CSEP590 : Applied Cryptography: Homework 2

Karuna Sagar Krishna

April 22, 2023

Task 1a

We are given that Mr. Cipher uses AES 128 bit CTR mode to encrypt messages using the same secret key everyday. We are also told that Mr. Cipher only sends either M1 or M2. Since message M1 is shorter than M2, the cipher text corresponding to M1 would be shorter than M2. This is simply because CTR mode encrypts each 16 byte plain text block to produce a 16 byte cipher text block. Though CTR mode provides IND-CPA security the length of the plain text is not hidden. Hence by looking at the cipher text length we can clearly say if the plain text was M1 or M2.

We are also told that Mr. Cipher prepends the current data (8 bytes) to the message before encrypting them. Since CTR mode offers IND-CPA security i.e. confidentiality of the message and we don't know the secret key, it is not possible for us to decrypt. However, we know that the current date is used and the message asks to meet tomorrow at the rendezvous point. So if we know that the cipher text was sent out today by Mr. Cipher, we can be sure to find Mr. Cipher at rendezvous point tomorrow.

So, in summary if we know on which date Mr. Cipher sent the encrypted message and looking at the length of the cipher text, we can predict with certainty on which date Mr. Cipher will show up at the rendezvous location.

Task 1b

Mr. Cipher can protect from the above attack by ensuring both M1 and M2 are of the same length. With this the length of cipher texts for both messages would be the same and we will not be able to distinguish them as described above. Given CTR mode provides IND-CPA security the cipher text looks like random strings. To achieve this Mr. Cipher can simply pad the shorter message M1 with 00 bytes so that after padding $|M1| = |M2|$. We could use PKCS#7 to pad instead but this would have to be applied to both messages so that they both can be padded to same number of 16 byte blocks.

This is sufficient to prevent the above attack since we cannot use distinguish the messages based on the length of the cipher text. Further, since don't know the secret key and CTR mode provide IND-CPA we cannot distinguish the

cipher texts from a random string in reasonable amount of time. As we saw in class, the distinguishing advantage is $\leq \frac{t}{2^{126}} + \frac{L^2}{2^{142}}$. To keep the distinguishing advantage low (2^{-48}), we can safely encrypt 256GB data and then rotate the secret key.

Task 1c

In this case, the plain texts M1 and M2 are of the exact same length (similar to the above suggestion). Both M1 and M2 are 32 bytes long and we are given 48 bytes long cipher text. This implies the first 16 bytes of cipher text is initialization vector (IV) and then we have 2 blocks of 16 bytes that correspond to the plain text. In CTR mode, each plain text block is xor-ed with a random string that generated by running block cipher. In other words, the encryption for each plain text block is independent; and hence we can parallelize CTR.

Messages M1 and M2 differ in the last word. We also notice that in the cipher text, the last 2 blocks of 16 bytes are identical. It might be tempting to conclude that the plain text is M1 since last two 16 byte blocks are the same in both cipher text and plain text. To confirm this, lets say the cipher text is represented by 16 byte blocks as $C[0]$, $C[1]$ and $C[2]$ and the plain texts are represented as $M1[1]$, $M1[2]$ and $M2[1]$, $M2[2]$. So we have $IV = C[0]$, $C[1] = C[2]$ and $M1[1] = M1[2]$. As shown in class lectures, the intermediate random string generated by block cipher can be named as $P[1]$ and $P[2]$. Assuming that given cipher text decrypts to M1, then $P[1] = P[2]$ since $C[0] = C[1]$ implies $M[1] \oplus P[1] = M[2] \oplus P[2]$. But this is only possible if the inputs to block cipher are the same. This is a contradiction since $IV + 1 \neq IV + 2$. Hence, the other option is that the cipher text decrypts to M2.

Though we are able to identify the plain text from the given cipher text, I don't think IND-CPA security of CTR mode is violated. Note, IND-CPA security essentially says that there is a low probability that we can distinguish cipher text when choosing plain texts. In this case, the block cipher for inputs $IV + 1$ and $IV + 2$ produced random string $P[1]$ and $P[2]$ such that though $P[1] \neq P[2]$ and $M[1] \neq M[2]$, we have $C[1] = C[2]$. The probability for this to happen is $\frac{1}{2^{128}-1}$ since if we fix $P[1]$, then $P[2] = M[2] \oplus P[1] \oplus M[1]$ and hence there is only one way to choose $P[2]$ from $2^{128} - 1$ possible values (since inputs to block cipher are different $P[2]$ cannot be equal to $P[1]$ and we can think of block ciphers as pseudo random permutations). This is a low probability that we happen to hit upon in the given cipher text. So the IND-CPA security still holds in general.

