CS255: Cryptography and Computer Security **Winter 2011**

Assignment #3

Due: Monday, Mar. 7, 2011. (in class)

Problem 1 Let's explore why in the RSA public key system each person has to be assigned a different modulus $N = pq$. Suppose we try to use the same modulus $N = pq$ for everyone. Each person is assigned a public exponent e_i and a private exponent d_i such that $e_i \cdot d_i = 1 \bmod \varphi(N)$. At first this appears to work fine: to encrypt a message to Bob, Alice computes $c = m^{e_{\text{bob}}}$ and sends c to Bob. An eavesdropper Eve, not knowing d_{bob} appears to be unable to decrypt c. Let's show that using e_{eve} and d_{eve} Eve can very easily decrypt c.

a. Show that given e_{eve} and d_{eve} Eve can obtain a multiple of $\varphi(N)$.

b. Show that given an integer k which is a multiple of $\varphi(N)$ Eve can factor the modulus N. Deduce that Eve can decrypt any RSA ciphertext encrypted using the modulus N intended for Alice or Bob.
Hint: Consider the sequence $g^k, g^{k/2}, g^{k/4}, \ldots g^{k/\tau(K)} \in \mathbb{Z}_N$ where g is random in $\mathbb{Z}_N$ and $\tau(k)$ is the largest power of 2 dividing k. Use the the left most element in this sequence which is not equal to ± 1 in $\mathbb{Z}_N$.

Problem 2. Time-space tradeoff. Let $f : X \to X$ be a one-way permutation. Show that one can build a table T of size B bytes ($B \ll |X|$) that enables an attacker to invert f in time $O(|X|/B)$. More precisely, construct an $O(|X|/B)$-time deterministic algorithm $\mathcal{A}$ that takes as input the table T and a $y \in X$, and outputs an $x \in X$ satisfying $f(x) = y$. This result suggests that the more memory the attacker has, the easier it becomes to invert functions.
Hint: Pick a random point $z \in X$ and compute the sequence

$$z_0 := z, \quad z_1 := f(z), \quad z_2 := f(f(z)), \quad z_3 := f(f(f(z))), \quad \ldots$$

Since f is a permutation, this sequence must come back to z at some point (i.e. there exists some $j > 0$ such that $z_j = z$). We call the resulting sequence $(z_0, z_1, \ldots, z_j)$ an f-cycle. Let $t := \lceil |X|/B \rceil$. Try storing $(z_0, z_t, z_{2t}, z_{3t}, \ldots)$ in memory. Use this table (or perhaps, several such tables) to invert an input $y \in X$ in time $O(t)$.

Problem 3 Commitment schemes. A commitment scheme enables Alice to commit a value x to Bob. The scheme is *secure* if the commitment does not reveal to Bob any information about the committed value x. At a later time Alice may *open* the commitment and convince Bob that the committed value is x. The commitment is *binding* if Alice cannot convince Bob that the committed value is some $x' \neq x$. Here is an example commitment scheme:

Public values: (1) a 1024 bit prime p, and (2) two elements g and h of $\mathbb{Z}_p^*$ of prime order q.

Commitment: To commit to an integer $x \in [0, q-1]$ Alice does the following: (1) she picks a random $r \in [0, q-1]$, (2) she computes $b = g^x \cdot h^r \bmod p$, and (3) she sends b to Bob as her commitment to x.

Open: To open the commitment Alice sends (x, r) to Bob. Bob verifies that $b = g^x \cdot h^r \bmod p$.

Show that this scheme is secure and binding.

a. To prove security show that b does not reveal any information to Bob about x. In other words, show that given b, the committed value can be any integer x' in $[0, q-1]$.
Hint: show that for any x' there exists a unique $r' \in [0, q-1]$ so that $b = g^{x'} h^{r'}$.

b. To prove the binding property show that if Alice can open the commitment as (x', r') where $x \neq x'$ then Alice can compute the discrete log of h base g. In other words, show that if Alice can find an (x', r') such that $b = g^{x'} h^{r'} \bmod p$ then she can find the discrete log of h base g. Recall that Alice also knows the (x, r) used to create b.

