Data Encryption Standard (DES)

Alex Corner

Sheffield Hallam University

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.
- They still didn't necessarily ensure security.

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.
- They still didn't necessarily ensure security.
- The advent of the computer allowed even more complex algorithms to be used, using the `XOR` operation in conjunction with more familiar methods of substitution and transposition.

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.
- They still didn't necessarily ensure security.
- The advent of the computer allowed even more complex algorithms to be used, using the `XOR` operation in conjunction with more familiar methods of substitution and transposition.
- In the 1970s two very different breakthroughs changed the face of cryptography:

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.
- They still didn't necessarily ensure security.
- The advent of the computer allowed even more complex algorithms to be used, using the `XOR` operation in conjunction with more familiar methods of substitution and transposition.
- In the 1970s two very different breakthroughs changed the face of cryptography:
  - **Public Key Cryptography**

The Story So Far

- Mechanical ciphers like Enigma allowed much more complex algorithms to be used for encryption.
- They still didn't necessarily ensure security.
- The advent of the computer allowed even more complex algorithms to be used, using the `XOR` operation in conjunction with more familiar methods of substitution and transposition.
- In the 1970s two very different breakthroughs changed the face of cryptography:
  - **Public Key Cryptography**
  - **DES**

XOR: Exclusive-OR

- There are various ways to write the exclusive-OR. We might write x XOR y, or in logical notation $x \oplus y$. (Even sometimes $\underline{\vee}$.)

XOR: Exclusive-OR

- There are various ways to write the exclusive-OR. We might write x XOR y, or in logical notation $x \oplus y$. (Even sometimes $\veebar$.)
- It is described by the truth table below:

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

XOR: Exclusive-OR

- There are various ways to write the exclusive-OR. We might write x XOR y, or in logical notation $x \oplus y$. (Even sometimes $\veebar$.)
- It is described by the truth table below:

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

- This is in fact the same as performing modular arithmetic using modulus 2: $0 + 0 = 0 = 1 + 1 \bmod 2$ and $1 + 0 = 1 = 0 + 1 \bmod 2$.

Bitwise XOR

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

Bitwise XOR

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

- We can apply this operator to whole strings of bits.

Bitwise XOR

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

- We can apply this operator to whole strings of bits.
- For example, calculating $1010 \oplus 1011$ we perform XOR on the bits individually.

Bitwise XOR

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

- We can apply this operator to whole strings of bits.
- For example, calculating $1010 \oplus 1011$ we perform XOR on the bits individually.
- So $1\,0\,1\,0 \oplus 1\,0\,1\,1 = 0\,0\,0\,1$.

Bitwise XOR

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

- We can apply this operator to whole strings of bits.
- For example, calculating $1010 \oplus 1011$ we perform XOR on the bits individually.
- So $1010 \oplus 1011 = 0001$.
- Digital data is generally stored in binary or hex, using bits, so this type of operation is well-suited for implementation in a computer-aided cipher scheme.

ASCII: American Standard Code for Information Interchange

DEC	OCT	HEX	BIN	Symbol	Description
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G

- A standard way to encode text and other characters. Each character is represented by a 7-bit binary string.

ASCII: American Standard Code for Information Interchange

DEC	OCT	HEX	BIN	Symbol	Description
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G

- A standard way to encode text and other characters. Each character is represented by a 7-bit binary string.
- Microsoft Excel uses 8-bit ANSI as above - not *quite* the same mapping of characters to strings.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve."

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.
- The original idea of a **one-time pad** was developed in 1918 by Vernam and Mauborgne, implemented mechanically.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.
- The original idea of a **one-time pad** was developed in 1918 by Vernam and Mauborgne, implemented mechanically.
- The idea is to write the plaintext in binary, e.g., using ASCII.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.
- The original idea of a **one-time pad** was developed in 1918 by Vernam and Mauborgne, implemented mechanically.
- The idea is to write the plaintext in binary, e.g., using ASCII.
- Then a random key of 0s and 1s is chosen which is the same length as the plaintext.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.
- The original idea of a **one-time pad** was developed in 1918 by Vernam and Mauborgne, implemented mechanically.
- The idea is to write the plaintext in binary, e.g., using ASCII.
- Then a random key of 0s and 1s is chosen which is the same length as the plaintext.
- Encryption is performed via the `XOR` operation and the key is then discarded.

