CS195-7: Introduction to Cryptography	DUE: October 1, 2002
Problem Set 3-4	
Instructor: Anna Lysyanskaya	**** Your name here ****

Problem 1

In this problem, we show that if for a given RSA public key (n, e), the RSA function $f_{n,e}(x) = x^e \bmod n$ can be inverted on an ϵ fraction of $\mathbb{Z}_n^*$ in time t, then $f_{n,e}^{-1}(y)$ can be computed for any y in expected time t/ϵ.

(a) Show that for all RSA public keys (n, e), for all values $y \in \mathbb{Z}_n^*$, the following experiments produce identically distributed outcomes (recall that $x \leftarrow X$ notation means that x was chosen uniformly at random from set X):

Experiment 1: Pick $r \leftarrow \mathbb{Z}_n^*$, and output r.
Experiment 2: Pick $r \leftarrow \mathbb{Z}_n^*$, and output r^e.
Experiment 3: Pick $r \leftarrow \mathbb{Z}_n^*$, and output $r^e y$.

Solution:

All the three experiments amount to choosing a value uniformly at random from $\mathbb{Z}_n^*$. To see that, let us estimate the probability, for a value $u \in \mathbb{Z}_n^*$, that u is chosen.

In Experiment 1:

$$\Pr[u] = \Pr_r[r = u] = 1/|\mathbb{Z}_n^*|$$

In Experiment 2, note that if $u \in \mathbb{Z}_n^*$, and e is relatively prime to $\phi(n)$, then there exists a unique element v such that $v^e = u \bmod n$. This is simply because, as we saw in class, RSA is a permutation. (By $u^{1/e}$, let us denote such an element v.) Then, for Experiment 2:

$$\Pr[u] = \Pr_r[r^e = u] = \Pr_r[r = u^{1/e}] = 1/|\mathbb{Z}_n^*|$$

In Experiment 3, note that if $y, u \in \mathbb{Z}_n^*$, and e is relatively prime to $\phi(n)$, then there exists a unique element w such that $w^e = u/y$. Again, this is because RSA is a permutation, and u/y is just some element of $\mathbb{Z}_n^*$. Therefore, in Experiment 3:

$$\Pr[u] = \Pr_r[r^e y = u] = \Pr_r[r^e = u/y] = \Pr_r[r = (u/y)^{1/e}] = 1/|\mathbb{Z}_n^*|$$

(b) Let (n, e) be fixed. Suppose that algorithm A has the following property: there exists a set $Y \subseteq \mathbb{Z}_n^*$, $|Y| = \phi(n)\epsilon$, such that for all $y \in Y$, $A(y)^e = y$, i.e., A inverts RSA for an ϵ fraction of $\mathbb{Z}_n^*$. Suppose that A's running time is t steps. (And assume that it always halts after t steps, even if its input is $y \notin Y$.) Show that

$$\Pr[r \leftarrow \mathbb{Z}_n^*; v = A(r) \ : \ v^e = r] \geq \epsilon$$

.

Solution:

It is clear that if $r \in Y$ is chosen, then $A(r) = r^{1/e}$. Since r is chosen uniformly at random from $\mathbb{Z}_n^*$, then with probability $|Y|/|\mathbb{Z}_n^*| = \epsilon$, $r \in Y$.

(c) Assume algorithm A as descibed in part (b). Give an algorithm that runs in expected time t/ϵ (up to a multiplicative factor that is independent on (n, e)) and inverts $f_{n,e}$ for all $y \in \mathbb{Z}_n^*$.

Solution:

The algorithm will repeat the following until the value $x \in \mathbb{Z}_n^*$ such that $x^e = y \bmod n$ is found:

- choose a value r uniformly at random
- let $z = r^e y \bmod n$
- run $A(z)$ to obtain a value w
- check if $w^e = z$:
  - If so, output $x = w/r$. Note that

$$x^e = (w/r)^e = w^e/r^e = z/r^e = yr^e/r^e = y$$

  - Otherwise, repeat.

Problem 2

Suppose that Alice uses the same password x to log into three different machines A, B, and C. Each machine uses RSA with exponent 3 for password authentication. That is to say, each machine has generated its own RSA modulus, but happens to use $e = 3$. By n_X let us denote the modulus of machine $X = A, B, C$. The value $y_X = x^3 \bmod n_X$ is stored in a world-readable file at machines $X = A, B, C$. Give an algorithm that computes Alice's password x using the values n_A, n_B, n_C and y_A, y_B, y_C.

Solution:

First, let us argue that the moduli n_A, n_B and n_C are relatively prime with all but negligible probability. Note that each modulus was generated using the RSA key generation procedure. Recall that this procedure picks two prime integers of length k uniformly at random. The event that the moduli are not relatively prime is the event that, when the procedure is run three times, two of the prime numbers it picks coincide. By the prime number theorem, there are approximately $2^k/k$ prime numbers of length k. Therefore the probability of a collision is negligible.

