} = (\mathcal{K}, \mathcal{T}, \mathcal{V})$ is an insecure message-authentication scheme by presenting a practical adversary A such that $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(A) = 1$. Say how many queries A makes to each of its oracles, and what is its running time. (The number of points you get depends on these quantities.)

Dicsussion. We saw that the CBC-MAC is not secure when one wants to authenticate strings of varying length. The above is a possible fix, which appends the number of blocks in the message to the message before computing the CBC-MAC. Your task is to show that this fix does not work, meaning the scheme is still insecure.

Problem 2. [**40 points**] Consider the following computational problem:

INPUT: N, a, b, x, y where $N \geq 1$ is an integer, $a, b \in \mathbf{Z}_N^*$ and x, y are integers with $0 \leq x, y < N$
OUTPUT: $a^x b^y \bmod N$

Let $k = |N|$. The naive algorithm for this first computes $a^x \bmod N$, then computes $b^y \bmod N$, and multiplies them modulo N. This has a worst case cost of $4k + 1$ multiplications modulo N.

Design an alternative, faster algorithm for this problem that uses at most $2k + 1$ multiplications modulo N.