The One-Time Pad

- Edgar Allen Poe: "Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Wrong.
- The original idea of a **one-time pad** was developed in 1918 by Vernam and Mauborgne, implemented mechanically.
- The idea is to write the plaintext in binary, e.g., using ASCII.
- Then a random key of 0s and 1s is chosen which is the same length as the plaintext.
- Encryption is performed via the `XOR` operation and the key is then discarded.
- Shannon and Kotelnikov had both showed that this encryption scheme is *theoretically* unbreakable.

The One-Time Pad

- Theoretically unbreakable, but not really very practical.

The One-Time Pad

- Theoretically unbreakable, but not really very practical.
  1. There is a problem generating a truly random sequence. Computers can never produce truly random numbers, only **pseudo-random** numbers, which are prone to attack.

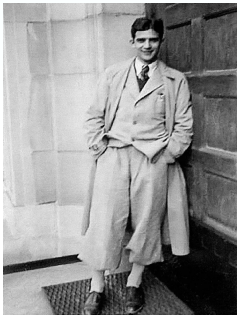
The One-Time Pad

- Theoretically unbreakable, but not really very practical.
  1. There is a problem generating a truly random sequence. Computers can never produce truly random numbers, only **pseudo-random** numbers, which are prone to attack.
  2. It requires a very long key which is computationally expensive to produce and difficult to distribute securely.

The One-Time Pad

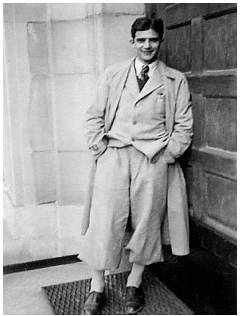
- Theoretically unbreakable, but not really very practical.
  1. There is a problem generating a truly random sequence. Computers can never produce truly random numbers, only **pseudo-random** numbers, which are prone to attack.
  2. It requires a very long key which is computationally expensive to produce and difficult to distribute securely.
  3. A new key is required each time.

Development of DES



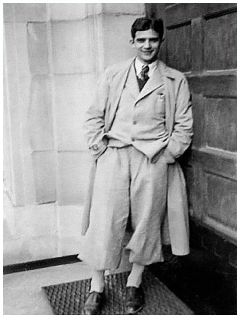
- **DES: D**ata **E**ncryption **S**tandard.

Development of DES



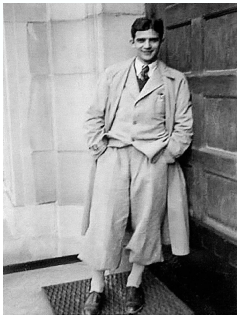
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.

Development of DES



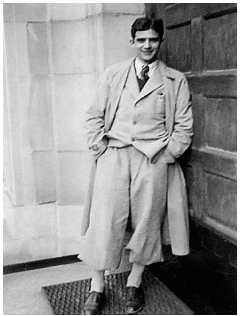
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.

Development of DES



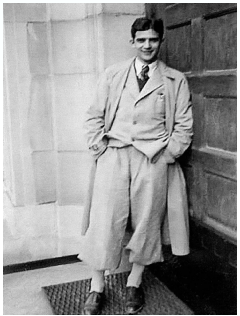
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal

Development of DES



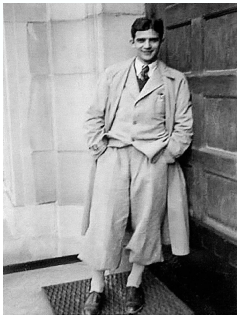
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal $\rightarrow$ Demonstration Cipher

Development of DES



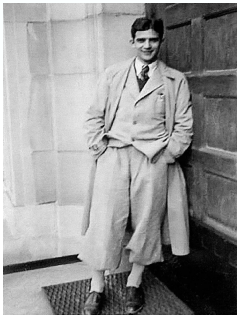
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal $\rightarrow$ Demonstration Cipher $\rightarrow$ Demon

