CS255: Introduction to Cryptography **Winter 2012**

Assignment #1: Solutions

Answer 0. *(2 points)* It makes more sense to compress the data and then encrypt the result. Compression takes advantage of redundancies in data, while encryption makes the data look like a random string of bits. Because the ciphertext looks like a random string of bits (which has high entropy), it can't be compressed efficiently.

■

Answer 1. (a) *(2 points)* Alice can first pick a random key K to encrypt the message M, and then she encrypts the key K with both K_{AB} and K_{AC}. In other words, Alice does the following:

$$\underbrace{E_{K_{AB}}[E_{K_{AC}}(K)]}_{\text{header}} \| E_K(M)$$

(b) *(1 point)* Similar to Part (a), Alice does the following:

$$\underbrace{E_{K_{AB}}[E_{K_{AC}}(K)] \| E_{K_{AB}}[E_{K_{AD}}(K)] \| E_{K_{AC}}[E_{K_{AD}}(K)]}_{\text{header}} \| E_K(M)$$

(c) *(1 point)* It is straightforward to generalize the above scheme to the case where any k out of the n recipients can decrypt, but any $k-1$ cannot. The length of the header would be $\binom{n}{k} = n!/k!(n-k)!$. Note that we have the inequalities:

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$$

This shows that the proposed solution scales poorly.

■

Answer 2. (a) *(1 point)* In CBC mode, decryption is as follows:

$$P_i = D_K(C_i) \oplus C_{i-1}, \quad C_0 = \text{IV}.$$

Blocks $\ell/2$ and $\ell/2+1$ will be affected by a corrupted aiphertext $C_{\ell/2}$. Therefore, 2 blocks will be decrypted incorrectly.

(b) *(1 point)* In randomized counter mode decryption, we regenerate the keystream using the randomized counter and incrementing it for every subsequent block. Therefore, only block $\ell/2$ will be decrypted incorrectly.

■

Answer 3. (a) *(2 points)* A simple way to construct an efficient adversary is as follows: Send the packet containing all 0s to the server to be encrypted. Since we are playing the CPA security game, the server will then send us the encryption of this packet, which we call c_0. This packet will contain exactly the encryption of $0 \oplus$ IV, where IV is the initialization vector chosen by the encryption system. Next, suppose we take the last block of c_0 and XOR it with the all zeroes block to get p_1. Sending across p_1 to the server to be encrypted results in the following: $E(p_1) = E(c_0' \oplus c_0' \oplus 0) = E(0)$, where c_0' refers to the last block of c_0. This means our output is just the encryption of 0 and therefore, our next IV is 0.

In the next phase of the game, we use two challenge plaintexts: the all zeroes plaintext referred to as m_0 and a randomly chosen string $m_1 \neq m_0$. Note that the encryption of m_0 is $E(0)$, which is already known. The encryption of m_1 will be, with sufficiently high probability, something else. If the challenge string returned is equal to $E(0)$, then we output the bit 0, i.e., guess that the original plaintext was m_0. Otherwise, we guess 1. We win the CPA game with a probability 1. This attack works with any fixed input that will constitute m_0. We do not necessarily require the all zeroes string.

(b) *(1 point)* Re-encrypt the last block of the packet (with a new key derived from the original key in a secure manner) before using it as the IV. In order to predict the IV, the adversary would have to be able to guess the encryption of the ciphertext (which is distributed approximately uniformly at random) under some unknown key with a non-trivial success probability. An adversary that could do this would break the semantic security of AES, so we assume the method works.

(c) *(2 points)* Our CPA adversary $\mathscr{A}$ interacts with the challenger as follows. Send the query $q^* = 0^n$, the all-zeroes string. $\mathscr{A}$ receives $C^* = E_{\text{CBC}}(k, 0^n)$ that includes the IV chosen (let's call it IV^*) and the encrypted message block. For the semantic security challenge, $\mathscr{A}$ chooses $M_0 = 0^n$ and $M_1 = 1^n$ (or equivalently any other message). It receives C_b and must guess b. If $C_b = C^*$, output 0, else output 1.

To analyze the advantage of $\mathscr{A}$, in Exp_0, if the IV chosen matches IV^* (which happens with probability $\frac{1}{2^n}$), then $\mathscr{A}$ output 0. In all other cases, it outputs 1. Thus $\Pr[\mathscr{A}(\mathsf{Exp}_0) = 1] = 1 - \frac{1}{2^n}$. In Exp_1, $\mathscr{A}$ never outputs 0, thus $\Pr[\mathscr{A}(\mathsf{Exp}_1) = 1] = 1$. The advantage is: $|\Pr[\mathscr{A}(\mathsf{Exp}_0) = 1] - \Pr[\mathscr{A}(\mathsf{Exp}_1) = 1]| = \frac{1}{2^n}$ as required.

