Problem 1: Attacking the RSA TDP

We saw in class how the RSA is a candidate trapdoor permutation (TDP) with the following algorithms:

\[ \text{KeyGen}(1^k) : \text{On input } 1^k, \text{ pick } 2 \text{-bit primes } p \text{ and } q \text{ and let } N = pq. \text{ Find } e \text{ relatively prime to } \phi(N) = \phi(pq) = (p-1)(q-1) \text{ where } \phi() \text{ is the Euler’s totient function. Let } d \text{ be such that } ed \equiv 1 \mod \phi(N). \text{ Output } PK = (N, e) \text{ and } SK = d. \]

\[ \text{Eval}(PK, x) : \text{If } x \notin \mathbb{Z}_N^* \text{ fail, else output } x^e \mod N. \]

\[ \text{Invert}(PK, SK, y) : \text{If } y \notin \mathbb{Z}_N^* \text{ fail, else output } y^d \mod N. \]

We will denote the TDP instantiated by \((N, e) \leftarrow \text{KeyGen}(1^k)\) as \(f\) and evaluating the TDP is \(f(x) = \text{Eval}((N, e), x) = x^e \mod N. \)

We will now see how this TDP is vulnerable to certain types of attacks:

a. **Malleability of** \(f\): We say that a function \(f\) is **malleable** if given the value of \(f(x)\), you can compute the value of \(f(g(x))\) for some function \(g()\) of your choosing, without knowing \(x\). Show that given the value of \(f(x)\), it is possible to compute the value of \(f(g(x))\) without knowing \(x\) and for \(g(x) = cx\) where \(c\) is a constant. Assume that \(g(x) \in \mathbb{Z}_N^*\) and hence \(f(g(x))\) does not fail.

b. **Inverting \(f\) for small \(e\) values**: Suppose you have three one-way functions: \(f_{N_1,e}()\), \(f_{N_2,e}()\) and \(f_{N_3,e}()\) all with \(e = 3\). Assume that \(\{N_i\}_{i=1}^3\) are pairwise relatively prime. Show that given values \(v_1 = f_{N_1,e}(a)\), \(v_2 = f_{N_2,e}(a)\) and \(v_3 = f_{N_3,e}(a)\) for \(0 \leq a < \min\{N_1, N_2, N_3\}\), it is possible to recover \(a\).

**Bonus Question**: For an arbitrary \(e\) relatively prime to \(\phi(N)\), how many \(f_{N_i,e}(a)\) values would you need to recover \(a\)? (Assume that all \(N_i\)s are pairwise relatively prime)

Problem 2: Paillier Cryptosystem

The Paillier public key cryptosystem \((G, E, D)\) works as follows:

**Key generation** First \(G(1^k)\) picks two \(k\)-bit primes \(p\) and \(q\). Let \(n = pq\) and \(\alpha\) such that \(\alpha n \equiv 1 \mod \phi(n)\). Set \(PK = n\) and \(SK = \alpha\) and output \((PK, SK)\). (Note: This is similar to an RSA key pair, only here \(e = n\).)

**Encryption** To encrypt a message \(m\) where \(0 \leq m < n\), pick a random \(r \in \mathbb{Z}_n^*\) and treat it as an element of \(\mathbb{Z}_n^{*2}\). Then \(E(PK, m)\) outputs

\[ c = (1 + n)^m r^n \mod n^2 \]

where \((1 + n)\) is treated as an element of \(\mathbb{Z}_n^{*2}\).
Decryption To decrypt a ciphertext $c$, $D(PK, SK, c)$ computes

$$R = c^a \mod n$$
$$z = \frac{c}{R^n} \mod n^2$$
$$M = \frac{z - 1}{n}.$$ 

Then output $M$. Note that the first operation is modulo $n$, the second is modulo $n^2$, and the third is simply over the integers.

a. Prove that the Paillier cryptosystem is correct. In other words, show that $D(PK, SK, E(PK, m)) = m$.

b. One cool thing about the Paillier cryptosystem is that it is additively homomorphic. In other words, if $c_1 = E(PK, m_1)$ and $c_2 = E(PK, m_2)$, then

$$D(PK, SK, c_1c_2 \mod n^2) = m_1 + m_2 \mod n.$$ 

Prove this fact.

c. A similar property is called homomorphic scaling: if $c_1 = E(PK, m_1)$ and $c_2 = E(PK, m_2)$, then

$$D(PK, SK, c_1^{m_2} \mod n^2) = D(PK, SK, c_2^{m_1} \mod n^2) = m_1m_2 \mod n.$$ 

Prove that the Paillier cryptosystem scales homomorphically.

