Problem Set 5 Solutions

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.

Recall that the adversary is given oracles $\mathbf{Tag}(\cdot)$ and $\mathbf{Verify}(\cdot,\cdot)$. Our adversary A proceeds as follows:

adversary A
$Tag_0 \leftarrow \mathbf{Tag}(0^l)$
$Tag_1 \leftarrow \mathbf{Tag}(0^l \,\|\, \langle 1 \rangle \,\|\, Tag_0)$
$d \leftarrow \mathbf{Verify}(0^l \,\|\, \langle 3 \rangle \,\|\, Tag_1, Tag_1)$

This adversary makes two queries to its $\mathbf{Tag}(\cdot)$ oracle and one to its $\mathbf{Verify}(\cdot,\cdot)$ oracle, and has running time $O(l)$ plus the time for the computations of responses to oracle queries. We claim that $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MA}}(A) = 1$. Let us now justify this. We let $Z = E_K(0)$. Then notice that

$$\begin{aligned} Tag_0 &= E_K(Z \oplus \langle 1 \rangle) \\ Tag_1 &= E_K(Z \oplus \langle 3 \rangle) \,. \end{aligned}$$

However, it is also the case that

$$\mathcal{T}_K(0^l \parallel \langle 3\rangle \parallel Tag_1) \quad = \quad E_K(Z \oplus \langle 3\rangle) \; .$$

Thus $\mathcal{V}_K$ will accept Tag_1 as the tag for $0^l \parallel \langle 3\rangle \parallel Tag_1$.

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.

Let us first explain the claim about the naive algorithm. On inputs N, a, b, x, y it would do the following:

$A \leftarrow$ MOD-EXP(a, x, N)
$B \leftarrow$ MOD-EXP(b, y, N)
$z \leftarrow$ MOD-MULT(A, B, N)
Return z

The algorithm MOD-EXP was presented in class and is shown in the slides for the Computational Number Theory chapter. It is the special case of algorithm EXP$_G$ when the group G is $\mathbf{Z}_N^*$. Each iteration of the for loop of that algorithm uses two modular multiplications in the worst case, the first to obtain $w = y^2 \bmod N$ from y and the second to obtain $w \cdot a^{b_i} \bmod N$. Thus, MOD-EXP uses $2k$ modular multiplications in all. So the above naive algorithm uses $4k+1$ modular multiplications.

The faster algorithm extends the ideas of EXP$_G$. It works as follows:

<u>**Alg** FASTEXP(N, a, b, x, y)</u>
Let $x_{k-1} \ldots x_1 x_0$ be the binary representation of x
Let $y_{k-1} \ldots y_1 y_0$ be the binary representation of y
$c \leftarrow ab \bmod N$
$z \leftarrow 1$
for $i = k-1$ downto 0 do
  if $x_i = 1$ and $y_i = 1$ then $z \leftarrow z^2 \cdot c \bmod N$
  if $x_i = 1$ and $y_i = 0$ then $z \leftarrow z^2 \cdot a \bmod N$
  if $x_i = 0$ and $y_i = 1$ then $z \leftarrow z^2 \cdot b \bmod N$
  if $x_i = 0$ and $y_i = 0$ then $z \leftarrow z^2 \bmod N$
return z

Since $0 \leq x, y < N$ and N is k-bits long, we know that x and y are also at most k bits long. Therefore, the number of iterations for the loop is at most k. Since each loop incurs at most two modular multiplications, the total number of multiplications in the for loop is $2k$. Adding the one multiplication done on the 4th line of the code to get c, we have that the total number of multiplications for FASTEXP is $2k + 1$ as desired.