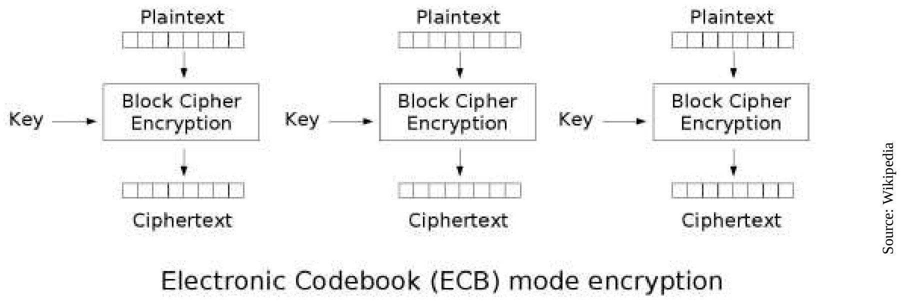
Handout 4

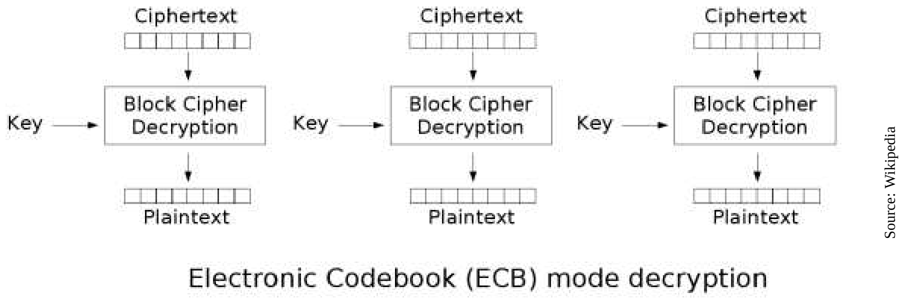
Summary of this handout: *Block Ciphers continued — Modes of Operation — Cryptomeria*

II.1.5 Modes for Block Ciphers

In practical use block ciphers are used to encode messages many times longer than the block size of a particular cipher. There are several different ways of applying block ciphers in order to encrypt long messages, which are called *Modes of Operation* There are five major modes of operation that we will have a closer look at.
In the following, let (E, D) be a block cipher with block length n, where E is the encryption and D is the decryption algorithm of the cipher.

34. ECB
The *Electronic Codebook mode* (ECB) corresponds to the “naïve” use of a block cipher. The message M is split into a sequence $M_1, \ldots, M_t$ of blocks of the length n that is handled by E. If necessary, the last block is padded, i.e. it is filled up with bits that do not obscure the message, such that its original length can be recovered. The simplest padding is achieved by adding null bits.
Each block M_i is then with the same key K. Thus we obtain $C_i := E_K(M_i)$ with resulting ciphertexts $C_1, \ldots, C_t$. Encryption and decryption works therefore as follows:





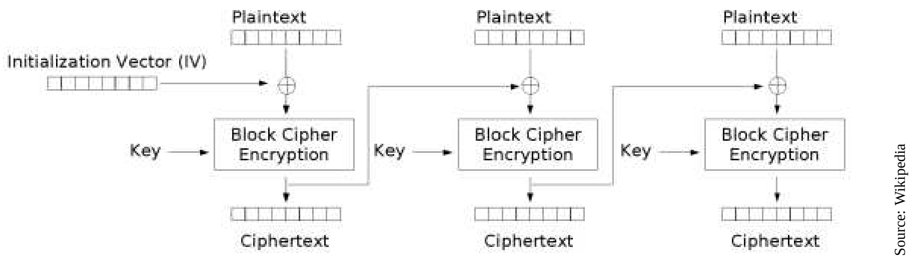
Since every block is encrypted independently the advantage of ECB is that if one block C_i gets corrupted due to unreliable channels, then only one block is affected while all other blocks can still be decrypted.
However, the main weakness of ECB is that identical blocks M_i are encrypted to the same ciphertext C_i, which makes the cipher vulnerable. Moreover, there is no protection against deletion or insertion of blocks. Thus Bob has no way of knowing if the message from Alice is complete or that Eve has not inserted something.

