6 Block Ciphers

6.1 Block Ciphers

Block Ciphers

Plaintext is divided into blocks of fixed length and every block is encrypted one at a time.

A block cipher is a set of ‘code books’ and every key produces a different code book. The encryption of a plaintext block is the corresponding ciphertext block entry in the code book.

\[
\text{Plaintext block } m \rightarrow \text{Cipher function } e \rightarrow \text{Ciphertext block } c
\]

Block Ciphers

We have \( c = e_k(m) \) where:

- \( m \) is the plaintext block
- \( k \) is the secret key
- \( e \) is the encryption function
- \( c \) is the ciphertext

In addition we have a decryption function \( d \) such that:

\[
m = d_k(c)
\]

Symmetric Key Cryptography

Geoff Hamilton
\[ m = \text{plaintext}, \ c = \text{ciphertext}, \ k = \text{key}, \ KG = \text{key generator}. \]

**Symmetric Key Cryptography**
We write \( c = e_k(m) \), where:
- \( m \) is the plaintext,
- \( e \) is the encryption function,
- \( k \) is the secret key,
- \( c \) is the ciphertext.

Decryption is given by \( m = d_k(c) \).
Both sides need to know the key \( k \), but \( k \) needs to be kept secret.
- secret-key, single-key or one-key algorithms.

**Iterated Block Ciphers**
An iterated block cipher involves repeated use of a round function.
The idea is to make a strong encryption function out of a weaker round function (easy to implement) by repeatedly using it.
The round function takes an \( n \)-bit block to an \( n \)-bit block.
**Parameters:** number of rounds \( r \), blocksize \( n \), keysize \( s \).
Each use of the round function employs a subkey:
\[ k_i \text{ for } 1 \leq i \leq r \]
derived from the key \( k \).
For every subkey the round function must be invertible; if not decryption is impossible.
6.2 DES

Data Encryption Standard (DES)
Known as Data Encryption Algorithm DEA (ANSI) or DEA-1 (ISO).

- A de-facto international standard for banking security.
- An example of a Feistel Cipher.
- Most analyzed cryptographic algorithm.

Short history of DES:

- Work started in early 1970s by IBM.
- Based on IBMs Lucifer, but amended by NSA.
- Design criteria were kept secret for more than 20 years.
- Supposed to be reviewed every 5 years.

Data Encryption Standard (DES)
Block length is 64 bits, key length 56 bits.

Feistel Cipher:

- Initial permutation of bits.
- Split into left and right half.
- 16 rounds of identical operations, depending on round key.
- Inverse initial permutation.

Round transformation:

- Linear expansion: 32 bits → 48 bits.
- XOR with subkey of 48 bits (key schedule selects 48 bits of key $k$).
- 8 parallel non-linear S-boxes (6 input bits, 4 output bits).
- Permutation of the 32 bits.
Data Encryption Standard (DES)

Each DES round consists of the following six stages:

1. **Expansion Permutation:**
   - Right half (32 bits) is expanded (and permuted) to 48 bits.
   - Diffuses relationship of input bits to output bits.
   - Means one bit of input affects two substitutions in output.
   - Spreads dependencies.

2. **Use of Round Key:**
   - 48 bits are XOR-ed with the round key (48 bits).

3. **Splitting:**
   - Result is split into eight lots of six bit values.

Data Encryption Standard (DES)

4. **S-Box:**
   - Each six bit value is passed into an S-box to produce a four bit result in a non-linear way. (S = Substitution)

5. **P-Box:**
   - 32 bits of output are combined and permuted. (P = Permutation)

Geoff Hamilton
6. **Feistel Part:**

- Output of $f$ is XOR-ed with the left half resulting in new right half.
- New left half is the old right half.

**Data Encryption Standard (DES)**

![Diagram of Data Encryption Standard (DES)]

**Data Encryption Standard (DES)**

S-Boxes represent the non-linear component of DES.

There are eight different S-boxes.

The original S-boxes proposed by IBM were modified by NSA.

Each S-box is a table of 4 rows and 16 columns

- The 6 input bits specify which row and column to use.
- Bits 1 and 6 generate the row.
- Bits 2-5 generate the column.

### 6.3 Feistel Cipher

Feistel Cipher

Geoff Hamilton
Feistel Cipher
The round function is invertible regardless of the choice of \( f \).