Development of DES



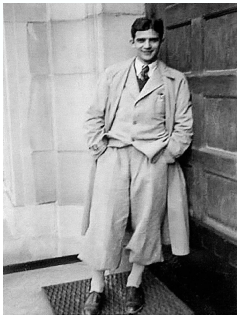
- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal $\rightarrow$ Demonstration Cipher $\rightarrow$ Demon $\rightarrow$ Lucifer

Development of DES



- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal $\rightarrow$ Demonstration Cipher $\rightarrow$ Demon $\rightarrow$ Lucifer $\rightarrow$ DES.

Development of DES



- **DES: D**ata **E**ncryption **S**tandard.
- Developed by Horst Feistel at IBM.
- The National Bureau of Standards held a competition in 1974: IBM submitted Lucifer.
- Dataseal $\rightarrow$ Demonstration Cipher $\rightarrow$ Demon $\rightarrow$ Lucifer $\rightarrow$ DES.
- Made the official worldwide standard in 1977.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.
- It was 56 bits compared to IBM's original Lucifer key size of 128 bits.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.
- It was 56 bits compared to IBM's original Lucifer key size of 128 bits.
- Others believed the National Security Agency (NSA) had included a **trapdoor**: a secret weakness which could be exploited by the security agencies.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.
- It was 56 bits compared to IBM's original Lucifer key size of 128 bits.
- Others believed the National Security Agency (NSA) had included a **trapdoor**: a secret weakness which could be exploited by the security agencies.
- The most widely used cryptographic method ever designed - largely used for secure financial transactions by banks.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.
- It was 56 bits compared to IBM's original Lucifer key size of 128 bits.
- Others believed the National Security Agency (NSA) had included a **trapdoor**: a secret weakness which could be exploited by the security agencies.
- The most widely used cryptographic method ever designed - largely used for secure financial transactions by banks.
- Expected to be used for 10 years from 1977.

DES: A Controversial Choice

- A large key size or length is an important part of a secure cryptographic system.
- Some people, such as Whitfield Diffie and Martin Hellman, believed the DES key size to be too small to be secure.
- It was 56 bits compared to IBM's original Lucifer key size of 128 bits.
- Others believed the National Security Agency (NSA) had included a **trapdoor**: a secret weakness which could be exploited by the security agencies.
- The most widely used cryptographic method ever designed - largely used for secure financial transactions by banks.
- Expected to be used for 10 years from 1977. Triple-DES, or 3DES, developed in 1981 to be more secure, but is now also disallowed as of 2023.

Block Ciphers vs. Stream Ciphers

- The ciphers we have seen so far have all been **stream ciphers**: each letter is encrypted separately.

Block Ciphers vs. Stream Ciphers

- The ciphers we have seen so far have all been **stream ciphers**: each letter is encrypted separately.
- DES and many modern ciphers are a form of **block cipher**: how letters are encrypted depends on the letters around them.

Block Ciphers vs. Stream Ciphers

- The ciphers we have seen so far have all been **stream ciphers**: each letter is encrypted separately.
- DES and many modern ciphers are a form of **block cipher**: how letters are encrypted depends on the letters around them.
- **Confusion:** Obscures the relationship between the key and the ciphertext - typically achieved with substitution.

Block Ciphers vs. Stream Ciphers

- The ciphers we have seen so far have all been **stream ciphers**: each letter is encrypted separately.
- DES and many modern ciphers are a form of **block cipher**: how letters are encrypted depends on the letters around them.
- **Confusion:** Obscures the relationship between the key and the ciphertext - typically achieved with substitution.
- **Diffusion:** Spreads the influence of one plaintext bit over many ciphertext bits to hide statistical properties of the plaintext - typically achieved with transposition.