Problem 4 Threshold signatures. A company wants to institute a policy that two executives are needed to sign a contract. The process is as follows: a secretary sends the contract to both execs, they each sign and send their signature back to the secretary. The secretary then assembles the two signatures into a valid signature on the contract. Note that the two execs communicate with the secretary, but are not allowed to communicate with each other. One option is to give each exec a signature key and say that a signature is valid only if it contains valid signatures from both execs. In this question we develop a method that results in a shorter signature. Let (N, e) be the company's RSA public key and let d be the corresponding signing key.

a. Let d_1 be a random integer in $[1, \ldots, N]$ and let $d_2 = d - d_1$. Suppose we give d_1 to one exec and d_2 to the other. Explain how the secretary can interact with the execs to generate a signature under the company's RSA public key (N, e). The execs cannot communicate with one another and should keep their secrets to themselves.

b. Are both execs needed to generate a signature under (N, e), or is one execs sufficient? Briefly explain your answer.

c. Generalize the mechanism from part (a) so that any 2 out of 3 execs can generate a signature under (N, e), but no single exec can do it.

Problem 5. Access control and file sharing using RSA. In this problem $N = pq$ is some RSA modulus. All arithmetic operations are done modulo N.

a. Suppose we have a file system containing n files. Let $e_1, \ldots, e_n$ be relatively prime integers that are also relatively prime to $\varphi(N)$, i.e. $\gcd(e_i, e_j) = \gcd(e_i, \varphi(N)) = 1$ for all $i \neq j$. The integers $e_1, \ldots, e_n$ are public. Choose a random $r \in \mathbb{Z}_N^*$ and suppose each file F_i is encrypted using the key $\mathsf{key}_i := r^{1/e_i}$.

Now, let $S_u \subseteq \{1, \ldots, n\}$ and set $b = \prod_{i \in S_u} e_i$. Suppose user u is given $K_u = r^{1/b}$. Show that user u can decrypt any file $i \in S_u$. That is, show how user u using K_u can compute any key key_i for $i \in S_u$.

With this mechanisn, every user u_j can be given a key K_{u_j} enabling it to access exactly those files to which it has access permission.

b. Next we need to show that user u, who has K_u, cannot construct a key key_i for $i \notin S_u$. To do so we first consider a simpler problem. Let d_1, d_2 be two integers relatively prime to $\varphi(N)$ and relatively prime to each other. Suppose there is an efficient algorithm $\mathcal{A}$ such that $\mathcal{A}(r, r^{1/d_1}) = r^{1/d_2}$ for all $r \in \mathbb{Z}_N^*$. In other words, given the d_1'th root of $r \in \mathbb{Z}_N^*$ algorithm $\mathcal{A}$ is able to compute the d_2'th root of r. Show that there is an efficient algorithm $\mathcal{B}$ to compute d_2'th roots in $\mathbb{Z}_N^*$. That is, $\mathcal{B}(x) = x^{1/d_2}$ for all $x \in \mathbb{Z}_N^*$. Algorithm $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine.

c. Show using part (b) that user u cannot obtain the key key_i for any $i \notin S_u$ assuming that computing e'th roots modulo N is hard for any e such that $\gcd(e, \varphi(N)) = 1$. (the contra-positive of this statement should follow from (b) directly).

Problem 6. Time lock. Our goal in this question is to build a mechanism by which Alice can encrypt a secret S that can be decrypted only after a certain amount of time has passed (e.g. a week, a year, a 100 years).

a. Alice's first solution is as follows. Let (E, D) be a symmetric cipher built from AES. Alice chooses a random AES key k and publishes (C, T) where $C \leftarrow E(k, S)$ and T contains all but t bits of k. Then by exhaustive search the attacker can decrypt C and recover S in time 2^t. By tuning t Alice can choose the time it will take for S to be revealed.

Unfortunately, this approach does not work. Briefly explain how an attacker can recover S in time $2^t/L$ for some L of the attacker's choosing.
Hint: think parallel processing.

b. Alice then remembers that she read somewhere that the best algorithm for computing g^x requires $O(\log x)$ sequential multiplications and that parallel processing cannot speed this up much. She decides to use the following approach. First, she generates two primes p and q and sets $n \leftarrow pq$. Next, she chooses a random g in $\mathbb{Z}_n^*$. Finally, she publishes (n, g, C, t) where

$$C \leftarrow S + g^{(2^{(2^t)})} \quad \in \mathbb{Z}_n$$

Describe an algorithm that enables anyone to recover S from (n, g, C) using 2^t modular multiplications. Hence, by tuning t Alice can make the puzzle take as long as she wants, even if the attacker mounts your attack from part (a).

c. Finally, show that Alice need not spend time 2^t herself to prepare the puzzle. Show that Alice can use her knowledge of $\varphi(n)$ to construct C using only $O(t)$ modular multiplications.

d. After setting this up Alice wondered if she could use a prime p in place of the RSA modulus n in the system above. Will the resulting time-lock system remain secure if n is replaced by p? If so, explain why. If not, describe an attack.