T-79.159 Cryptography and Data Security **Spring 2003**
Tutorial 6
Tue 18.3.2003 12–14 T3 or Wed 19.3.2003 18–20 T1

1. Why is it not a good idea even for the same person to use two RSA keys with the same modulus?

2. In an RSA signature scheme implementation based on the Chinese remainder theorem, x is signed by computing $y_p = x^d \bmod p$ and $y_q = x^d \bmod q$ and then combining the results using the Chinese remainder theorem to get $y = x^d \pmod{n}$. Here $n = pq$, d is the secret exponent, and e is the public exponent.

   Suppose that a random bit error occurs when computing the signature of a message. Suppose furthermore that you know the message being signed and the resulting faulty signature. How can you compute the private key based on this information?

3. Consider the following simple public-key identification protocol.

   - A picks a random nonce N_A and sends $a = E_B(A, N_A)$ to B.
   - B picks a random nonce N_B and sends $c = E_A(N_A, N_B)$ to A.
   - A sends $z = E_B(N_B)$ to B.
   - B accepts if and only if $E_B^{-1}(z) = N_B$.

   What is wrong with this protocol?

4. Recall that the Schnorr identification scheme works as follows. Alice (A) is trying to identify herself to Bob (B).

   Let p and q be primes with $q|(p-1)$. Let $g \in \mathbb{Z}_p$ have order q (that is, $g \neq 1$ and $g^q = 1 \pmod{p}$). Let t be a security parameter, $2^t < q$. A chooses a secret exponent w and computes $h = g^w$. The values (g, h, p, q) are known by B.

   The actual protocol works as follows.

   - A picks a random $r \in \mathbb{Z}_q$, computes $a = g^r \bmod p$ and sends it to B.
   - B picks a random challenge $c \in \{0,1\}^t$ and sends it to A.
   - A computes $z = cw + r \pmod{q}$ and sends it to B.
   - B accepts if and only if $g^z = ah^c \bmod p$.

   Answer the following questions.

   (a) Why does the verification work?
   (b) Why is the arithmetic in the first step done modulo p and in the third step modulo q?
   (c) After the protocol run, does A know who B is?
   (d) Why is the protocol secure against passive attacks? (The intuitive idea is enough.)