Block Ciphers vs. Stream Ciphers

- The ciphers we have seen so far have all been **stream ciphers**: each letter is encrypted separately.
- DES and many modern ciphers are a form of **block cipher**: how letters are encrypted depends on the letters around them.
- **Confusion:** Obscures the relationship between the key and the ciphertext - typically achieved with substitution.
- **Diffusion:** Spreads the influence of one plaintext bit over many ciphertext bits to hide statistical properties of the plaintext - typically achieved with transposition.
- **Multiple Rounds:** Intended to amplify the effect of confusion and diffusion.

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

0101100 0001111 1011110 1001010 1101001 1010010 0101001 1110101

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

  0101100 0001111 1011110 1001010 1101001 1010010 0101001 1110101

- The first seven bits (on the left) are 0101100.

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

0101100 0001111 1011110 1001010 1101001 1010010 0101001 1110101

- The first seven bits (on the left) are 0101100.
- We put an extra bit on the *right* to make an *odd* number of 1s in each 8-bit byte:

0101100

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

0101100 0001111 1011110 1001010 1101001 1010010 0101001 1110101

- The first seven bits (on the left) are 0101100.
- We put an extra bit on the *right* to make an *odd* number of 1s in each 8-bit byte:

01011000.

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

$$0101100\,0001111\,1011110\,1001010\,1101001\,1010010\,0101001\,1110101$$

- The first seven bits (on the left) are 0101100.
- We put an extra bit on the *right* to make an *odd* number of 1s in each 8-bit byte:

$$01011000.$$

- In hex this is $0101\,1000_2 = 58_{16}$.

Generating a Key

- Keys in DES are of length 56, but are expanded by **parity bits** to be 64 bits.

$$0101100\,0001111\,1011110\,1001010\,1101001\,1010010\,0101001\,1110101$$

- The first seven bits (on the left) are 0101100.
- We put an extra bit on the *right* to make an *odd* number of 1s in each 8-bit byte:

$$01011000.$$

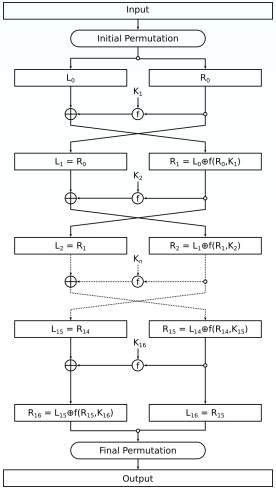
- In hex this is $0101\,1000_2 = 58_{16}$.
- This is done for all bytes, which eventually gives the key as:

$$K = 01011000\,00011111\,10111100\,10010100\,11010011\,10100100\,01010010\,11101010$$

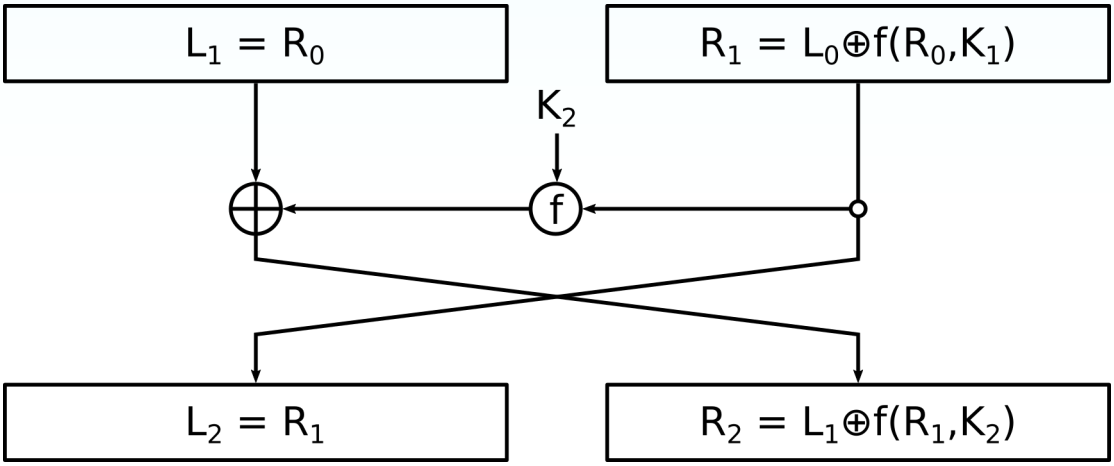
$$K = 58 \quad 1F \quad BC \quad 94 \quad D3 \quad A4 \quad 52 \quad EA.$$

- DES consists of 16 **rounds**, each using a 48-bit **round key** K_i derived from the original 56-bit key K.

