CS255 Homework 1

Rory MacQueen SUNet ID: macqueen

February 5, 2014

Problem 0

The one time pad encryption of the message "attack at dusk" would be: 09e1c5f70a65ac519458e7f15a37

Problem 1

A. Alice can first encrypt the message with the secret key of Bob K_{AB}, and then take that output and encrypt it with the secret key of Charlie K_{AC}. When the message is received, it must be decrypted first by Charlie with K_{AC} and then by Bob with K_{AB}. Neither Bob nor Charlie can decrypt the message on his own, since it requires decryption by both of them. So we have to send two random IVs in the clear - one for the first level of encryption (with K_{AB}) and one for the second level (with K_{AC}). Any two recipients and decrypt one of these three cipher blocks and hence can retrieve the key K_r with which they can decrypt the whole message!

B. Choose a random message encryption key, called K_r. Create three cipher text blocks: each cipher text block consists of K_r encrypted with a different pair of the recipients keys. So we have c_{BC}, which is K_r encrypted twice, with K_{AB} and K_{AC}, c_{BD} is encrypted twice, with K_{AB} and K_{AD}, and c_{CD}, encrypted twice with K_{AC} and K_{AD}.

C. The length of the header would be $\binom{n}{k}$

Problem 2

A. We progress down the binary tree and use keys as follows: Each node at a specific level in the tree has two children n_1 and n_2. Only one of those children is the subtree which contains the compromised leaf node r. Without loss of generality, suppose it is n_1. Then we know that the other node, n_2 is clean - that is, all of its children are still valid players. We then include in the header the content key K, encrypted with the key associated with n_2. We then progress to n_1 and repeat - we ask which of its children has beneath it the compromised node r and send the key associated with the other child. Since we are only sending 1 cipher text per level of the binary tree, and we do not send anything for the lowest level (since that will be the compromised node r itself) we send exactly $1 + \log_2(n) - 1 = \log_2(n)$ cipertexts of the content-key K.

B. We can show that only $O(k \log n)$ keys will be used to ensure that all players except the pirated ones can decrypt a movie. Consider the binary tree in the problem setup, where each node in the tree corresponds to a key and each leaf to a DVD player. A given player is embedded with the $1 + \log_2 n$ keys which lie on the path from the root to its leaf. All nodes start out colored white. Now, some of these players (i.e. leaves) have been compromised by hackers. Color the leaf nodes, corresponding to hacked players, black. Now recurse up the tree and, at each layer, color the parent of a black node black until you reach the root. Once we have finished, we know that a node that is colored black is a node who has a hacked player as one of its descendants. Now, at each level of the tree, we must, in the worst case, give out a key to the sibling of every black node (since this would be the only way to ensure that the clean player was able to decrypt but a hacked player was unable to decrypt). Recognize that since it is a binary tree, each black node has exactly one sibling. Therefore, we must (in the worst case) give out one key for every black node. We know that the lowest layer of the tree has k black nodes (since there are k hacked players, and the lowest layer corresponds to players). We also know that at layer $i-1$ there are at most the same number of black nodes as at layer i because each black node has at most one parent ($i = 1$ is the root layer and $i = \log_2 n$ is the leaf layer). In other words, the number of black nodes in a given layer can only decrease as you move from the leaf layer to the root. So we can bound the number of black nodes per layer at k. Since there are $\log_2 n$ layers to a tree, we can bound the total number of black nodes in the tree at

$O(k(1 + \log_2 n))$

$= O(k \log_2 n)$

Since we showed above that at worse case we must give out exactly one key for every black node (i.e. to the one sibling) we also know that the number of keys we must distribute is bounded by

$O(k \log_2 n)$

Problem 3

A. We know that for each value i, each player receives either k_0 or k_1 in their key K_l. Therefore, we know that when the cipher text is broadcast, every player must have either k_0 or k_1 for the random value i (which is known because it is sent in the clear). Since the message m is encrypted with both k_0 and k_1, each player will be able to decrypt one of those, and so will be able to retrieve the message m.

B. We recover one bit of j at a time as follows: for each bit from 0 to $n-1$, we construct a cipher text where we use a different message for the k_0 encryption and the k_1 encryption. So, for each bit i, our cipher text C_i will be:

$C_i = (i, E(k_{i,0}, m_x), E(k_{i,1}, m_y))$

where $m_x \neq m_y$. Now, we look at the decrypted text spit out by the blackbox P. If it is m_x, then we know that the bit at position i is 0; if it is m_y, then we know that the bit at position i is 1. Since we do this for each bit of j, in total we send n total cipher texts.

C. Since this pirate player has two keys (I am going to use the notation K_a and K_b for the two keys since i is getting overloaded), it can be constructed as follows: for each encrypted message that it receives, it has the choice to either use K_a or K_b to decrypt. It can make this decision randomly and decrypt the appropriate message depending on the result of this choice - that is, if it chooses to use the K_a key then it decrypts $k_{i,b}$ message where b is the i^{th} bit on K_a. Since the output will be a mix of decrypted messages from both keys, one will not be able to tell which keys are being used. Of course, there is a problem if $a \oplus b$ is a power of two, since, in that case, it means that K_a and K_b only differ by one bit, so there will be no real choice in what they output (except for that one different bit), and their results will uniquely identify the two compromised players.