35. CBC

The *Cipher Block Chaining mode* (CBC) overcomes the problem of unwanted insertion and deletion by adding information on the context of the message. Again, the message M is split into a sequence $M_1, \ldots, M_t$ of blocks of length n. Encryption is then performed by the following operations:

$$\begin{aligned} C_1 &= E_K(M_1 \oplus IV) \\ C_i &= E_K(M_i \oplus C_{i-1}), \text{ for } i > 1 \end{aligned}$$

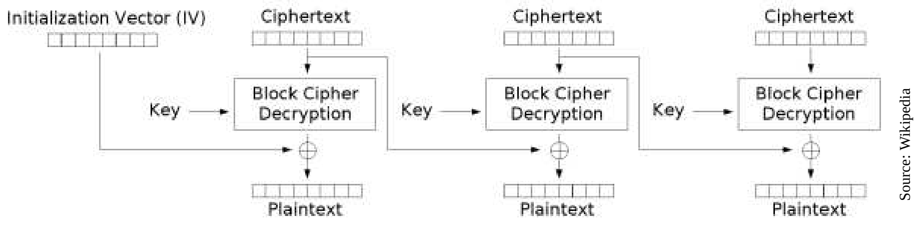
Note that to start off CBC encryption we need an *initial value* (or initialisation vector) IV to be passed to the encryption function. The initial value is used to ensure that two encryptions of the same plaintext yield different ciphertexts. IV does not have to be kept secret and is usually transmitted as plaintext together with the ciphertext message. Encryption is shown below:



Cipher Block Chaining (CBC) mode encryption

Decryption is achieved by “xor-ing out” the previous ciphertext from the decrypted message:

$$\begin{aligned} M_1 &= D_K(C_1) \oplus IV \\ M_i &= D_K(C_i) \oplus C_{i-1} \text{ for } i > 1 \end{aligned}$$



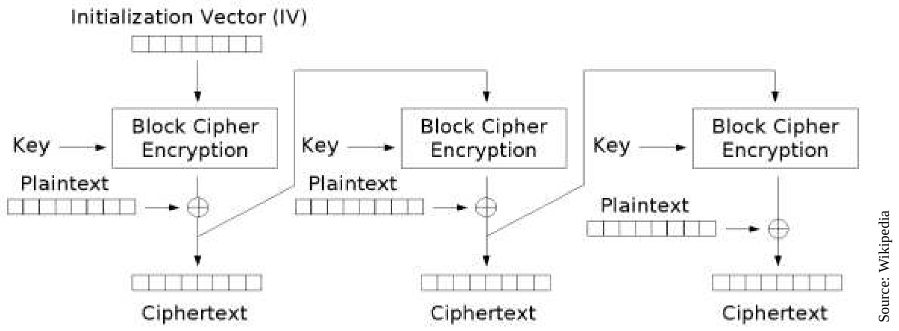
Cipher Block Chaining (CBC) mode decryption

Since each C_i is xor-ed with the next block of the plaintext, a one-bit error in the ciphertext gives not only a one-block error in the corresponding message block but also a one-bit error in the next decrypted plaintext block.

36. CFB

The *Cipher Feedback mode* (CFB) uses again an initial value IV to kick off encryption. It encrypts IV with E_K and then xor-s the first plaintext block with the result. This yields the first ciphertext block, which is used in the next round of encryption, where it is passed to E_K again and the next plaintext block is xor-ed to the result. Encryption thus works as follows:

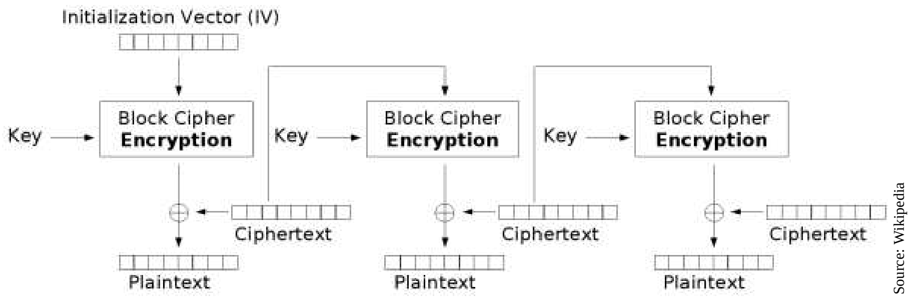
$$\begin{aligned} C_1 &= E_K(IV) \oplus M_1 \\ C_i &= E_K(C_{i-1}) \oplus M_i \end{aligned}$$