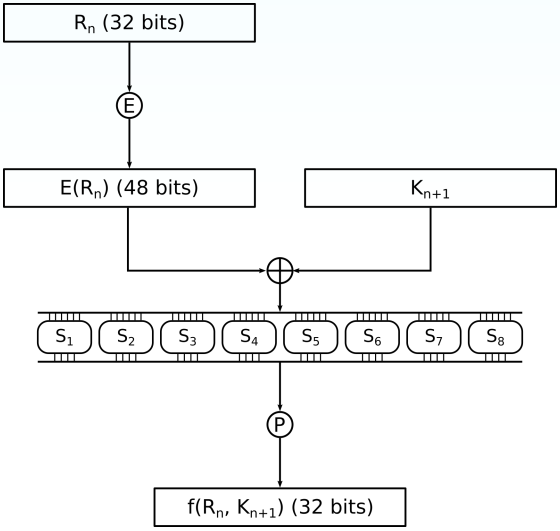
DES Scheme


Input
Initial Permutation
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
f
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
K_n
f
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
f
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
Final Permutation
Output

DES Scheme: Single Round



DES Scheme: Mangler Function



DES Scheme: Right Expansion

A 32-bit *right half* is expanded to 48-bits in the following way, by repeating some of the bits.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES Scheme: Right Expansion

A 32-bit *right half* is expanded to 48-bits in the following way, by repeating some of the bits.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

For example, if $R = 0000\,0001\,0010\,0011\,0100\,0101\,0110\,0111$, then

$$E(R) = 100000\,000010\,100100\,000110\,101000\,001010\,101100\,001110.$$

DES Scheme: XOR with Round Key

After the right half is expanded, it is XORed with the current round key. E.g., the first 48-bit round key K_1 from our example key is

$$001001\,111010\,000101\,101001\,111001\,011000\,110111\,011010.$$

When XORed with the expanded right half

$$100000\,000010\,100100\,000110\,101000\,001010\,101100\,001110$$

we get

$$101001\,111000\,100001\,101111\,010001\,010010\,011011\,010100.$$

DES Scheme: S-Boxes

S_7	**Column Number**															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$$K_1 \oplus E(R) = 100111\,100010\,101001\,010011\,010100\,000110\,011110\,100100.$$

- Each block passes through a separate S-box.

DES Scheme: S-Boxes

S_7	**Column Number**															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$$K_1 \oplus E(R) = 100111\,100010\,101001\,010011\,010100\,000110\,011110\,100100.$$

- Each block passes through a separate S-box.
- A block $b_1 b_2 b_3 b_4 b_5 b_6$ is split into a pair $(b_1 b_6, b_2 b_3 b_4 b_5)$.

DES Scheme: S-Boxes

S_7	**Column Number**															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$$K_1 \oplus E(R) = 100111\,100010\,101001\,010011\,010100\,000110\,011110\,100100.$$

- Each block passes through a separate S-box.
- A block $b_1b_2b_3b_4b_5b_6$ is split into a pair $(b_1b_6, b_2b_3b_4b_5)$.
- The left part is a 2-bit binary number (decimal 0-3), the right part is a 4-bit binary number (decimal 0-15).

DES Scheme: S-Boxes

S_7	**Column Number**															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$$K_1 \oplus E(R) = 100111\,100010\,101001\,010011\,010100\,000110\,011110\,100100.$$

- Each block passes through a separate S-box.
- A block $b_1b_2b_3b_4b_5b_6$ is split into a pair $(b_1b_6, b_2b_3b_4b_5)$.
- The left part is a 2-bit binary number (decimal 0-3), the right part is a 4-bit binary number (decimal 0-15).
- These give the row and column to look up in the S-box.