■

Answer 4. *(3 points)* $F(k,x)$ is *not* a secure PRF. Given two pairs of inputs (x_1, x_2), (x_3, x_4) such that $x_1 \oplus x_2 = x_3 \oplus x_4$, it is easy to check that $F(k,x_1) \oplus F(k,x_2) = F(k,x_3) \oplus F(k,x_4)$. To distinguish F from a random function, the adversary simply queries the Challenger with 4 inputs as above and checks to see if the relation is satisfied. For a random function, this will be true with probability $\frac{1}{2^n}$, whereas for F, this is true with probability 1. Thus, such an adversary, in polynomial time, has non-negligible advantage in distinguishing F from a random function. In the example given, it is easy to see that:

$$F(k,0000) \oplus F(k,1111) = F(k,0011) \oplus F(k,1100) = k[1] \oplus k[2] \oplus k[3] \oplus k[4].$$

■

Answer 5. (a) *(2 points)* Let nodes $v_1, \ldots, v_{\log_2 n}$ be the nodes along the path from the root of the tree to leaf node number r. Since the tree is binary every node v_i for $i = 2, \ldots, \log_2 n$ has exactly one sibling. Let $u_2, u_3, \ldots, u_{\log_2 n}$ be the siblings of nodes $v_2, \ldots, v_{\log_2 n}$. Let $K_2, \ldots, K_{\log_2 n}$ be the AES keys associated with the nodes $u_2, \ldots, u_{\log_2 n}$. The header will contain the encryption of the content-key K under all keys

$\mathscr{K} = \{K_2, \ldots, K_{\log_2 n}\}$. Clearly player number r cannot decrypt the movie since it does not have any of the keys in $\mathscr{K}$. However, any other player $t \neq r$ can decrypt the movie. To see this, consider the path from the root of the binary tree to leaf node t. Let u be the highest node along this path that does not appear on the path from the root to leaf node r. Then the key K_u associated with node u is in the set $\mathscr{K}$ (since the sibling of u is a on the path to r). Furthermore, player t contains the key K_u and it can therefore obtain the content-key and then the movie. Hence, player r cannot play the movie, but all other players can.

(b) *(2 points)* For $i = 1, \ldots, k$ let U_i be the set of consecutive leaves in the range $[r_i + 1, r_{i+1} - 1]$ (we are assuming $r_0 = 0$ and $r_{k+1} = n + 1$). For an internal node v in the binary tree let S_v be the set of leaves in the subtree rooted at v. We say that a set of leaves U_i is exactly covered by internal nodes $v_1, \ldots, v_b$ if $U_i = \cup_{j=1}^{b} S_{v_j}$. We show below that each set U_i can be exactly covered by a set of at most $2 \cdot \log_2 n$ internal nodes. This means that to exactly cover all sets $U_0, \ldots, U_r$ we need at most $c = 2(r+1)\log_2 n$ internal nodes. Let $v_1, \ldots, v_c$ be the internal nodes needed to cover all of $U_0, \ldots, U_r$. If we encrypt the content-key using the keys K_v associated with each of these nodes then all players other than those in R can recover the content-key while the players in R cannot. This shows that using header containing at most $2(r+1)\log_2 n$ ciphertexts we can revoke all players in R without affective any of the other players.

It remains to show that given a set of consecutive leaves U in some range $[a, b]$ it is possible to exactly cover U using at most $2 \cdot \log_2 n$ internal nodes. Let u_1 be the highest node in the tree so that the subtree rooted at u_1 has leaf a as its left most leaf. Let v_1 be the highest node in the tree so that the subtree rooted at v_1 has leaf b as its right most leaf. Now, for $i = 2, \ldots, \log_2 n$ define u_i to be the right sibling of the parent of u_{i-1} (the right sibling of a node w is the node at the same height as w which is immediately on the right of w). Similarly, define v_i to be the left sibling of the parent of v_{i-1}. Let j be the smallest value so that $u_j = v_j$ or that u_j, v_j are adjacent siblings in the tree. Then it is easy to see that $u_1, \ldots, u_j, v_j, \ldots, v_1$ is an exact cover of $[a, b]$. This covering set contains at most $2 \cdot \log_2 n$ nodes as required.

■

Answer 6. (a) *(2 points)* For every $i \in \{1, \ldots, 32\}$, a particular player ℓ has either key $k_{i,0}$ or key $k_{i,1}$, depending on b_i—the i-th bit in the binary representation of ℓ. Given a ciphertext (i, c_0, c_1), player ℓ can obtain the message m by decrypting c_{b_i}.

(b) *(1 point)* Assume that the pirate decoder has the key K_j embedded. Suppose we feed the ciphertext $C_i = (\ i,\ E(k_{i,0}, m_0),\ E(k_{i,1}, m_1)\ )$ into the player, with m_0 and m_1 corresponding to observably different content. If the player plays content m_0 (resp. m_1) then it must possess the key $k_{i,0}$ (resp. $k_{i,1}$). Hence, by observing the result, we learn the i-th bit of index j. By feeding n different ciphertexts, we can retrieve the complete index j bit by bit.

(c) *(2 points)* Let $i_{n-1} i_{n-2} \ldots i_0$ and $j_{n-1} j_{n-2} \ldots j_0$ be the binary representations of i and j respectively. Let a be an index where the two binary representation differ—i.e. $a \in \{0, 1, \ldots, n-1\}$ such that $i_a \neq j_a$. We build the pirate player P by replacing the a-th key in player i with the a-th key from player j. In other words, player P has the following key embedded:

$$K_P = \left(k_{i_0}, k_{i_1}, \ldots, k_{i_{a-1}}, k_{j_a}, k_{i_{a+1}}, \ldots, k_{i_{n-1}}\right)$$

Tracing algorithm from part (b) produces the index $\ell = i_{n-1} \ldots i_{a+1} j_a i_{a-1} \ldots i_0$. Since $i_a \neq j_a$ index ℓ is obviously different from i, since $i \oplus j$ is not a power of 2, index ℓ has to be different from j as well.

■