Cipher Feedback (CFB) mode encryption

Observe that the plaintext messages are never actually encrypted directly with E_K. This has two effects. Firstly the length of plaintext message blocks does not necessarily have to be the same as the block length n of the cipher. Instead it is allowable to have any plaintext message block length $1 \leq j \leq n$, since only IV and the resulting ciphertext blocks have to have block size n. Obviously only the relevant j bits have to be sent and the value of j then needs to be known for decryption.
Secondly during decryption we only need to reproduce that part of the cipher that has to be “xor-ed out” from the received ciphertext, which can be done with initial value IV and the right encryption function E_K alone, i.e., we do not need D_K. Decryption works than as follows:

$$\begin{aligned} M_1 &= E_K(IV) \oplus C_1 \\ M_i &= E_K(C_{i-1}) \oplus C_i \end{aligned}$$

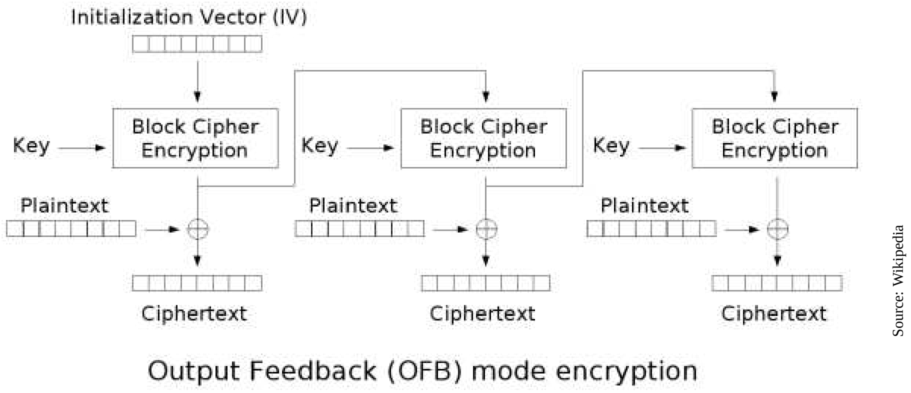


Cipher Feedback (CFB) mode decryption

In CFB a one-bit error in the ciphertext causes a one-bit error in the corresponding plaintext block and a block-error in the next decrypted plaintext block, i.e., the opposite of what happens in the CBC mode.

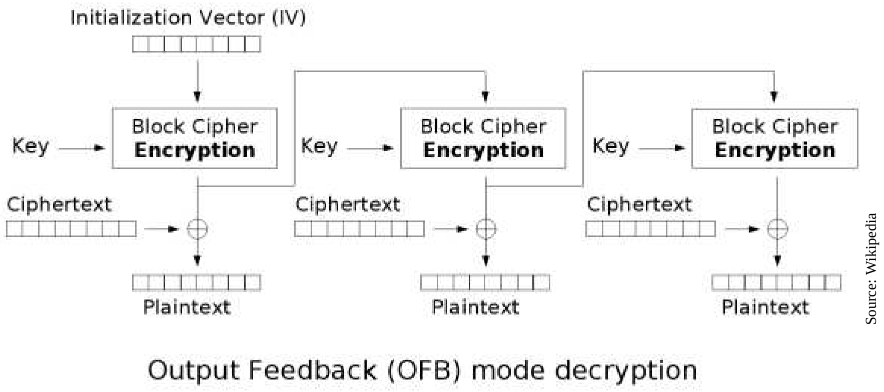
37. OFB The *Output Feedback mode* (OFB) is very similar to CFB with the exception that the result of E_K is passed on directly into the encryption of the next block. Thus the encryption depends essentially on the initialisation vector IV only:

$$\begin{aligned} O_1 &= E_K(IV), & C_1 &= O_1 \oplus M_1 \\ O_i &= E_K(O_{i-1}), & C_i &= O_i \oplus M_i \end{aligned}$$



