6.875: Solutions for Homework 3

Yevgeniy Dodis

Main Problem

See the paper handed out in class and the handout with a simple proof that GM-security implies Semantic security.

Extra Problem 1: Trapdoor Permutations

Try to formalize the notion of a *trapdoor permutation*, whose informal description is the following:

> *bijective function that is easy to compute, hard to invert but there exists a secret (necessarily hard to find!) such that the function is easy to invert if the secret is known.*

(It might help to think whether RSA or exponentiation modulo a prime are good candidates.)

Solution: I will give a slightly less general definition than the one we often need in practice (for example, the family will have a function for each input length k, and sometimes we need to exclude certain bad input lengths, so we need an appropriate sampling algorithm, etc). But this will clarify the point of the problem without removing any concetual ideas. The important point is the fact that we shoudl allow a different trapdoor for each input length (we can't expect one secret can help us invert a function for *any* x). Thus, despite great similarity with one-way functions (that can be defined as a single function that is hard to invert on each input length), we really define a "probabilistic ensamble" of functions that can be efficiently generated (together with the trapdoor), but which are hard to invert from scratch. Here is what we obtain.

Definition 1 *A family of* trapdoor permutations *is a triple* (G, A, B) *where*

1. *G is a function generation algorithm, i.e. a PPT s.t. $G(1^k) \to (p_k, s_k)$. (p_k describes a permutation over $\{0,1\}^k$, s_k is the trapdoor).*
2. *A is a polynomial time algorithm computing the function, i.e. $A(x, p_k)$ is a permutation over $\{0,1\}^k$ when x ranges through $\{0,1\}^k$.*
3. *It is hard to invert A, i.e. for any PPT C*

   $$\Pr((p_k, s_k) \leftarrow G(1^k), x \leftarrow \{0,1\}^k, y \leftarrow A(x, p_k) : x \leftarrow C(y, p_k)) = negl(k)$$

   (as usual, for all $a > 0$, less than $1/k^a$ for s.l. k)
4. *Easy to invert A with a trapdoor, where B is the inverting algorithm, i.e. B is polytime and $B(y, s_k) = x$ for any k, any possible pair (p_k, s_k), any y in $\{0,1\}^k$, where $y = A(x, p_k)$.*

It is not hard to model RSA as a family of trapdoor permutations: $p_k = (n, e)$, $s_k = (e^{-1} \bmod \varphi(n))$, where n is a product of 2 random primes of length k and e is a random element relatively prime to $\varphi(n)$. Exponentiation modulo a prime is unknown to be a trapdoor family: while it is believed to be a family of one-way functions, nobody knows how to find discrete logs even with some short trapdoor.

Extra Problem 2: Encrypting the Secret Key

Assume that there exists s GM-secure system $\mathcal{C}$ (under three-pass, for example). Show that there exists a GM-secure system $\mathcal{C}'$ that can be completely broken once the encryption of the secret key is known.

Solution: The point of this exercise is simple. GM-security guarantees us that the system is secure if the adversary $\mathcal{A}$ can only try to break it "on its own" (in polynomial time). In other words, we give $\mathcal{A}$ a random cryptosystem and ask him to "break" it (in this case, by distinguishing encryption of m_0 from that of m_1). However, once we "help" the adversary by giving him for free something that he *provably* could not have generated on his own (like the encryption of the secret key), GM-security leaves us no guarantee on the safety of the system.

For this particular problem, we simply modify $\mathcal{C}$ by letting it encrypt everything as before except if outputs the secret key in the clear when the message happens to coincide with the secret key. Intuitively, GM-security is preserved both in one- and three-pass scenarios. Informally, in the three-pass scenario there is only negligible chance that a random secret key will coincide with one of 2 fixed messages since otherwise we could have broken $\mathcal{C}$ itself. And in the one-pass scenario, the message finder has a negligible chance to output a secret key as one of the challenge messages since otherwise we could have completely broken $\mathcal{C}$ again. Therefore, up to a negligible advantage, $\mathcal{C}'$ is as secure as $\mathcal{C}$ is while $\mathcal{C}'$ is totally broken when someone leaks the encryption of the secret key.

Let's do this slightly more formally for the three-pass; I omit several technical and unimportant details below, but even what I show illustrates that some conceptually simple ideas might be non-trivial to write formally. So don't get scared, I did not expect you to write this, but what to show what a formal proof means.

Before doing the proof, we need one simple observation: for any message m, probabilistic encryption algorithm $\tilde{E}$ and decryption algorithm $\tilde{D}$, we can check by sampling whether $\tilde{D}$ is a valid decryption algorithm for m w.r.t. $\tilde{E}$, i.e. that $\Pr(\tilde{D}(\tilde{E}(m)) = m) = 1 - negl(k)$, where the probability is only over the *coins of the encryption algorithm* $\tilde{E}$ (and $\tilde{D}$).