Encryption round is:

- \( L_i = R_{i-1} \)
- \( R_i = L_{i-1} \oplus f(R_{i-1}, k_i) \)

Remark: in last step, the left half and right half are swapped:

- Decryption is achieved using the same \( r \)-round process.
- But with round keys used in reverse order, i.e. \( k_r \) first and \( k_1 \) last.

Feistel Cipher
Same algorithm can be used for decryption since we are using the Feistel structure:

\[ L_{i-1} \oplus f(R_{i-1}, k_i) \oplus f(R_{i-1}, k_i) = L_{i-1} \]

Remark: for decryption the subkeys are used in the reverse order.

Note that the effect of \( IP^{-1} \) is cancelled by \( IP \).

Also note that \( R_{16}, L_{16} \) is the input of encryption since halves were swapped and that swapping is necessary.

6.4 AES

Advanced Encryption Standard (AES)
January 1997: NIST call for algorithms to replace DES.

- Block cipher: 128-bit blocks, 128/192/256-bit keys.
- Strength \( \approx 3 \times \) DES, much more efficient.
• Documentation, reference C code, optimised C and JAVA code, test vectors.

• Designers give up all intellectual rights.

Open process: public comments, international submissions.
Website: http://www.nist.gov/aes/

Advanced Encryption Standard (AES)
Official scope was limited:


• Documents that are ‘sensitive but not classified’.

• 2003: NSA has approved AES-128 also for secret information, and AES with key sizes larger than 128 for top secret information.

• Significance is huge: AES is the successor of DES.

• Major factors for quick acceptance:
  – No royalties.
  – High quality.
  – Low resource consumption.

Rijndael
Block length, key length: varies between 128 - 256 bits.
Number of rounds is 10/12/14 depending on block and key length.
Uniform and parallel round transformation, composed of:

• Byte substitution.

• Shift rows.

• Mix columns.

• Round key addition.

Sequential and light-weight key schedule.
No arithmetic operations.

Geoff Hamilton
Rijndael

Plaintext block normally is 128 bits or 16 bytes \(m_0,\ldots,m_{15}\).

Key normally is 128/192/256 bits or 16/24/32 bytes \(k_0,\ldots,k_{31}\).

Both are represented as rectangular array of bytes.

\[
\begin{array}{cccc}
  m_0 & m_4 & m_8 & m_{12} \\
  m_1 & m_5 & m_9 & m_{13} \\
  m_2 & m_6 & m_{10} & m_{14} \\
  m_3 & m_7 & m_{11} & m_{15} \\
\end{array}
\quad |
\begin{array}{cccc}
  k_0 & k_4 & k_8 & k_{12} \\
  k_1 & k_5 & k_9 & k_{13} \\
  k_2 & k_6 & k_{10} & k_{14} \\
  k_3 & k_7 & k_{11} & k_{15} \\
\end{array}
\]

Rijndael: Byte Substitution

State matrix is transformed byte by byte.

S-box is invertible, else decryption would not work.

Only one S-box for the whole cipher (simplicity).

\[
\begin{array}{cccc}
  a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\
  a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\
  a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\
  a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\
\end{array}
\quad |
\begin{array}{cccc}
  b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\
  b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\
  b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\
  b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\
\end{array}
\]

Rijndael: Shift Rows

Rows shifted over different offsets (depending on block length).

Geoff Hamilton
Purpose: diffusion over the columns.

\[
\begin{array}{cccc}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cccc}
  f & g & h & e \\
  k & l & i & j \\
  p & m & n & o \\
\end{array}
\]

**Rijndael: Mix Columns**
- Operations over finite field \( GF(2^8) \): the new column \([b_{0,i}, b_{1,i}, b_{2,i}, b_{3,i}] \) is produced by taking the polynomial \( a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3 \) and multiplying it by the polynomial \( 2 + x + x^2 + 3x^3 \) modulo \( x^4 + 1 \).
- Bytes in columns are combined **linearly**.
- Good diffusion properties over rows.

**Rijndael: Round Key Addition**
Round key is simply XOR-ed with state matrix.

Geoff Hamilton
Rijndael: Round Function

![Diagram of Rijndael round function]

Rijndael: Pseudo-code
Rijndael with 10 rounds is described by the following code:

```plaintext
AddRoundKey(S,K[0]);
for (i = 1; i <= 9; i++)
    {
        SubBytes(S);
        ShiftRows(S);
        MixColumns(S);
        AddRoundKey(S,K[i]);
    }
SubBytes(S);
ShiftRows(S);
AddRoundKey(S,K[10]);
```

Rijndael: Key Schedule

Geoff Hamilton
Example: key of 192 bits or 6 words of 32 bits.

\[ W^6_k = W_{k-6} \oplus f(W^{6n-1}_k) \]
\[ W^i_k = W^{i-6}_k \oplus W^{i-1}_k \]

Cipher key expansion can be done just-in-time: no extra storage required.