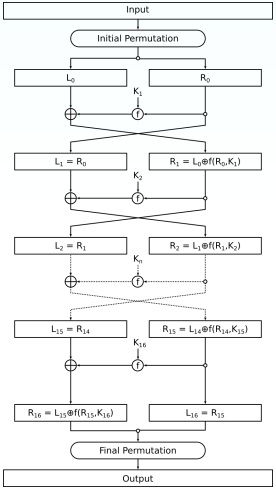
DES Scheme: S-Boxes

S_7	Column Number															
Row No.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$$K_1 \oplus E(R) = 100111\,100010\,101001\,010011\,010100\,000110\,011110\,100100.$$

- Each block passes through a separate S-box.
- A block $b_1b_2b_3b_4b_5b_6$ is split into a pair $(b_1b_6, b_2b_3b_4b_5)$.
- The left part is a 2-bit binary number (decimal 0-3), the right part is a 4-bit binary number (decimal 0-15).
- These give the row and column to look up in the S-box.
- Block seven (011110) becomes $(00, 1111) = (0, 15)$, giving $1 = 0001$.

DES Scheme


Input
Initial Permutation
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
f
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
K_n
f
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
f
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
Final Permutation
Output

Feistel System and Decryption

- The DES scheme is an example of a **Feistel system**:
  - Ladder structure;
  - Input split into left and right halves;
  - A round function of one half is computed and combined with the other half using bitwise `XOR`;
  - The halves are swapped over.

Feistel System and Decryption

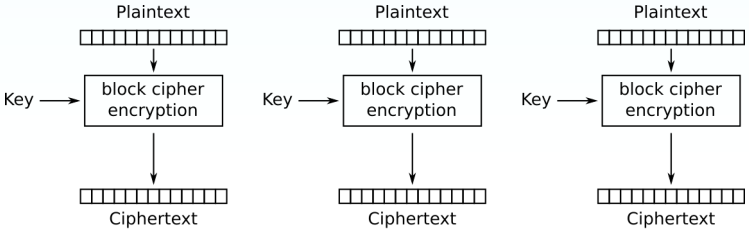
- The DES scheme is an example of a **Feistel system**:
  - Ladder structure;
  - Input split into left and right halves;
  - A round function of one half is computed and combined with the other half using bitwise `XOR`;
  - The halves are swapped over.
- To decrypt the ciphertext: retrace all the steps in the 16 rounds using the same key K, but with the round keys used in the opposite order.

Modes of Operation

Five standard ways of implementing DES:

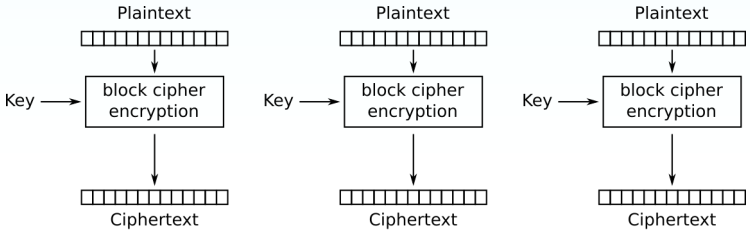
- Mode 1: Electronic Code Book
- Mode 2: Cipher Block Chaining
- Mode 3: Cipher Feedback Mode
- Mode 4: Output Feedback Mode
- Mode 5: Counter Mode

Modes of Operation: Electronic Code Book



Electronic Codebook (ECB) mode encryption

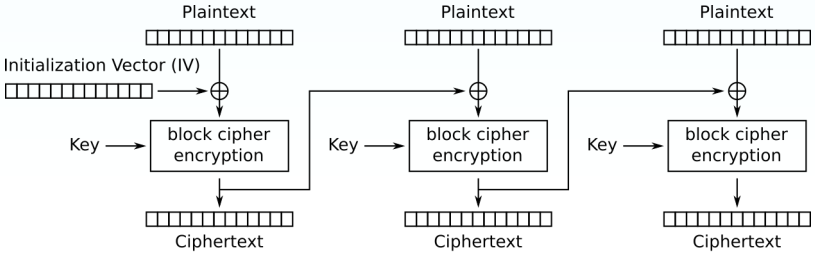
Modes of Operation: Electronic Code Book