Let $\mathcal{C}$ be GM-secure in three-pass. Given a circuit A, let $desc(A)$ be the description of A. Here is C' on input 1^k outputting a pair (E', D'):

1. Let $(E, D) \leftarrow \mathcal{C}(1^k)$.
2. Given ciphertext $y = b \circ c$, $b \in \{0, 1\}$, define D' as follows: if $b = 0$, $D'(y) = c$, else $D'(y) = D(c)$.
3. Define $E'(m) = 0 \circ m$ if $m = desc(D')$, and $E'(m) = 1 \circ E(m)$ otherwise.
4. Output constructed (E', D').

It is straightforward to see that $\mathcal{C}$ is a valid cryptosystem if $\mathcal{C}$ is (i.e. for all m it correctly decrypts the encryption of m). It is also clear that $\mathcal{C}'$ is completely broken when the encryption of $desc(D')$ (equal to $desc(D')$ itself) is known. On the other hand, we show that if $\mathcal{C}'$ is not GM-secure, then so is $\mathcal{C}$. Assume A' violates GM-security of $\mathcal{C}'$, i.e. for some m_0, m_1 we have (as usual, for infinitely many k)

$$\Pr(A'(E', m_0, m_1, \alpha') = m \mid m \leftarrow \{m_0, m_1\}; (E', D') \leftarrow \mathcal{C}'(1^k);\ \alpha' \leftarrow E'(m)) > \frac{1}{2} + k^{-d}.$$

We now define $A(E, m_0, m_1, \alpha)$ that would violate GM-security of C. Let $\tilde{E}(m) = 1 \circ E(m)$, i.e. we for $m \neq desc(D')$, $\tilde{E}(m) = E'(m)$. Algorithm A simply runs $A'(\tilde{E}, m_0, m_1, 1 \circ \alpha)$ with the following simple modification. Whenever A' queries $\tilde{E}$ on a message m, interpret m as a $desc(\tilde{D})$ for some $\tilde{D}$. If this is not possible, continue to run A'. Otherwise, test whether $\tilde{D}$ is a good decryption algorithm w.r.t. $\tilde{E}$ of both m_0 and m_1. If not, continue to run A'. Else, output m_0 if $\tilde{D}(1 \circ \alpha) = m_0$, and output m_1 if $\tilde{D}(1 \circ \alpha) = m_1$.

Intuitively, we rely on the following fact that I prove at the end. It has to do with the fact that the cryptosystem is chosen *after* the messages m_0 and m_1.

- Assume we choose a random cryptosystem $(E, D) \leftarrow \mathcal{C}(1^k)$. Then with high probability both m_0 and m_1 are different from $desc(D')$, where D' and E' are as described in construction of $\mathcal{C}'$, so that $\tilde{E}(m) = E'(m) = 1 \circ E(m)$ when $m \in \{m_0, m_1\}$.

Provided this is true, we are done as when $\alpha = E(m_b)$ for random b, then $\alpha' = 1 \circ \alpha = E'(m_b)$, so we are running A' on what we are supposed to run it to get a non-negligible advantage. The only catch is that E' is different from $\tilde{E}$ when $m = desc(D')$. There are two ways out. One way is to argue that if A' has a non-negligible chance to call A' on $desc(D')$ then $\mathcal{C}$ is not GM-secure. But an easier way is what we did. We explicitly test each m to see if it is a description of some $\tilde{D}$ that works great on m_0 and m_1. If yes, we simply stop and use it by decrypting the challenge $1 \circ \alpha$. Even though we are *not* guaranteed that $\tilde{D} = D'$ (even though we know that if $\tilde{D} = D'$ we *will pass the test!*), we do not need it since α' is a *random* encryption of m_b w.r.t. $\tilde{E}$ which is identical to E' on m_0 and m_1. So any good decryptor is good for us giving us a distinguisher that is correct with probability close to 1. So either A' never calls $\tilde{E}$ on $desc(D')$ in which case we get the same advantage of at least k^{-d} as A' did, or (which we can argue will never happen but we do not need it), we get advantage of almost $1/2 \gg k^{-d}$. So A indeed violates GM-security of $\mathcal{C}$.

It remains to prove the claim that I made. It is simple. The probability is 0 if m_0, m_1 are not descriptions of some circuit $\tilde{D}$. If any of them is a description of some circuit $\tilde{D}$ and a random $(E, D) \leftarrow \mathcal{C}(1^k)$ has a good probability that $\tilde{D} = D'$, we can use this *fixed* $\tilde{D}$ as our universal guess for D. This allows us to distinguish with non-negligible probability encryptions of m_0 and m_1, for any m_0 and m_1, a contradiction.