**CS255: Cryptography and Computer Security** **Winter 2012**

# Assignment #3

Due: Monday, Mar. 12, 2012. (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 $d_{\text{bob}}$ appears to be unable to decrypt $c$. Let's show that using $e_{\text{eve}}$ and $d_{\text{eve}}$ Eve can very easily decrypt $c$.

**a.** Show that given $e_{\text{eve}}$ and $d_{\text{eve}}$ 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 $\pm 1$ 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.** In class we showed a collision resistant hash function from the discrete-log problem. Here let's do the same, but from the RSA problem. Let $n$ be a random RSA modulus, $e$ a prime relatively prime to $\varphi(n)$, and $u$ random in $\mathbb{Z}_n^*$. Show that the function

$$H_{n,u,e} : \mathbb{Z}_n^* \times \{0, \ldots, e-1\} \to \mathbb{Z}_n^* \qquad \text{defined by} \qquad H_{n,u,e}(x, y) := x^e u^y \quad \in \mathbb{Z}_n$$

is collision resistant assuming that taking $e$'th roots modulo $n$ is hard.

Suppose $\mathcal{A}$ is an algorithm that takes $n, u$ as input and outputs a collision for $H_{n,u,e}(\cdot, \cdot)$. Your goal is to construct an algorithm $\mathcal{B}$ for computing $e$'th roots modulo $n$.

**a.** Your algorithm $\mathcal{B}$ takes random $n, u$ as input and should output $u^{1/e}$. First, show how to use $\mathcal{A}$ to construct $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}$ such that $a^e = u^b$ and $0 \neq |b| < e$.

**b.** Clearly $a^{1/b}$ is an $e$'th root of $u$ (since $(a^{1/b})^e = u$), but unfortunately for $\mathcal{B}$, it cannot compute roots in $\mathbb{Z}_n$. Nevertheless, show how $\mathcal{B}$ can compute $a^{1/b}$. This will complete your description of algorithm $\mathcal{B}$ and prove that a collision finder can be used to compute $e$'th roots in $\mathbb{Z}_n^*$.
**Hint:** since $e$ is prime and $0 \neq |b| < e$ we know that $b$ and $e$ are relatively prime. Hence, there are integers $s, t$ so that $bs + et = 1$. Use $a, u, s, t$ to find the $e$'th root of $u$.

**c.** Show that if we extend the domain of the function to $\mathbb{Z}_n^* \times \{0, \ldots, e\}$ then the function is no longer collision resistant.

**Problem 5** Recall that a simple RSA signature $S = H(M)^d \bmod N$ is computed by first computing $S_1 = H(M)^d \bmod p$ and $S_2 = H(M)^d \bmod q$. The signature $S$ is then found by combining $S_1$ and $S_2$ using the Chinese Remainder Theorem (CRT). Now, suppose a Certificate Authority (CA) is about to sign a certain certificate $C$. While the CA is computing $S_1 = H(C)^d \bmod p$, a glitch on the CA's machine causes it to produce the wrong value $\tilde{S}_1$ which is not equal to $S_1$. The CA computes $S_2 = H(C)^d \bmod q$ correctly. Clearly the resulting signature $\tilde{S}$ is invalid. The CA then proceeds to publish the newly generated certificate with the invalid signature $\tilde{S}$.

**a.** Show that any person who obtains the certificate $C$ along with the invalid signature $\tilde{S}$ is able to factor the CA's modulus.
Hint: Use the fact that $\tilde{S}^e = H(C) \bmod q$. Here $e$ is the public verification exponent.

**b.** Suggest some method by which the CA can defend itself against this danger.

**Problem 6.** 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 $\mathsf{key}_i$ 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 $\mathsf{key}_i$ 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 $\mathsf{key}_i$ 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).