Comparing AES and DES

**DES:**
- S-P Network, iterated cipher, Feistel structure
- 64-bit block size, 56-bit key size
- 8 different S-boxes
- non-invertible round
- design optimised for hardware implementations
- closed (secret) design process

**AES:**
- S-P Network, iterated cipher
- 128-bit block size, 128-bit (192, 256) key size
- one S-box
- invertible round
- design optimised for byte-orientated implementations
- open design and evaluation process

Geoff Hamilton
6.5 Block Cipher Security

Attacks on Symmetric Ciphers

Modern cryptography operates under Kerckhoff’s principle: attacker knows everything about the algorithm, only the keys are assumed to be secret.

The typical goal of an attacker is to find the key given a pair of plaintext and ciphertext (known-plaintext attack).

Hence, the first observation is that we must make sure that exhaustive keysearch does not succeed:

- Given a few plaintext/ciphertext pairs, search through all possible keys until the correct key is found.
- The number of keys must be large enough.

Time-memory trade-off: pre-compute a list consisting of pairs of plaintexts and ciphertexts and the associated (random) keys. During an attack this leads to a speed-up: given the plaintext, the attacker looks for it in the pre-computed list.

Attacks on Symmetric Ciphers

Ciphertext-only attack:

- Adversary only has ciphertext of several messages.
- Recover the plaintext or deduce the keys used during encryption.

Known-plaintext attack:

- Adversary has several ciphertexts and corresponding plaintexts.
- Deduce keys used to encrypt messages or decrypt new messages.

Chosen-plaintext attack:

- Adversary can choose the plaintexts that get encrypted.
- More powerful than known-plaintext attack, because specific plaintexts can be chosen.

Block Size

The block size $n$ needs to be reasonably large, $n > 64$ bits, to avoid:

- Text dictionary attacks: plaintext-ciphertext pairs for fixed key.
- Matching ciphertext attacks: uncover patterns in plaintext.
Security of DES

**Exhaustive key search** (1 or 2 known plaintext/ciphertext pairs).

- Number of possibilities for DES is $2^{56} = 7.2 \times 10^{16}$.

**Software** (PC with 3.2 GHz Processor): $2^{48}$ encryptions per year.

- $2^{23}$ keys per second, $2^{16.4}$ seconds per day, $2^{8.5}$ days per year.
- 1 PC: 125 years, 125 PC’s: 1 year, 1500 PC’s: 1 month.

**Hardware** (ASIC), Cost = $250,000, ‘Deep Crack’ (EFF, 1998).

- 1 key in 50 hours, less than $500 per key.
- Time halved by working in conjunction with distributed.net
- For $1M$: 1 key in 1/2 hour.

**Hardware** (FPGA), Cost = $10,000, ‘COPACABANA’, 2006

- 9 days (later reduced to 6 days)
- Reduced to less than one day by successor machine ‘RIVYERA’

Security of DES

**Export from US**: 40-bit keys (SSL, Lotus Notes, S/MIME).

- Obviously much less secure.

**Moore’s ‘law’**:

- Computing power doubles every 18 months.
- After 21 years the effective key size is reduced by 14 bits.

**Long term**: key length and block length of 128 bits.

Multiple Encryption

DES is a ‘standard’: neither key size nor block size nor number of rounds can be changed (easily).

**Remember**: more rounds bring more security.

**Idea**: iterating the entire cipher might bring more security.

Double encryption, triple encryption, quadruple encryption, etc.

What makes most sense?

Geoff Hamilton
**Double Encryption**

Time-memory trade-off via a **meet-in-the-middle** attack is possible:

- Pre-compute for all keys \(k_1 : C_1 = E_{k_1}(P)\).
- Given a ciphertext \(C\): invert \(C\) and compare with list of \(C_1\). Hit indicates \(k_2\) and gives \(k_1\). Check with one more \(C\).
- Double encryption does not provide double security.

**Triple DES**

Invented to get around the problem of a **short key**.
Involves use of 2 or 3 DES keys.