Electronic Codebook (ECB) mode encryption

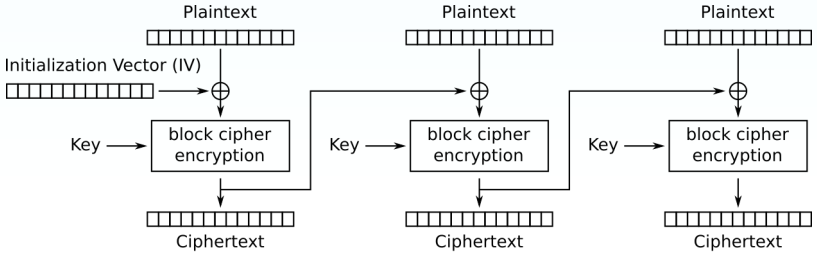
ECB encrypts the blocks one after another, but is vulnerable to attack for long messages because of plaintext patterns.

Modes of Operation: Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

Modes of Operation: Cipher Block Chaining

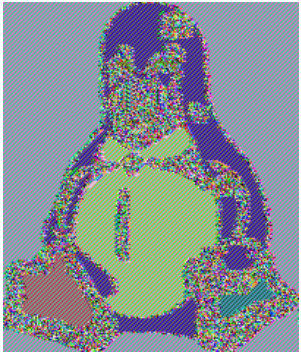


Cipher Block Chaining (CBC) mode encryption

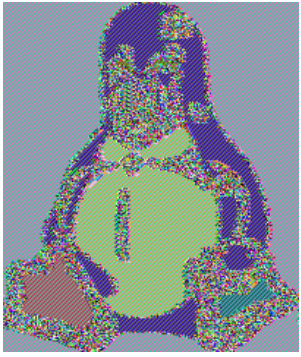
CBC uses the previous ciphertext block `XOR`ed with the new plaintext block before proceeding with encryption. An Initialisation Vector (IV) is used to `XOR` the first plaintext block. ECB is better at hiding plaintext patterns than ECB.

Modes of Operation: ECB vs Other Modes

Modes of Operation: ECB vs Other Modes



Modes of Operation: ECB vs Other Modes



Breaking DES

- DES was expected to be used for 10 years: 1977-1987.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.
- By the 2000s, computers were 100,000 times more powerful making brute force attacks quite straightforward.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.
- By the 2000s, computers were 100,000 times more powerful making brute force attacks quite straightforward.
- A (second) challenge was issued in 1998 by the RSA Laboratory, with a $\$10,000$USD prize:
  - Electronic Frontier Foundation's DES Cracker, working as a botnet, took only 3 days to crack DES.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.
- By the 2000s, computers were 100,000 times more powerful making brute force attacks quite straightforward.
- A (second) challenge was issued in 1998 by the RSA Laboratory, with a $\$10,000$USD prize:
  - Electronic Frontier Foundation's DES Cracker, working as a botnet, took only 3 days to crack DES.
  - The previous record was 39 days.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.
- By the 2000s, computers were 100,000 times more powerful making brute force attacks quite straightforward.
- A (second) challenge was issued in 1998 by the RSA Laboratory, with a $\$10,000$USD prize:
  - Electronic Frontier Foundation's DES Cracker, working as a botnet, took only 3 days to crack DES.
  - The previous record was 39 days.
  - In 1999, a network of 100,000 PCs on the Internet cracked DES in 22 hours and 15 minutes.

Breaking DES

- DES was expected to be used for 10 years: 1977-1987.
- It remained the standard **secret key** encryption method until 1998.
- Ongoing debates about security of DES focussed on the short key length.
- Brute force attacks were theoretically possible, but the hardware in the 1970s would take too long to make this an efficient attack.
- In the 1980s rumours circulated that intelligence agencies had already built machines to search for DES keys.
- By the 2000s, computers were 100,000 times more powerful making brute force attacks quite straightforward.
- A (second) challenge was issued in 1998 by the RSA Laboratory, with a $\$10,000$USD prize:
  - Electronic Frontier Foundation's DES Cracker, working as a botnet, took only 3 days to crack DES.
  - The previous record was 39 days.
  - In 1999, a network of 100,000 PCs on the Internet cracked DES in 22 hours and 15 minutes.
  - DES needed to be replaced.