Task 2a

The encryption scheme proposed is vulnerable to the penguin attack and hence does not offer IND-CPA security. Consider the following attack:

```
procedure PenguinAttack
    X = {0,1}^128
    M = X||X
    C = encrypt(M)
    if C[1] = C[2] then
        return 1
    else
        return 0
```

The above attack is the penguin attack, where we encrypt a message consisting of 16 byte blocks that are repeated. We can clearly see that the proposed encryption scheme produces cipher text of length 3 blocks i.e. $R||Y||Y$ for some $Y = \{0,1\}^{128}$. This is exactly the penguin attack that we discussed in class. So the distinguishing advantage is very high $1 - 2^{-128}$ because $Pr(A(S_0) = 1) = 1$ (we can certainly identify the proposed encryption with the above attack) and $Pr(A(S_1) = 1) = 2^{-128}$ (once we fix the first 16 bytes, the second 16 byte has to match; so we have 2^{128} options out of 2^{256}). The running time of the attack is $O(n)$ and it only encrypts 256 bits. Hence this encryption scheme does not offer IND-CPA security.

Task 2b

Yes, I think the proposal from Bob to fix the encryption does provide IND-CPA security. Firstly, the above attack is not possible on Bob's encryption scheme i.e. even if $M[i] = M[j] : i \neq j$, $C[i] \neq C[j]$ because $M[i] + i \neq M[j] + j$ and $R \oplus (M[i] + i) \neq R \oplus (M[j] + j)$, so the inputs to AES is different hence it produces different ouputs. Further since AES is a secure block cipher it produces random looking outputs for different inputs. This makes it harder for an attacker to distinguish the cipher text from a truly random generator i.e. the distinguishing advantage is low.

However, this encryption scheme requires padding i.e. $M[l]$ could be less than 16 bytes and we need to pad it to 16 bytes so that we can xor with R and AES block cipher requires 16 byte block. This might expose the encryption scheme to padding oracle attacks which is outside the scope of this question.

Task 3

We have been given two encryption schemes $E1$ and $E2$. One of these schemes is IND-CPA secure while the other is not. Both these encryption schemes encrypt n bit message using k bit key to produce l bit cipher text. Consider the following encryption scheme which uses both $E1$ and $E2$ to encrypt n bit message using $2k$ bit key to produce $2l$ bit cipher text.

procedure Gen
    $K1 = E1.Gen()$
    $K2 = E2.Gen()$
    return $K1||K2$

procedure $Enc(K, M)$
    $K1||K2 = K$
    $R1 = \{0,1\}^n$
    $R2 = R1 \oplus M$
    $C1 = E1.Enc(K1, R1)$
    $C2 = E2.Enc(K2, R2)$
    return $C1||C2$

procedure $Dec(K, C)$
    $K1||K2 = K$
    $C1||C2 = C$
    $R1 = E1.Dec(K1, C1)$
    $R2 = E2.Dec(K2, C2)$
    return $R1 \oplus R2$

The encryption algorithm breaks up the key into two k bit keys each used for $E1$ and $E2$ respectively. Next, we calculate two n bit string $R1$ and $R2$ such that $R1 \oplus R2 = M$ i.e. we break up the message into two parts using xor. We then encrypt each part of this message with $E1$ and $E2$ and concatenate them as the final cipher text of length $2l$.

The decryption algorithm performs the inverse of the encryption algorithm described above. It breaks up the cipher text into two l bit cipher texts. It then decrypts each part of the cipher text using $E1$ and $E2$ respectively which is then xor-ed to produce the n bit plain text.

Hence, the above encryption scheme is correct because $Dec(K, Enc(K, M)) = M$.

To prove the above encryption scheme is IND-CPA secure, lets look at the encryption algorithm. The n bit plain text message is broken into two n bit strings using xor. This is accomplished by first generating a random n bit string $R1$ and then calculating $R2 = R1 \oplus M$. Notice, the masking trick in action here i.e. $R1$ is a random string independent of M so $R2$ is random and independent of M. This implies that having revealing either $R1$ or $R2$ does not reveal any information about M. Next, we encrypt $R1$ and $R2$ using $E1$ and $E2$ respectively. Since either $E1$ or $E2$ is IND-CPA secure, the confidentiality of $C1$ or $C2$ is promised. In other words, though we don't have the confidentiality guarantees of either $C1$ or $C2$, as mentioned previously it does not leak any

information about M. So as long as one of the underlying encryption schemes is IND-CPA secure, the above encryption scheme is also IND-CPA secure.