Problem 4

A. Advantages of the adversaries:

$\mathcal{A}_1 : |1 - 1| = 0$
$\mathcal{A}_2 : |0.5 - 0.5| = 0$
$\mathcal{A}_3 : |0.5 - 0| = 0.5$
$\mathcal{A}_4 : |0.5 - 1| = 0.5$
$\mathcal{A}_5 : |(0.5 * (1) + 0.5 * (0.5)) - 0.5(1)| = 0.25$

B. The maximum possible advantage is 0.5. This is because EXP(0) is genuinely random and returns both heads and tails with probability 0.5. Any distinguishing characteristics between the two experiments come from EXP(0) since it is the one that introduces any variability. Since it returns each value with 50/50 probability, an adversary cannot possibly do better than that in distinguishing the two.

Problem 5

A. Adversary does the following: send message m_1 to challenger. Challenger sends back c_1, which includes the IV for that message IV_1 (sent in the clear) and the IV for the next message IV_2 (send as the last ciphertext block of c_1). Adversary now constructs m_2 as:

$m_2 = m_1 \oplus IV_1 \oplus IV_2$

We know that the challenger is going to first XOR the message m_2 with IV_2 so the encryption will proceed as:

$c_2 = E(k, m_2 \oplus IV_2) = E(k, (m_1 \oplus IV_1 \oplus IV_2) \oplus IV_2) = E(k, m_1 \oplus IV_1)$

The two IV_2 XOR operations will cancel each other out, and so the encryption will run as if it was using IV_1 as the nonce and so the resultant cipher text c_2 will be equal to c_1 since it is the same message and the same IV. So, we send that m_2 as one of the messages in the CPA game, and the other message m_3 can just be all zeros. We get two cipher text responses back, c_2 and c_3. If one of them matches c_1, we say that the it was our m_2 that the challenger decided to return - if it does not match c_1, we say that it was our other m_3. Our adversary will have advantage close to one; it will not be exactly one since there is the small possibility that our third message m_3 happens to get encrypted to the same cipher text c_1. But other than that small collision possibility, our adversary will always win this CPA game.

B. A simple fix to the problem in part a) would be to always use a random IV!

C. Our attack outlined in part a) was contingent only on our adversary knowing the IV for the next message. So we can repeat the attack in part a) and simply guess the IV by picking a random number in the space of the block size, which is 2^n big. So we have a $1/2^n$ probability of selecting the correct IV when we use that and our attack from part a). So our advantage is now close to $1/2^n$ rather than close to 1, for the same reason as outlined above.

Problem 6

A. We can show that F_1 is not secure by defining an adversary that has non-negligible advantage. The adversary, A, will act as follows: if the response $f(x_i)$ to his submission ends in 0, output 0, else output 1. The advantage of this adversary will be:

Advantage $= |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]|$
$= |0 - 0.5| = 0.5$ which is not negligible. Therefore F_1 is NOT secure.

B. To show that F_2 is secure, we can prove the contra-positive. Assume we have a distinguisher $\mathcal{A}$ that breaks F_2. From that, we can construct a distinguisher $\mathcal{B}$ which breaks F as follows. Distinguisher $\mathcal{B}$ wraps itself around $\mathcal{A}$. $\mathcal{A}$ outputs a submission, x_i to the challenger. $\mathcal{B}$ then makes its own $x'_i = x_i \oplus 1^n$ and submits that to the challenger, which is using F as its PRF. In that way we essentially mimic the total functionality of F_2 (since $F_2(k, x) = F(k, x \oplus 1^n)$) , which we know that $\mathcal{A}$ can break. When the response $f(x'_i)$ comes back, we feed it to $\mathcal{A}$, who can then determine whether it came from a truly random function or from our PRF. We then have $\mathcal{B}$ output whatever $\mathcal{A}$ outputs. Our advantage for $\mathcal{B}$ against F will be exactly the advantage $\mathcal{A}$ has against F_2. Hence, in total, our distinguisher $\mathcal{B}$ can break F.

C. We construct F_3 as follows: Let $k_3 = k||b$ where $k \in \{0,1\}^n$ and $b \in \{0,1\}$. We set $F_3(k_3, x)$ to be the same as $F(k, x)$ for all $x \neq 0^n$. We define the output of $F_3(k_3, 0^n)$ to be the output of $F(k, 0^n)$ with the last bit then set to b. Now, we can see that, assuming the key k_3 is secret, F_3 is secure. We know this because F is given as a secure PRF, and F_3 only differs from F by one bit and that one bit is unknown to the adversary so there is no possible way the adversary could gain an advantage from it. Put another way, if an adversary were able to break F_3, that same adversary would be able to break F since the only difference is something that is unknown to it. However, we can also see that if an adversary did know the last bit of k_3, namely b, then he could distinguish it from a random function.

What this adversary would do is issue to the challenger the message $x_i = 0^n$. When the response, $f(0^n)$ was sent back, the adversary could check the last bit of $f(0^n)$ and see if it matches the b that it knows. If it does, output 1, else output 0. Using this policy, an adversary's advantage is:

Adv = |Pr[EXP(0) = 1] - Pr[EXP(1) = 1]| = $|1 - 0.5| = 0.5$

This is clearly a non-negligible value and so the PRF is no longer secure.