Decryption again only needs the encryption function E_K and the initial value IV:

$$\begin{aligned} O_1 &= E_K(IV), & M_1 &= O_1 \oplus C_1 \\ O_i &= E_K(O_{i-1}), & M_i &= O_i \oplus C_i \end{aligned}$$

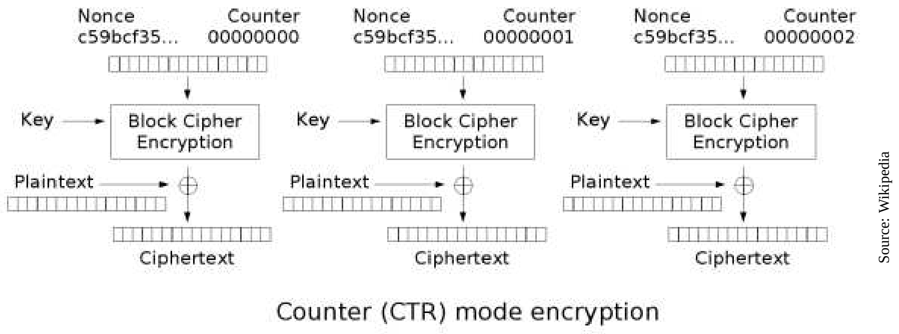


Similar to CFB, OFB mode can work on smaller plaintext block sizes.
Since it does not propagate the context by carrying over ciphertexts it has the advantage that a one-bit error in the ciphertext gives only a one-bit error in the decrypted ciphertext. Nevertheless it is immune to deletion or insertion attacks since decryption of plaintext is only successful if performed in sync with the O_i values.

38. CTR

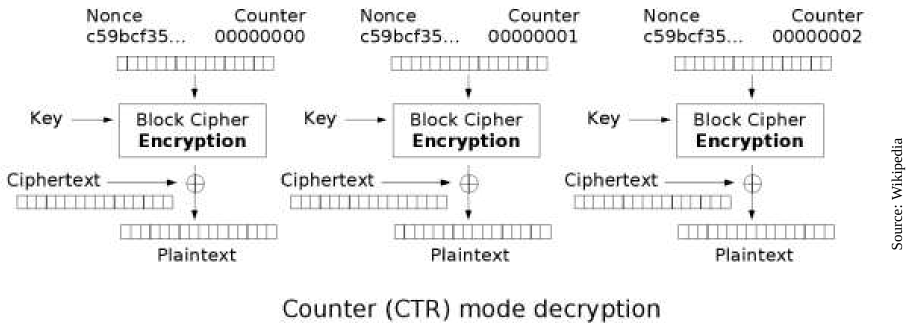
The *Counter mode* (CTR) is an even simpler variant of OFB. It generates each block of ciphertext by encrypting successive values of a counter and xor-ing it with the plaintext blocks. The counter can be any simple function that produces a sequence that is guaranteed not to repeat for a long time. However, using an actual counter is the simplest and most popular choice. In the graph below we use a counter together with a *Nonce* combined for example by xor or concatenation. This results in a non-trivial counter. The expression *Nonce* stands for "a number that is only used once" and is similar to the initial value IV used in the previous block modes. Encryption can therefore be implemented easily as:

$$C_i \quad = \quad E_K(Count_i) \oplus M_i$$



And similarly decryption is implemented as:

$$M_i \quad = \quad E_K(Count_i) \oplus C_i$$

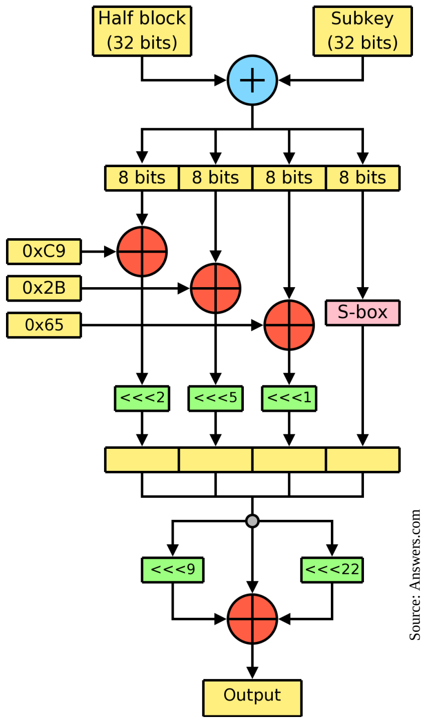


CTR has similar properties to OFB in that a one-bit error in a ciphertext block also only results in a one-bit error in the decrypted ciphertext. Moreover, each round of encryption and decryption is independent of the previous round and therefore parallelisation can be used for both types of algorithms.
Finally, while it is impossible to recover decryption for CFB if blocks have been lost during message transmission and it is relatively difficult to do for OFB, in CTR error recovery on missing blocks, i.e. decrypting successfully the received part of the message is fairly easy.

Observe that the CFB, OFB, and CTR modes of operation turn a block cipher effectively into a stream cipher since we essentially have an encryption method that on the one hand keeps track of states and on the other hand is independent of the actual plaintext message to transmit. We will have a close look at stream ciphers, nonces and their creation later on in this term.

II.1.6 Cryptomeria

Cryptomeria (or C2) is a Feistel Cipher derived from DES that is used as an content protection mechanism for some modern multimedia DVDs and was developed by a consortium of IBM, Intel, Matsushita and Toshiba. It has a block size of 64 bit, a key size of 56 bit, and uses 10 rounds of encryption. Its structure and implementation is publicly available, with the exception of a proprietary *secret constant*, which is essentially an S-Box substitution, that is used in the Feistel function and the key schedule. Here is an overview of the Feistel function. Observe, that the topmost operation is an actual arithmetic plus that adds the two 32 bit numbers of the half block and the round key together.



The key schedule for Cryptomeria works as follows: The round key is produced by taking the third 8 bit block of the key K, xor-ing it with the round number, and substituting it in the secret S-Box. The result is then shifted (not cyclically!) 4 bits left and added (again arithmetically) to the right 32 bits of the key. This yields the round key. For the next round the key K is then shifted cyclically left by 9 bits, i.e. $K \lll 9$.
DVD content encryption is done in four layers of encryption using Cryptomeria in a Cipher Block Chaining mode. Copy protection is achieved essentially by the following means:

1. A *Media Key Block* (MKB) is stored as a file on the disc and contains a very large number of keys. Each licensed DVD player has assigned to it a set of unique device keys that allow it to obtain the Media Key from the MKB and decrypt the audio content. This makes it possible to revoke hacked devices (i.e. a player that has been purpose built or manipulated to crack the cipher).

2. Each medium has a *unique media ID* given as a bar code in the so called *Burst Cutting Area* around the disc hub. This ID is used for decryption and prevents bit-wise copying as a different DVD will have a different ID number.

Cryptography Glossary 4

C2	Short for Cryptomeria.	34
CBC	Short for Cipher Block Chaining	30
CFB	Short for Cipher Feedback mode	31
Cipher Block Chaining mode	A mode of operation for block ciphers that propagates context information.	30
Cipher Feedback mode	A mode of operation that turns a block cipher effectively into a stream cipher by using context information.	31
Counter mode	A mode of operation that turns a block cipher effectively into a stream cipher by using a counter.	33
Cryptomeria	A Feistel Cipher used as content protection mechanism for multimedia DVDs.	34
CTR	Short for Counter mode	33
ECB	Short for Electronic Codebook mode	29
Electronic Codebook mode	A naïve mode of operation for block ciphers that encrypts every block independently.	29
Initial Value	An arbitrary value used to kick off encryption in block cipher modes. Initial values are normally denoted as IV and do not need to be kept secret.	30
Initialisation Vector	See Initial Value.	30
Modes of Operation	Ways of applying block ciphers to encode large messages.	29
Nonce	A number that is only used once. It is usually a random or pseudo-random number.	33
OFB	Short for Output Feedback mode	32
Output Feedback mode	A mode of operation that turns a block cipher effectively into a stream cipher by using context information.	32
Padding	Adding nulls at the end of messages to obtain the required block size. It is important that the nulls do not obscure the message, such that its original length can be recovered. The simplest is padding is achieved by adding null bits.	29