Now, suppose that the values n_A, n_B, and n_C are relatively prime. We will use the Chinese remainder theorem (CRT) to retrieve the value $y = x^3$ *over the integers*. Since we are given the values $y_X = y \bmod n_X$ for $X = A, B, C$, and the moduli n_A, n_B, and n_C are relatively prime, the CRT gives us y. To retrieve x, all that is needed is to compute the cube root of y over the integers. This can be done, for example, by binary search.

Problem 3

Recall the GM definition of security for one bit for public-key cryptosystems that you saw in lecture:

Definition. A cryptosystem (G, E, D) is GM-secure for one bit if: (1) it is a faithful cryptosystem where decryption retrieves the encrypted message for all choices of the public and secret keys; and (2) it is secure: for all probabilistic polynomial-time families of adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that

$$\Pr[(PK, SK) \leftarrow G(1^k); b \leftarrow \{0,1\}; c \leftarrow E(PK, b); b' \leftarrow A_k(PK, c) \ : \ b = b'] \leq 1/2 + \nu(k)$$

Note that in the above definition, the bit b is chosen at random, and then encrypted, and then the adversary cannot tell what it was better than by random guessing.

(a) Do we get an equivalent definition of a secure cryptosystem if we replace condition (2) of the definition above by the following:

For all probabilistic polynomial-time families of adversaries $\{A_k\}$, for $b = 0, 1$, there exists a negligible function $\nu(k)$ such that

$$\Pr[(PK, SK) \leftarrow G(1^k); c \leftarrow E(PK, b); b' \leftarrow A_k(PK, c) \ : \ b = b'] \leq 1/2 + \nu(k)$$

Solution:

The definition given in this problem is unachievable. Consider the trivial adversary who always outputs "0." Such an adversary will always give the correct answer when a "0" is encrypted, for any cryptosystem. Therefore, assuming the existence of GM-secure cryptosystems, the two definitions are not equivalent. If we do not make the assumption that GM-secure cryptosystems exist, then nothing can be said about the equivalence, since two non-existent objects can be considered equivalent.

(b) (Extra credit — do this problem last!) Let the notation $b \leftarrow^q \{0,1\}$ denote that b is a *biased* bit: it is 0 with probability q, and 1 with probability $1-q$, for $1/2 \leq q < 1$.

For what values of q do we get an equivalent definition of a secure cryptosystem if we replace condition (2) of the definition above by the following requirement:

For all probabilistic polynomial-time families of adversaries $\{A_k\}$, there exists a negligible function $\nu(k)$ such that

$$\Pr[(PK, SK) \leftarrow G(1^k); b \leftarrow^q \{0,1\}; c \leftarrow E(PK, b); b' \leftarrow A_k(PK, c) \ : \ b = b'] \leq q + \nu(k)$$

Solution:

If we assume that GM-secure cryptosystems (G, E, D) exists, then I can exhibit a cryptosystem that is q-secure for any $q < \frac{1}{2}$ but not GM-secure. Therefore, the two notions are not equivalent for $q \neq 1/2$, unless no GM-secure cryptosystems exist (in which case, it is easy to see that q-secure cryptosystems do not exist either and so the two objects are equivalent simply because of their non-existence.)

Let me outline the structure of this argument. Given a GM-secure cryptosystem, I will show how to construct one that is not GM-secure, but at the same time still q-GM secure.

In order to show that it is q-GM secure I will assume that it is not and arrive at a statement that contradicts the GM security of the original cryptosystem (G, E, D). This is a rather tricky proof outline, so please pause for a second and read this paragraph again to make sure you understand the direction in which I am going to proceed with this proof.

Next, I will give some intuition for this argument. The crucial problem with q-GM security is that it only prevents adversaries which are very successful. Basically, an adversary would only violate q-GM security for small q if its probability of being correct was essentially 1. In this argument, we not only show there is some q for which q-GM security does not imply GM-security, but actually we show a single cryptosystem which proves that for all $0 < q < 1/2$, q-GM security does not imply GM-security. The basic idea of the construction is that we will leak a polynomially vanishing amount of information about the encrypted bit, which is enough to violate GM-security but not enough to make the scheme vulnerable under q-GM security.

Let q be given, and suppose we are given a cryptosystem (G, E, D) which is GM-secure. Then in particular, if we restrict the message space to two messages, call them 0 and 1, it remains GM secure. In our new cryptosystem, $G'(1^k)$ just returns the same result as $G(1^k)$. Now let us make the following modification to the encryption algorithm: in our new cryptosystem, the encryption algorithm E' works as follows. Let $\circ$ denote concatenation. Define $E'(PK, 0) = E(PK, 0) \circ X$ and $E'(PK, 1) = E(PK, 1) \circ Y$, where X is a random variable which is 0 with probability $1/2 + \frac{1}{k}$, and 1 otherwise, and Y's distribution is biased the other way around.

Clearly this new cryptosystem is not GM secure, since the last bit of the encryption is going to leak information about the plaintext.

