Cryptology 1 - Homework 1

Asbjørn Olling - s163615 Christoffer Luplau - s204498

Exercise 1.1

1. Prove that for any padding algorithm P where at most $n/8 - 1$ bytes are appended, there are two messages m_1 and m_2 that are mapped to the same message m' by P. For simplicity, you can consider messages of at most $n/8$ bytes, i.e. messages that fit into one block.

It is necessary, since a message which already has a length which is a multiple of the block size n, would otherwise be left unchanged. Because of this property, it is possible to craft an input message which maps to another valid input message, which maps to itself.

For example, if we choose a block size of $n = 32$ and two messages (shown as arrays of bytes here) $m_1 = \texttt{[2,2]}$ and $m_2 = \texttt{[2,2,2,2]}$, m_1 would result in $\hat{l} = 2$ which would make the padded message `[2,2,2,2]`, exactly that of m_2

$$m_1 = \texttt{[2,2]} \qquad \mapsto \texttt{[2,2,2,2]} \tag{1}$$
$$m_2 = \texttt{[2,2,2,2]} \qquad \mapsto \texttt{[2,2,2,2]} \tag{2}$$

The two messages will result in the same outputs, creating ambiguity for the padding-removal algorithm on the decryption side.

2. Describe an attack that allows you to learn the length of a message that, say, Alice sends to Bob.

1. Intercept the message Alice sends to Bob.
2. Modify the last byte in the second-to-last block.
3. Forward the malicious message to Bob.
4. If the message is invalid Bob will request that Alice resends the message with valid padding. Repeat where we modify the byte before the one we just modified.
5. If the message is valid Bob must not have modified the padding but instead the message, thereby revealing the length by showing where the message stopped and the padding began (in the decrypted plaintext).

We could not achieve the same result by simply modifying the last byte of the last block, since AES has the diffusion property. Modifying one byte would mess the entire block up.

This works specifically in CBC mode, since each AES-decrypted block is XORed with the ciphertext of the block before it. This One-Time-Pad-style XOR application does not have any diffusion, thus we can control the position of the plaintext byte we manipulate.

Exercise 1.2

1. Prove that all invertible functions from 1 bit to 1 bit are affine.

There are only four functions from 1 bit to 1 bit.

1. the identity function
2. logical not
3. always return 0
4. always return 1

Of these, only the identity function and logical-not are invertible.

We know that the identity function is affine, because of the following equality:

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

The logical not:

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

We can prove this by exhausting all cases, and checking whether or not the property holds. We fill out the truth table for the left hand side of the equation

x	y	$\neg x$	$\neg y$	$\neg x \oplus \neg y$
1	1	0	0	0
1	0	0	1	1
0	1	1	0	1
0	0	1	1	0

And do the same for the right hand side. First where $a = 0$

x	y	$x \oplus a$	$y \oplus a$	$\neg(x \oplus a)$	$\neg(y \oplus a)$	$\neg(x \oplus a) \oplus \neg(y \oplus a)$
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	1	0	0	1	1
1	1	1	1	0	0	0

Then, where $a = 1$

x	y	$x \oplus a$	$y \oplus a$	$\neg(x \oplus a)$	$\neg(y \oplus a)$	$\neg(x \oplus a) \oplus \neg(y \oplus a)$
0	0	1	1	0	0	0
0	1	1	0	0	1	1

x	y	$x \oplus a$	$y \oplus a$	$\neg(x \oplus a)$	$\neg(y \oplus a)$	$\neg(x \oplus a) \oplus \neg(y \oplus a)$
1	0	0	1	1	0	1
1	1	0	0	1	1	0

As we can see, the equivalence holds in both cases, thus proving that

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

2. Give an example of a 3 bit to 3 bit invertible function that is not affine.

$$f(x) = x + 1 \mod 8$$

is an example of a 3 bit to 3 bit invertible functino that is not affine.

Consider the following example:

$$x = 0y = 1a = 2$$

$$f(0) \oplus f(1) = 1 \oplus 2 = 3$$

$$f(0 \oplus 2) \oplus f(1 \oplus 2) = f(2) \oplus f(3) = 3 \oplus 4 = 7$$

$$3 \neq 7$$

This example shows that the $x+1 \mod 8$ function is not affine (for 3-bit inputs).

Exercise 1.3

Show that this encryption scheme does not fulfill perfect secrecy. To that end, exhibit a message $m \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^n$ such that

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

Seeing as XOR'ing with a key of only zeroes would be equivalent to the identity function and not change the plaintext whatsoever, any pair of message m and ciphertext c where $m = c$ would be impossible in the function $c = m \oplus k$, without k being 0. Any key k not equal to 0 (the only possible ones in this altered one-time pad encryption scheme), would change the message such that the equality does not hold.

> Explain how this implies that the modified one-time pad does not fulfill perfect secrecy

Seeing as the property of perfect secrecy requires that an adversary attains absolutely no knowledge about the contents of the message, after attacking the ciphertext with infinite resources, we can conclude that the modified one-time pad does not fulfull perfect secrecy.

The adversary has gained the knowledge that the plaintext cannot be exactly that of the ciphertext, and has therefore eliminated 1 of the possible plaintext messages.