### 6.6 Other Block Ciphers

**IDEA**
IDEA (International Data Encryption Algorithm) encrypts 64-bit plaintext to 64-bit ciphertext blocks, using a 128-bit input key \(K\).
Uses a novel involution which is a generalization of the Feistel structure:

IDEA

8 computationally identical rounds followed by an output transformation. Uses 3 different 16-bit operations in three different algebraic groups:

- bitwise XOR denoted $a \oplus b$
- addition mod $2^{16}$ denoted $a \boxplus b$
- multiplication mod $2^{16} + 1$ denoted $a \odot b$

Geoff Hamilton
SAFER
SAFER K-64 (Secure And Fast Encryption Routine, with 64-bit key) is an iterated block cipher with 64-bit plaintext and ciphertext blocks. Targeted at 8-bit microprocessors.
8 computationally identical rounds followed by an output transformation.
In contrast to Feistel-like ciphers, the operations used for encryption differ from those for decryption.
Can be viewed as a substitution-permutation network.

RC5
Invented by Ron Rivest (Ron’s Cipher 5).
Encryption can be described by the following pseudo-code, where the input block is (A,B):

\[
\begin{align*}
A &= A + K[0] \\
B &= B + K[1] \\
A &= (A \oplus B) \ll B + K[2i] \\
B &= (B \oplus A) \ll A + K[2i+1]
\end{align*}
\]

Where A and B are w-bit halves of the state, r is the number of rounds, and b is the number of bytes in the key.
A particular implementation of RC5 is referred to as RC5-w/r/b, for example RC5-32/12/16.
The operation \(\ll x\) implies a rotation by the number obtained from the \(\log_2(w)\) low-order bits of x.

Geoff Hamilton
TEA

TEA (Tiny Encryption Algorithm).

64 bit block, 128 bit key.

Number of rounds is variable (32 is considered secure).

Uses “weak” round function, so large number of rounds required.

Almost a Feistel cipher: uses + and - instead of ⊕.

Simple, easy to implement, fast, low memory requirement, etc

TEA Encryption:

(K[0], K[1], K[2], K[3]) = 128 bit key
(L,R) = plaintext (64-bit block)
delta = 0x9e3779b9
sum = 0
for i = 1 to 32
    sum = sum + delta
    L = L + ((R << 4) + K[0]) ⊕ (R + sum) ⊕ ((R >> 5) + K[1])
    R = R + ((L << 4) + K[2]) ⊕ (L + sum) ⊕ ((L >> 5) + K[3])
next i
ciphertext = (L,R)

TEA Decryption:

(K[0], K[1], K[2], K[3]) = 128 bit key
(L,R) = ciphertext (64-bit block)
delta = 0x9e3779b9
sum = delta << 5
for i = 1 to 32
    R = R - ((L << 4) + K[2]) ⊕ (L + sum) ⊕ ((L >> 5) + K[3])
    L = L - ((R << 4) + K[0]) ⊕ (R + sum) ⊕ ((R >> 5) + K[1])
    sum = sum - delta
next i
plaintext = (L,R)

FEAL

Fast data Encryption Algorithm.

Designed as replacement for DES.

Designed to be fast and efficient with modest security.

Original version (FEAL-4) found to be weak.

• Many “improved” versions followed
• All are flawed to some degree

Important in history of cryptanalysis.

Differential cryptanalysis developed for FEAL.

Good example to illustrate both linear and differential attacks.

Geoff Hamilton
FEAL-4
We will look more closely at FEAL-4.

- 4-round Feistel cipher with a 64-bit block and 64-bit key.
- A key scheduler expands a 64-bit key into 12 16-bit subkeys or round keys.
- Some of these subkeys are used for key whitening, in which they are combined with portions of the data using XOR before the first and after the last round.
- The cipher can be simplified to an equivalent cipher in which some of the whitening has been removed and 6 32-bit subkeys $K_0, \ldots, K_5$ are used.
- Round function $F$ maps 32 bits to 32 bits.