Now we claim that this new cryptosystem (G', E', D') is still q-GM secure. Suppose it is not. Then there exists an adversary $\{A_n\}$ such that for infinitely many k the following holds:

$$Pr[PK \leftarrow G(1^k); b \leftarrow \{q : 0, 1\}; x \leftarrow E'(PK, b); b \leftarrow A_k(PK, c) : b = b'] \geq 1 - q + \epsilon$$

where ϵ is non-negligible, e.g. $1/p(n)$ for some polynomial p.

In particular this holds for some k such that $\frac{1/2-1/k}{1/2+1/k} > \frac{q}{1-q}$. This is possible because $\frac{1/2-1/k}{1/2+1/k} = \frac{k-2}{k+2} \rightarrow 1$. Note that if $q < 1/2$ then $\frac{q}{1-q} < 1$. However, if $q = 1/2$, $\frac{q}{1-q} = 1$; that is to say, this argument only works for $q < 1/2$. This is a good thing to notice as a sanity check - after all, 1/2-GM security *is* GM-security, so we'd better not disprove their equivalence!

First of all, let us describe our "maximum likelihood"[1] strategy. Then we will argue that in fact the adversary $\{A_n\}$ is using something more clever than the maximum likelihood strategy. Our maximum likelihood strategy will always output 1, and here is why: Every time we see a ciphertext that ends with a 1, we output 1, because the conditional probability that the message was 1 given that ciphertext ends in a 1, is greater than the conditional

[1]If we were talking about unconditional security, we would have solved the problem already just by proving that the maximum likelihood strategy cannot do better than $1 - q$. However, our setting is computational, and therefore we cannot use a probability theorem like that, we have to arrive at a contradiction to a computational assumption.

probability that the message was a 0 (it's obvious, so won't formally prove this). But every time we see a ciphertext that ends with a 0 we will still guess 1, because the conditional probability that the message was 1 given that the ciphertext ends in a 0, is greater than the conditional probability that the message was 0 given that the ciphertext ends in a 0.

Let us verify that this is indeed the case. Let us denote the last bit of $E'(b)$ as c.

$$\Pr[b=0|c=0] = \Pr[c=0|b=0]\Pr[b=0]/\Pr[c=0] = \frac{(1/2+1/k)q}{(1/2+1/k)q+(1/2-1/k)(1-q)}$$

$$\Pr[b=1|c=0] = \Pr[c=0|b=1]\Pr[b=1]/\Pr[c=1] = \frac{(1/2-1/k)(1-q)}{(1/2+1/k)q+(1/2-1/k)(1-q)}$$

and so the statement above follows because we picked n sufficiently large.)

This strategy will give us only probability $1-q$ of a success, since we guess 1 no matter what, and 1 occurs only with probability $1-q$. So then the adversary must be using a better strategy. Then the adversary must sometimes be guessing 0 when we would be guessing 1 (otherwise there is simply no room for improvement since we are guessing 1 everywhere!). Moreover, the adversary must be correct when it guesses a 0 with at least an inverse polynomial probability ϵ, otherwise where would its advantage come from? Therefore, at least one of the statements below is true:

1. At least half of the adversary's advantage, $\epsilon/2$ is coming from the adversary being correct because it guesses 0 when $c=0$.

2. At least half of the adversary's advantage, $\epsilon/2$ is coming from the adversary being correct because it guesses 0 when $c=1$.

Assume for now that (1) is the true statement. Then, we can use this adversary to break the encryption scheme (G, E, D). If we are given x, we simply run the q-GM adversary on $x \circ 0$. Let $m(k) = \frac{(1/2+1/k)q}{(1/2+1/k)q+(1/2-1/k)(1-q)}$. Now we know that

$$Pr[(PK, SK) \leftarrow G(1^k); b \leftarrow \{m(k) : 0, 1\}; x \leftarrow E(PK, b); b' \leftarrow A_k(PK, x \circ 0)] \geq 1 - m(k) + \epsilon/2$$

since this is what we mean when we say that half the adversary's advantage is coming from the case when it guesses 0 when $c = 0$. Call this probability $\Pr[S_{mGM}(k, A)]$.[2] Now by the argument we used to show that q-GM security implies GM-security, we know that $\Pr[S_{GM}(k, A)]$, the probability that A is successful if we just append 0 to x and hand it off to A, is at least $1/2 + \epsilon/4(1-m)$, so we have broken the GM-security of (G, E, D).

If at least half the adversary's advantage comes from the case when the adversary guesses 0 when $c = 1$, we do the same thing, except we ask about $x \circ 1$, and the argument is exactly the same.

Thus, we are done. We have shown that (G', E', D') is not GM-secure, and furthermore that if it is not q-GM secure, then (G, E, D) is not GM-secure. Therefore, q-GM security does not imply GM-security.

[2]The overlap of this notation with $S_{qGM}(k, A)$ make sense since this is really just the same as the even for m-GM security.