Problem 3: The Discrete Logarithm

In class we looked at the Discrete Log assumption on $\mathbb{Z}_p^*$. We also saw that there are ways to sample a $k$-bit prime $p$ such that you know the factorization of $(p - 1)$ and then it is easy to find the generator $g$ for $\mathbb{Z}_p^*$. Note that we must be careful when picking a prime $p$ to use since there are some bad choices we can make. Let $p = 2^k + 1$. Let $g$ be a generator of $\mathbb{Z}_p^*$. Show that given $y = g^x$ ($x$ unknown), it is easy to compute $x$.

Hint: We know that if $g$ is a generator of $\mathbb{Z}_p^*$, and $y = g^x \mod p$, then the LSB of $x$ is 0 iff $y$ is a square. We also know how to test whether an element of $\mathbb{Z}_p^*$ is a square. Generalize this observation. Work recursively, bit by bit.

Problem 4: Random Self-reducibility of RSA

(In this problem, when talking about multiplication of elements in $\mathbb{Z}_N^*$, we omit the implicit $\mod N$.)

a. Prove that all of the following distributions are identical for all RSA public keys $(N, e)$, and any $y \in \mathbb{Z}_N^*$:

$$D_1 = \{r \in \mathbb{Z}_N^* : r\}$$
$$D_2 = \{r \in \mathbb{Z}_N^* : r^e\}$$
$$D_3 = \{r \in \mathbb{Z}_N^* : r^gy\}$$

By identical distributions we mean that for all $x \in D_1 \cup D_2 \cup D_3$, $p_1(x) = p_2(x) = p_3(x)$, where $p_i(x) = \Pr[\hat{x} \in D_i : \hat{x} = x]$ for $i \in \{1, 2, 3\}$.

HW 5-2
b. Suppose that you are given an RSA public key \((N, e)\), and the values \(y, r, z \in \mathbb{Z}_N^*\) such that \(z^e = r^y\). How do you compute \(x\) such that \(x^e = y\)?

c. Suppose you are given an algorithm \(A\) and an RSA public key \((N, e)\) such that for some \(\epsilon > 0\)

\[
\Pr[r \leftarrow \mathbb{Z}_N^*; z \leftarrow A(N, e, r) : z^e = r] = \epsilon
\]

In other words, when given as input a random element of \(\mathbb{Z}_N^*\), \(A\) outputs its \(e\)th root with probability \(\epsilon\). Design an algorithm \(B\) such that for all \(y \in \mathbb{Z}_N^*\)

\[
\Pr[x \leftarrow B(N, e, y) : x^e = y] = \epsilon
\]

Your algorithm \(B\) must run \(A\) as a subroutine. In other words, your algorithm must take its input \(y\) and transform it to some random-looking \(r\), run \(A(N, e, r)\), obtain \(z\) such that \(z^e = r\), and from \(z\), find out \(x\).

d. The RSA assumption is that, given an RSA public key \((N, e) \leftarrow G_{RSA}(1^k)\) and a random \(y \in \mathbb{Z}_N^*\), it is hard to find \(x \in \mathbb{Z}_N^*\) such that \(x^e = y\). In experiment notation: for all probabilistic polynomial-time adversaries \(A\), there exists a negligible function \(\nu(k)\) such that

\[
\Pr[(N, e) \leftarrow G_{RSA}(1^k) ; y \leftarrow \mathbb{Z}_N^* ; x \leftarrow A(N, e, y) : y = x^e] = \nu(k)
\]

Show that the following assumption is equivalent to the RSA assumption (i.e., it holds if and only if the RSA assumption is true): for all probabilistic polynomial-time adversaries \(A\), there exists a negligible function \(\nu(k)\) such that

\[
\Pr[(N, e) \leftarrow G_{RSA}(1^k) ; y \leftarrow \text{QR}_N ; x \leftarrow A(N, e, y) : y = x^e] = \nu(k)
\]

*Hint:* You should use reductions.