TP4 Report

A85272	Jorge Mota
A83840	Maria Silva

This work had the objective to reflect on the bad implementations of CBC-MAC for data integrity validation.

First of all we considered that the messages had 2 AES blocks in size for this exercise.

We could weaken this MAC in two ways:

1. Using a random IV, instead of a fixed zero IV
2. Using as tag every block of the cryptogram

Using a random IV

To be able to get the tag for a message with a random IV then we need a `mac` function that receives a `iv` as parameter in addition to the `key` and `message`, because we need to pass the iv to the other side to be able to verify a MAC produced by this.

This fact will break the mac since the ability to change the iv passed to the mac producer function will enable to manipulate the message's first AES block.

```
# Receives the key, the message and the wanted
# "random" iv and produces a MAC
cbcmac(key, msg, iv)
```

Since we know that a cbc mac will **XOR** the message's first block with the selected **IV** then this part's output is predictable. With this information and a given pair of valid *message* and *tag* for an unknown key we can forge a valid pair with a different message and IV but with the same tag.

We first **XOR** the IV used to compute the given `tag` and the message's first block

```
MSG1 = Message's block 1
MSG2 = Message's block 2
IV = IV used to compute the TAG
MSG = MSG1:MSG2


NEWMSG = (MSG1 ^ IV):MSG2
```

After this we successfully have a `NEWMSG` that is valid for the same `tag` of the valid message given and an `IV` of zero's

Even better, we can produce a `NEWMSG` with a custom first block with the information we want

```
 # Having:
 CUSTOM_MSG1 = b'1234567890123456'


NEWIV = (MSG1 ^ IV) ^ CUSTOMMSG1
 MSG = CUSTOM_MSG1:MSG2
```

Using as tag every block

If every block of the ciphertext is used as a tag instead of just the last one, we can forge a fake tag for messages longer than one block with a chosen-plaintext attack:

If we intercept a message `M` and a tag `T`, with two blocks like this project's question:

```
M = M0||M1
T = T0||T1
```

We know that `T0 = E(M0)` and `T1 = E(T0 ⊕ M1)`.

So, if we make a new message

```
M' = (T0 ⊕ M1)||(T1 ⊕ M0)
```

and a new tag

```
T' = T1||T0
```

the verify function should accept it as valid.

`M` and `M'` will be different unless `M1 = M0 ⊕ E(M0)`, but we can take that risk, since it's not very likely to happen.