FEAL-4

To define the round function $F$, we first define:

- $G_0(a, b) = (a + b \mod 256) \ll 2$
- $G_1(a, b) = (a + b + 1 \mod 256) \ll 2$

Where $\ll$ is left cyclic shift (rotation)

Then $F(x_0, x_1, x_2, x_3) = (y_0, y_1, y_2, y_3)$ where:

- $y_0 = G_0(x_0, y_1)$
- $y_1 = G_1(x_0 \oplus x_1, x_2 \oplus x_3)$

Geoff Hamilton
• \( y_2 = G_0(y_1, x_2 \oplus x_3) \)
• \( y_3 = G_1(y_2, x_3) \)

**FEAL-4**
The round function \( F \) can be viewed schematically as follows:

---

**Non-Linearity in Block Ciphers**
Non-linearity is often achieved in block ciphers within the S-Boxes.
This can be achieved by implementing the S-Boxes as non-linear functions.
In AES, the S-Box is generated by determining the multiplicative inverse in \( GF(2^8) = \mathbb{Z}_2[x] \pmod{x^8 + x^4 + x^3 + x + 1} \).
This is a non-linear function.
In SAFER, the S-Box implements the function \( f(x) = 45^x \pmod{257} \) and the inverse S-Box implements the function \( f^{-1}(x) = \log_{45}x \pmod{257} \) (note that 45 is a primitive root in \( \mathbb{Z}_{257} \)).
These are also non-linear functions but they have some properties which leave them susceptible to attack e.g. \( f(x) + f(x + 128) = 1 \pmod{256} \).

**Non-Linearity in Block Ciphers**
The non-linearity in IDEA is achieved by mixing the three different algebraic group operations.
The structure is organised so that the output of one operation is never used as the input to an operation of the same type.
The three operations are incompatible in the sense that:

1. No pair of the 3 operations satisfies a distributive law e.g. \( a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c) \).

2. No pair of the three operations satisfies an associative law e.g. \( a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c \).

Geoff Hamilton
Non-Linearity in Block Ciphers
In RC5 the data-dependent rotations provide the non-linearity; the amount of rotation depends on the plaintext itself.
TEA relies on the alternate use of XOR and ADD to provide non-linearity.
The alternate use of addition modulo 256 and exclusive-or operations in SAFER is also used to provide confusion.
However, there is a simple and useful connection between the two operations when used on bytes: $a \oplus 128 = a + 128 \pmod{256}$.
In FEAL, it is the $G$ functions which are meant to provide the non-linearity, but they fail in this respect.

Block vs Stream Ciphers
Which is best?
Block ciphers:
- More versatile: can be used as stream cipher.
- Standardisation: DES and AES + modes of operation.
- Very well studied and accepted.
Stream ciphers:
- Easier to do the maths.
- Either makes them easier to break or easier to study.
- Supposedly faster than block ciphers (less flexible).

6.7 Modes of Operation
Modes of Operation
If message is longer than blocksize, block cipher can be used in a variety of ways to encrypt the plaintext.
Soon after DES was made a US Federal standard, another US standard appeared giving four recommended ways of using DES for data encryption.
These modes of operation have since been standardised internationally and can be used with any block cipher.

ECB Mode
ECB = Electronic Code Book
Simplest approach to using a block cipher.
Plaintext $m$ is divided into $t$ blocks of $n$ bits $m_1,m_2,\ldots,m_t$ (the last block is padded if necessary).
Ciphertext blocks $c_1,\ldots,c_t$ are defined as follows:
$$c_i = e_k(m_i)$$
Note that if $m_i = m_j$ then we have $c_i = c_j$; thus patterns in plaintext reappear in ciphertext.

Geoff Hamilton
ECB Encipherment

ECB Decipherment

ECB Mode

Properties:

- Blocks are independent.
- Does not hide patterns or repetitions.
- Error propagation: expansion within one block.
- Reordering of blocks is possible without too much distortion.
- Stereotyped beginning/ends of messages are common.
- Susceptible to block replay.

Geoff Hamilton
Block Replay

Block replay:

- Extracting ciphertext corresponding to a known piece of plaintext.
- Amending other transactions to contain this known block of text.

Countermeasures against block replay:

- Extra checksums over a number of plaintext blocks.
- Chaining the cipher; this adds context to a block.

CBC Mode

CBC = Cipher Block Chaining

Plaintext $m$ is divided into $t$ blocks of $n$ bits $m_1, m_2, \ldots, m_t$ (the last block is padded if necessary).

Encryption:

- $c_1 = e_k(m_1 \oplus IV)$.
- $c_i = e_k(m_i \oplus c_{i-1})$ for $i > 1$.

Decryption:

- $m_1 = d_k(c_1) \oplus IV$.
- $m_i = d_k(c_i) \oplus c_{i-1}$ for $i > 1$.

