Instructor: Adam O'Neill
adamo@cs.umass.edu

CS-466: Homework 7

Problem 1. (30 points.) Let G be the group $\mathbb{Z}_6^*$ under the operation of multiplication modulo 6. Is G cyclic? Prove your answer.

Problem 2. (40 points.) Let $p \geq 3$ be a prime, $g \in \mathbb{Z}_p^*$ be a generator of $\mathbb{Z}_p^*$, and $H\colon \mathbb{Z}_p^* \to \{0,1\}^k$ be a hash function. Consider the key-generation and encryption algorithms given below, where $M \in \{0,1\}^k$:

```
Algorithm $\mathcal{K}$:
    $x \leftarrow\!\!\$\ \mathbb{Z}_{p-1}^*$
    $X \leftarrow g^x \bmod p$
    Return $(X, x)$

Algorithm $\mathcal{E}(X, M)$:
    $y \leftarrow\!\!\$\ \mathbb{Z}_{p-1}$ ; $Y \leftarrow g^y \bmod p$
    $Z \leftarrow X^y \bmod p$ ; $W \leftarrow H(Y) \oplus M$
    Return $(Z, W)$
```

Specify an $O(|p|^3 + k)$-time (here $|p|$ denotes bit-length of p) decryption algorithm $\mathcal{D}$ such that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a public-key encryption scheme satisfying the correctness condition given in class, and prove this is the case (both the claim about the running-time and the correctness).

Problem 3. (40 points.) Let $\mathcal{K}_{\text{rsa}}$ be an RSA generator with modulus length k. Consider the key-generation and encryption algorithms given below, where $M \in \mathbb{Z}_N^*$:

```
Algorithm $\mathcal{K}$:
    $(N, p, q, e, d) \leftarrow\!\!\$\ \mathcal{K}_{\text{rsa}}$
    Return $(N, (N, p, q))$

Algorithm $\mathcal{E}(N, M)$:
    $C \leftarrow M^N \bmod N$
    Return $C$
```

Specify an $O(k^3)$-time decryption algorithm $\mathcal{D}$ such that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a public-key encryption scheme satisfying the correctness condition given in class, and prove this is the case (both the claim about the running-time and the correctness).

Problem 4. (20 points.) Recall that a message authentication code $\mathcal{T}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ has a "canonical" verification algorithm: On inputs m, σ, if $\mathcal{T}(K, m) = \sigma$ then output 1 (accept), else output 0 (reject). In the UF-CMA game, we did not give the adversary access to a verification oracle. It turns out that *black-box* access (the kind we have been considering in class) to a verification oracle does not help the adversary, But, to model non-blackbox access, suppose the verification oracle is augmented such that on input m', σ' it not only returns a bit indicating whether $\mathcal{T}(K, m) = \sigma$ but also the *time* it took to compute this equality check. The interpretation is that the adversary gets to know how long it took the receiver to accept or reject. Further, suppose the equality check is implemented in the straightforward way of checking one bit position of $\mathcal{T}(K, m)$ and σ at a time and stopping when a bit position differs.

Argue that this leads to a *timing attack*: the adversary is able to forge on an arbitrary message of its choice! That is, present pseudocode for an adversary playing this augmented version of the UF-CMA game that outputs a forgery on an arbitrary message m (20 points). Prove that your algorithm is correct.

For clarity, the augmented UF-CMA game referred to above is specified as followed, For a function family $\mathcal{T}: \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ define the games below:

```
proc INITIALIZE
$K \leftarrow\$ \mathcal{K}$
$S \leftarrow \emptyset$

proc TAG($x$)
$x \leftarrow S \cup \{x\}$
Return $\mathcal{T}(K, x)$

proc VER($x, \sigma$)
$acc \leftarrow \mathsf{false}$
If $x \in \mathcal{D}$ and $x \notin S$ and $\mathcal{T}(K, x) = \sigma$ then $acc \leftarrow \mathsf{true}$
Let $i$ be the number of steps it took for the above to execute
Return $(acc, i)$

proc FINALIZE($x, \sigma$)
$acc \leftarrow \mathsf{false}$
If $x \in \mathcal{D}$ and $x \notin S$ and $\mathcal{T}(K, x) = \sigma$
    Then $acc \leftarrow \mathsf{true}$
Return $acc$
```

Problem 5. (20 points.) Suppose we have a pseudorandom function $F\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ and want to build a PRF $G\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^{2k}$. (In other words, we want to double the output size.). Alice suggests setting $G(K, x)$ to $y_1 \| F(K, y_1)$ where $y_1 \leftarrow F(K, x)$. Does Alice's suggestion work? If yes, give convincing intuition. If no, give an attack with advantage and running-time analysis.