Double-DES: 2DES

- How do we make DES more secure?

Double-DES: 2DES

- How do we make DES more secure? Run it twice: Double DES.

Double-DES: 2DES

- How do we make DES more secure? Run it twice: Double DES.
- The idea here is that this would double the keyspace from 2^{56} options to 2^{112}.

Double-DES: 2DES

- How do we make DES more secure? Run it twice: Double DES.
- The idea here is that this would double the keyspace from 2^{56} options to 2^{112}.
- Unfortunately, due to **meet-in-the-middle attacks**, this actually only increased the number of keys to 2^{57}.

Triple-DES: 3DES

- How do we make DES more secure?

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.
- Uses three independent keys: K_1, K_2, K_3.

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.
- Uses three independent keys: K_1, K_2, K_3.
- $m \mapsto DES(m, K_1) \mapsto DES^{-1}(DES(m, K_1), K_2) \mapsto DES(DES^{-1}(DES(m, K_1), K_2, K_3))$

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.
- Uses three independent keys: K_1, K_2, K_3.
- $m \mapsto DES(m, K_1) \mapsto DES^{-1}(DES(m, K_1), K_2) \mapsto DES(DES^{-1}(DES(m, K_1), K_2, K_3))$
- If $K_1 = K_2 = K_3$, this is the same as single DES (good for backwards compatibility).

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.
- Uses three independent keys: K_1, K_2, K_3.
- $m \mapsto DES(m, K_1) \mapsto DES^{-1}(DES(m, K_1), K_2) \mapsto DES(DES^{-1}(DES(m, K_1), K_2, K_3))$
- If $K_1 = K_2 = K_3$, this is the same as single DES (good for backwards compatibility).
- Often used with two keys, setting $K_1 = K_3$, but keeping K_2 independent.

Triple-DES: 3DES

- How do we make DES more secure? Run it *three* times.
- Uses three independent keys: K_1, K_2, K_3.
- $m \mapsto DES(m, K_1) \mapsto DES^{-1}(DES(m, K_1), K_2) \mapsto DES(DES^{-1}(DES(m, K_1), K_2, K_3))$
- If $K_1 = K_2 = K_3$, this is the same as single DES (good for backwards compatibility).
- Often used with two keys, setting $K_1 = K_3$, but keeping K_2 independent.
- Much stronger against meet-in-the-middle attacks.

Triple-DES: 3DES

- 3DES encryption was the standard for ATMs: all machines had to be compliant by 2005.

Triple-DES: 3DES

- 3DES encryption was the standard for ATMs: all machines had to be compliant by 2005.
- CVE-2016-2183 (Critical Vulnerabilities and Exposures) released in 2016 describes the 'Sweet32' attack (a type of **birthday attack**, which shows that even Triple DES was not sufficient any more.

Triple-DES: 3DES

- 3DES encryption was the standard for ATMs: all machines had to be compliant by 2005.
- CVE-2016-2183 (Critical Vulnerabilities and Exposures) released in 2016 describes the 'Sweet32' attack (a type of **birthday attack**, which shows that even Triple DES was not sufficient any more.
- All applications of 3DES were to cease by 2023.

Triple-DES: 3DES

- 3DES encryption was the standard for ATMs: all machines had to be compliant by 2005.
- CVE-2016-2183 (Critical Vulnerabilities and Exposures) released in 2016 describes the 'Sweet32' attack (a type of **birthday attack**, which shows that even Triple DES was not sufficient any more.
- All applications of 3DES were to cease by 2023.
- DES finally superseded by the Advanced Encryption Standard (AES).

Tutorials

In the tutorial this week we will:

- Create a spreadsheet to perform bitwise XOR encryption.
- Expand this to XOR whole words at a time.