CBC Encipherment

![CBC Encipherment Diagram]

Geoff Hamilton
CBC Decipherment

CBC Mode
Properties:
- Ciphertext depends on all previous plaintext blocks (internal memory).
- Different IV hides patterns and repetitions.
- Error propagation: expansion in one block, copied in next block.
- Decryption of one block requires only ciphertext of previous block, so CBC is self-synchronizing.
- Rearranging order of blocks affects decryption (not if previous ciphertext block is correct).
- Default mode to use.

Padding
Problem: suppose length of plaintext is not multiple of block length.
- Last block $m_i$ only contains $k < n$ bits.

Padding schemes:
- Append $n - k$ zeroes to last block. Problem is that trailing 0-bits of plaintext cannot be distinguished from padding.
- Append 1 and $n - k - 1$ 0-bits. If $k = n$, then create extra block starting with 1 and remaining $n - 1$ bits 0.
- As above, but add an extra block which contains the length of the message in bits.

Geoff Hamilton
Example

Plaintext: original picture

Example

Ciphertext: ECB Encryption

Example

Geoff Hamilton
OBF Mode

OBF = Output FeedBack

This mode enables a block cipher to be used as a stream cipher.

- The block cipher creates the keystream.
- Block length for block cipher is $n$.
- Can choose to use keystream blocks of $j \leq n$ bits only.
- Divide plaintext into a series of $j$-bit blocks $m_1, \ldots, m_t$.
- **Encryption**: $c_i = m_i \oplus e_i$, where $e_i$ is selection of $j$ bits of ‘ciphertext’ generated by block cipher, starting with $IV$ in shift register.

OBF Encryption

Geoff Hamilton
OFB Decryption

OFB Mode
Properties:
- Synchronous stream cipher.
- No linking between subsequent blocks.
- Different IV necessary; otherwise insecure.
- Only uses encryption (no decryption algorithm necessary).
- If $j < n$: more effort per bit.

Geoff Hamilton
• Key stream independent of plaintext: can be precomputed.
• No error propagation: errors are only copied.

**CFB Mode**

CFB = Cipher FeedBack

In OFB Mode the keystream is generated by:

• Encrypting the IV.
• Encrypting the output from this encryption.

In CFB Mode the keystream is generated by:

• Encrypting the IV.
• Encrypting \( n \) bits of ciphertext.

**CFB Encryption**

![Diagram of CFB Encryption]

**CFB Mode**

Properties:

• Self-synchronizing stream cipher.
• Ciphertext depends on all previous plaintext blocks (internal memory).
• Different IV hides patterns and repetitions.
• Only uses encryption (no decryption algorithm necessary).
• If \( j < n \): more effort per bit.
• **Error propagation:** propagates over \([n/j] + 1\) blocks.

• **No synchronisation needed** between sender and receiver; synchronisation if \(n\) bits have been received correctly.

• Often used with \(j = 1, 8\) because of synchronisation.

### Mode Summary

<table>
<thead>
<tr>
<th>Patterns</th>
<th>ECB</th>
<th>CBC</th>
<th>OFB</th>
<th>CFB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repetitions</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Block length</td>
<td>(n)</td>
<td>(n)</td>
<td>(j)</td>
<td>(j)</td>
</tr>
<tr>
<td>Error prop.</td>
<td>1 block</td>
<td>1 block + 1 bit</td>
<td>1 bit</td>
<td>(n) bits + 1 bit</td>
</tr>
<tr>
<td>Synch.</td>
<td>block</td>
<td>block</td>
<td>exact</td>
<td>(j) bits</td>
</tr>
<tr>
<td>Application</td>
<td>key enc.</td>
<td>default</td>
<td>no error prop.</td>
<td>synch.</td>
</tr>
</tbody>
</table>

Modes of operation can be used to construct other symmetric primitives based on block ciphers.

### CTR Mode

**CTR = Counter**

• Proposed more recently.

• Also turns the block cipher into a stream cipher.

• Enables blocks to be processed in parallel.

• Combines many of the advantages of ECB Mode, but with none of the disadvantages.

• Need to select a public IV and a different counter \(i\) for each message encrypted under the fixed key \(k\).

• Encryption: \(c_i = m_i \oplus e_k(IV + i)\)

• Unlike ECB Mode two equal blocks will **not** encrypt to the same ciphertext value.

• Also unlike ECB Mode each ciphertext block corresponds to a precise position within the ciphertext, as its position information is needed to be able to decrypt it successfully.

### CTR Encipherment

Geoff Hamilton