Block cipher modes

We have discussed block ciphers, and how to encrypt blocks of a specified size (64 bit for DES, or 128 bit for AES). **Block cipher modes** specify how to encrypt larger plaintexts.

Let E_k be the encryption routine of a block cipher with block size n bit. As a first step, we split a plaintext m into blocks $m = m_1 m_2 m_3 \ldots$ such that each m_i is n bits (we may have to pad).

Example 118. (ECB, shouldn't be used) In the simplest mode, known as **electronic codebook**, we just encrypt each plaintext block individually:

$$c_j = E_k(m_j)$$

The ciphertext is $c = c_1 c_2 c_3 \ldots$. Decryption simply computes $D_k(c_j) = m_j$.

Though natural, ECB has several severe weaknesses. Can you think of some?

Solution. Using ECB is nothing else but a classical substitution cipher, except that ECB operates on larger blocks. Just like a classical substitution cipher is vulnerable to frequency attacks, ECB leaves patterns in the ciphertext. For a striking visual example when encrypting a picture, see:

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

If a block repeats later in the message (or in a later message), it will be encrypted the same way. Hence, Eve can notice such repetitions. This is problematic in practice, for instance, because certain files always begin with the same blocks, so that Eve has a good chance of detecting the file type.

Also, knowing the filetype, Eve might be able to rearrange the ciphertext blocks to adjust the message. She can also attempt to delete certain ciphertext blocks.

Conclusion. Unless you know exactly why (e.g. sending already randomized messages), you should not use ECB.

Example 119. (CBC) In **cipherblock chaining** mode, we encrypt each plaintext block after chaining it with the previous cipherblock; that is:

$$c_j = E_k(m_j \oplus c_{j-1})$$

In order to do that for $j = 1$, we need a value for c_0, known as an **initialization vector** IV.

The ciphertext is $c = c_0 c_1 c_2 c_3 \ldots$ (that's one more block than for the plaintext $m = m_1 m_2 m_3 \ldots$).

(a) How does decryption work?

(b) Why should the value IV be unpredictable (e.g. be chosen randomly)?

Solution.

(a) Since $c_j = E_k(m_j \oplus c_{j-1})$, we have $D_k(c_j) = m_j \oplus c_{j-1}$ or $m_j = D_k(c_j) \oplus c_{j-1}$.
For instance. $m_1 = D_k(c_1) \oplus c_0$

(b) The value IV should be unique, so that messages starting with the same plaintext block have different ciphertext blocks. More generally, it should be unpredictable so that Eve cannot mount a chosen-plaintext attack to test if an earlier plaintext equals her guess. See Example 120.

Just checking. What would happen if we set $c_j = E_k(m_j) \oplus c_{j-1}$ instead? In that case, we would gain nothing over ECB: since Eve knows all c_j, she can compute $c_j \oplus c_{j-1} = E_k(m_j)$.

Comment. CBC makes random access possible during decryption (but not encryption). That means, we don't need to decrypt $c = c_0 c_1 c_2 c_3 \ldots$ sequentially but can directly decrypt $c_N c_{N+1} \ldots$ for some random N.

Example 120. (BEAST attack) BEAST is short for Browser Exploit Against SSL/TLS and was brought to public attention in 2011. The attack is based on the fact that the IV used by SSL was obtained from a previous ciphertext block (instead of randomly!):

Scenario. Imagine that plaintext blocks $m_1 m_2 \ldots$ are continuously being encrypted using CBC to cipherblocks. However, the plaintexts are from different parties and Eve can ask for her own plaintexts to be encrypted along the way.

In such a scenario, different plaintexts should be separately encrypted using CBC, meaning that a new random IV should be chosen each time.

Eve's goal. Suppose Eve has observed the ciphertext blocks c_{j-1}, c_j and her goal is to find out whether $m_j = x$ where x is her educated guess. Obviously, this is something that Eve should not be able to do!

The exploit. Because the IV for the next encryption is c_j, and because Eve can interject plaintext blocks to be encrypted for her, she can ask for $m_{j+1} = x \oplus c_{j-1} \oplus c_j$ (these are all known to Eve!) to be encrypted next.

Because CBC with IV c_j is used, this results in $c_{j+1} = E_k(m_{j+1} \oplus c_j) = E_k(x \oplus c_{j-1})$.

Eve can now compare this with $E_k(m_j \oplus c_{j-1}) = c_j$ (which she knows!) to find out whether $m_j = x$.

https://en.wikipedia.org/wiki/Transport_Layer_Security#BEAST_attack

There exist many other modes, including modes which already include features like authentication. Other common basic modes such as OFB (output feedback) or CTR (counter) turn the block cipher into a stream cipher (one advantage of that is that we don't need to encrypt full blocks at a time).

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Comment. One issue of ECB and CBC is the need for padding. If not handled properly, this can be exploited by a **padding oracle attack**:

https://en.wikipedia.org/wiki/Padding_oracle_attack

Example 121. Consider the (silly) block cipher with 4 bit block size and 4 bit key size such that

$$E_k(b_1 b_2 b_3 b_4) = (b_2 b_3 b_4 b_1) \oplus k.$$

(a) Encrypt $m = (0000\ 1011\ 0000\ \ldots)_2$ using $k = (1111)_2$ and ECB mode.

(b) Encrypt $m = (0000\ 1011\ 0000\ \ldots)_2$ using $k = (1111)_2$ and CBC mode ($\mathrm{IV} = (0011)_2$).

Solution. $m = m_1 m_2 m_3 \ldots$ with $m_1 = 0000$, $m_2 = 1011$ and $m_3 = 0000$.

(a) $c_1 = E_k(m_1) = 0000 \oplus 1111 = 1111$
$c_2 = E_k(m_2) = 0111 \oplus 1111 = 1000$
Since $m_3 = m_1$, we have $c_3 = c_1$. Hence, the ciphertext is $c = c_1 c_2 c_3 \ldots = (1111\ 1000\ 1111\ \ldots)$.

(b) $c_0 = 0011$
$c_1 = E_k(m_1 \oplus c_0) = E_k(0000 \oplus 0011) = E_k(0011) = 0110 \oplus 1111 = 1001$
$c_2 = E_k(m_2 \oplus c_1) = E_k(1011 \oplus 1001) = E_k(0010) = 0100 \oplus 1111 = 1011$
$c_3 = E_k(m_3 \oplus c_2) = E_k(0000 \oplus 1011) = E_k(1011) = 0111 \oplus 1111 = 1000$
Hence, the ciphertext is $c = c_0 c_1 c_2 c_3 \ldots = (0011\ 1001\ 1011\ 1000\ \ldots)$.

Comment. Clearly, our cipher is not meant to be secure. One damning issue (besides the short key and block size) is that it is linear (in both the plaintext and the key).