Danmarks tekniske universitet

01410 Cryptology

Cryptology 1

Noah Grænge Surel
s215783

Christopher Zwinge
s204459

Tobias Collin
s183713

29. april 2022

Exercise 1.1.1

In this exercise we will show that it is necessary for the padding algorithm to pad the messages with at least one byte or up to n/8 bytes. To show that it is necessary we consider the situation where we can append at most n/8 -1 bytes or in other words a situation where message won't have a padding if it's length is a multiple of n.
We show this by constructing two messages m1 and m2, we start by having the two messages contain the exact same message. The messages both have the length 8*l and we define l as (n-1)/8, they are therefore 1 byte smaller than a multiple of n and the padding algorithm will add 1 byte of information to them if they were put through the algorithm. We then choose to change m2 by appending 00000001. This changes m2's length to be n/8 and therefore the padding algorithm will no longer add anything as it would have to pad with n/8 bytes to get to a new multiple of n. We now choose to put both messages through the padding algorithm and the results are the following.

`Before Padding:`

$$m2 = m1||00000001$$

`After Padding:`

$$m2 = m1$$

The message become the same therefore indistinguishable as the padding algorithm adds 1 in 8 bit to m1 according to the padding algorithm. Since these two messages are now identical there is no way to know how much of the messages are padding.

Exercise 1.1.2

In this exercise we describe a person in the middle attack that allow us to learn the length of the message Alice sends to Bob when they use AES in CBC mode with the PKCS #5 padding. The messages sent from Alice to Bob will look to us in the following form.

`First Message:`

$$(iv||C1)$$

`Second Message:`

$$(iv||C1||C2)$$

To learn how long the message being sent from Alice to Bob we use the fact that they are using CBC mode as CBC xor's the message with the previous ciphertext before it is encrypted and after it is decrypted. We will also use the fact that if the padding is wrong for a message the sender is asked to resend the message. In CBC mode if we change a single bit in ciphertext C1 and call it C1' and then give message (iv||C1') to Bob, the message will probably be accepted, but we can then use the fact that Alice will xor the next message with C1 and Bob will xor it with C1* to make sure that there is a 1 bit difference in between the original message and the decrypted message. If we want to know the length of the second message we can simply change a bit in the second to last byte of C1 this will be called C1' and bob will

receive the following.

$$(iv||C1'||C2)$$

Since there is a difference in the second to last byte after the decryption, if Bob asks Alice to resend the message we know that we changed the padding and that the message at least has two bytes of padding. If the message goes through we know that there was only a single byte of padding. We can repeat this by resending the message as with C1' replace by C1" where we have changed only a single bit in the third last byte and we can keep repeating this process until the message is accepted by Bob. The message then has the lenght of n-l where n amount of bytes in C2 and l is the amount of times we have sent the message to Bob.

Exercise 1.2.1

From a 1 bit to 1 bit function the number zero can either become a zero or a one. The same goes for the number 1, it can either become a zero or an one.

From this we can set up a function:

$$f(x) = x \texttt{ Meaning it stays the same} \tag{1}$$

$$f(x) = y \texttt{ Meaning it changes} \tag{2}$$

Since the transformation can either become the same or a different value. From here we can prove that these 2 functions are affine, by using the equality given.

First left side of the equation.

$$f(x) \oplus f(y) = x \oplus y$$

Then the right side of the equation

$$f(x \oplus a) \oplus f(y \oplus a) = (x \oplus a) \oplus (y \oplus a) = x \oplus y$$

As can be seen from when the value does not change the equation is the same. Now we do it for when the values change as seen in equation (2).

Again first the left side of the equation.

$$f(x) \oplus f(y) = y \oplus x$$

Then the right side:

$$f(x \oplus a) \oplus f(y \oplus a) = (y \oplus a) \oplus (x \oplus a) = x \oplus y$$

With this we have proven that for a 1 bit to 1 bit is affine.

Exercise 1.2.2

We tested several functions to check if they were affine in 3 bit and the first function we found that wasn't affine is incrementing the number by one.
By incrementing the number by one we mean the following: The binary increase of the given number in

the function by one.
Example:

$$f(000) = 001$$
$$f(011) = 100$$
$$f(111) = 000$$

To test this we go through the equality which has to be true for something to be affine.

First again the left side of the equation given:

$$f(000) \oplus f(001) = 001 \oplus 010$$

Then the right side, where the value of a is given to 111.

$$f(000 \oplus 111) \oplus f(001 \oplus 111) = f(111) \oplus f(110) = 000 \oplus 111$$

As can be seen these 2 are not equal and therefore the function can be said to not be affine. Since an instance has been found where it isn't.
From here we can prove that the function is indeed invertible by taking the same values and decrementing them.
Left side of the equation:

$$f'(001) \oplus f'(010) = 000 \oplus (010)$$

Right side of the equation:

$$f'(000 \oplus 111) \oplus f'(111 \oplus 111) = f'(111) \oplus f'(000) = 000 \oplus 001$$

Therefore the function is invertible.

Exercise 1.3

We have a modified one-time pad where the key is defined as:

$$k \leftarrow \mathbb{K} = (\{0,1\}^n \backslash \{0^n\}) \tag{3}$$

So for this OTP it's possible for the key to be every combination of 0 and 1 except for a full zero key. The reason to remove the full zero key is so the ciphertext can't be the same as the message. We then need to find a ciphertext and a message to show that it doesn't fulfill perfect secrecy.

$$\Pr_{k \leftarrow \mathbb{K}}[c = m \oplus k] = 0 \tag{4}$$

Since the key no longer is able to be a full zero key, then if both the ciphertext and the message only consisted of zeroes, then there wouldn't be a possible key for this example.

As explained in Lemma 9.3 in the book[1].

$$\#\mathbb{K} \geq \#\mathbb{C} \geq \#\mathbb{P} \tag{5}$$

[1] Nigel Smart. *Cryptography Made Simple*, p 166.

- $\#\mathbb{K}$ denotes the size of possible keys
- $\#\mathbb{C}$ denotes the size of possible ciphertexts
- $\#\mathbb{P}$ denotes the size of possible plaintexts

Since both ciphertext and message have the same size of possibilities, $(\{0,1\}^n)$ whereas the size of possible keys is the same but excluding the possibility of a full zero key. then the size of $\mathbb{K}$ is no longer greater or equal to the size of $\mathbb{C}$, and therefore this one-time pad no longer fulfill perfect secrecy.