Task 4a

PKCS#7 padding is valid if the last byte X of the padded message is repeated X times at the end of the padded message. Clearly, $X = 01$ is valid padding in all cases. Further say, the last but one byte is Y. If $X \neq Y$, then we can get never get a valid padding. If $X = Y$ and if this byte is repeated Y times, then we have a valid padding. So, there are two cases where we can get a valid padding.

For each of the above cases, we can manipulate the last byte of second last block of cipher text to get a valid padding.

Task 4b

In the context of padding oracle attack, we want to manipulate the last byte of second last block of cipher text so that we end up with a padding of 01. We can verify this by changing the second last byte of the second last block of cipher text and passing this to the oracle. If we get back a valid answer, then we know that the padding is 01 since the second last byte did not have any effect on the validity of the padding, otherwise we would have received an invalid answer from the oracle.

Task 4c

In class we recovered the last byte of the plain text by manipulating the cipher text to get a valid padding ending in 01. To recover the second last byte of the plain text, we can manipulate the second last byte of the second last block of cipher text to get a valid padding ending in 0202. The second last byte of the plain text is recovered by the same xor formula we saw in class i.e. we xor the manipulated byte value with 02 and then xor this with the original cipher text byte value. This can be repeated to recover other bytes of the last block of plain text.

Given the way CBC mode works, we can throw away the last block of the cipher text and repeat the above process to recover another block of plain text. This is because the decryption of the last block does not affect the decryption of the second last block. Further, though the original cipher text has some padding in the last block, we are not interested in that original padding itself. Instead we are manipulating the cipher text so that we can get a valid padding of our choice as described above.

To recover each byte, we potentially iterate over 256 choices for that byte. And for each of these choices we need to call the oracle to verify if we have a valid padding. Further, as explained above there could be atmost 2 choices that

can lead to a valid padding in which case we distinguish by calling the oracle again to disambiguate. So in total, we might call the oracle atmost 256+2 times to recover each byte of the plain text.

Task 4d

The python implementation is attached in Gradescope. Below is the decryption of $hw2-1.ctxt$ file ("121" appears in the middle of the decrypted text which seems odd)

> It was a long time before anyone spoke. Out of the corner of his eye Phouchg could see the sea of tense expectant faces down in the square outside. "We're going to get lynched aren't we?" he whispered. "It was a tough assignment," said Deep Thought mildly. "Forty-two!" yelled Loonquawl. "Is that all you've got to show for seven and a half million years' work?" "I checked it very thoroughly," said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is." "But it was the Great Question! The Ultimate Question of Life, the Universe and Everything!" howled Loonquawl. "Yes," said Deep Thought with the air of one who suffers fools gladly, "but what actually is it?" A slow stupefied silence crept over the men as they stared at the computer and then at each other. "Well, you know, it's just Everything . . . Everything . . . " offered Phouchg weakly. "Exactly!" said Deep Thought. "So once you do know what the question actually is, you'll know what the answer means." "Oh terrific," muttered Phouchg flinging aside his notebook and wiping away a tiny tear. "Look, alright, alright," said Loonquawl, "can you just please tell us the Question?" "The Ultimate Question?" "Yes!" "Of Life, the Universe, and Everything?" "Yes!" Deep Thought pondered this for a moment. "Tricky," he said. 121 "But can you do it?" cried Loonquawl. Deep Thought pondered this for another long moment. Finally: "No," he said firmly. Both men collapsed on to their chairs in despair. "But I'll tell you who can," said Deep Thought. They both looked up sharply. "Who?" "Tell us!" Suddenly Arthur began to feel his apparently non-existent scalp begin to crawl as he found himself moving slowly but inexorably forward towards the console, but it was only a dramatic zoom on the part of whoever had made the recording he assumed. "I speak of none other than the computer that is to come after me," intoned Deep Thought, his voice regaining its accustomed declamatory tones. "A computer whose merest operational parameters I am not worthy to calculate { and yet I will design it for you. A computer which can calculate the Question to the Ultimate Answer, a computer of such infinite and subtle complexity that organic life itself shall form part of its operational matrix.

```
And you yourselves shall take on new forms and go down into the
computer to navigate its ten-million-year program! Yes! I shall design
this computer for you. And I shall name it also unto you. And it shall
be called . . . The Earth."  Phouchg gaped at Deep Thought.  "What a
dull name," he said and great incisions appeared down the length of
his body. Loonquawl too suddenly sustained horrific gashed from
nowhere.  The Computer console blotched and cracked, the walls
flickered and crumbled and the room crashed upwards into its own
ceiling . . .  Slartibartfast was standing in front of Arthur holding
the two wires.  "End of the tape," he explained.
```

Below is the decryption of $hw2-2.ctxt$ file

```
Hi, how are you?
```