Stream Ciphers and Block Ciphers

2MMC10 Cryptology – Fall 2015

Ruben Niederhagen

Technische Universiteit
Eindhoven
University of Technology

October 6th, 2015

Introduction

Recall:

- **Public-key crypto:**
  - Pair of keys: public key for encryption, private key for decryption.

Introduction

Recall:

- **Public-key crypto:**
  - Pair of keys: public key for encryption, private key for decryption.
- **Symmetric-key crypto:**
  - Same shared secret key for encryption and decryption.

Recall:

- **Public-key crypto:**
  - Pair of keys: public key for encryption, private key for decryption.
- **Symmetric-key crypto:**
  - Same shared secret key for encryption and decryption.

What are the respective advantages, disadvantages, use-cases..?

Introduction: Symmetric-Key Crypto

Stream Cipher vs. Block Cipher:

- **Idea of a stream cipher:** partition the text into small (e.g. 1bit) blocks; encoding of each block depends on the previous blocks.
  $\rightarrow$ A different "key" is generated for each block.

Introduction: Symmetric-Key Crypto

Stream Cipher vs. Block Cipher:

- **Idea of a stream cipher:** partition the text into small (e.g. 1bit) blocks; encoding of each block depends on the previous blocks.
  $\rightarrow$ A different "key" is generated for each block.
- **Idea of a block cipher:** partition the text into "large" (e.g. 128bit) blocks; encode each block independently.
  $\rightarrow$ The same "key" is used for each block.

Introduction: Symmetric-Key Crypto

Stream Ciphers:

- symmetric-key cipher,
- state-driven: operates on arbitrary message length,
- commonly used stream ciphers: A5/1 and A5/2 (GSM), RC4 (SSL, WEP), eSTREAM Project.

Introduction: Symmetric-Key Crypto

Stream Ciphers:

- symmetric-key cipher,
- state-driven: operates on arbitrary message length,
- commonly used stream ciphers: A5/1 and ~~A5/2~~ (GSM), ~~RC4~~ (SSL, WEP), eSTREAM Project.

Introduction: Symmetric-Key Crypto

Stream Ciphers:

- symmetric-key cipher,
- state-driven: operates on arbitrary message length,
- commonly used stream ciphers: A5/1 and ~~A5/2~~ (GSM), ~~RC4~~ (SSL, WEP), eSTREAM Project.

Operate on an *internal state* which is updated after each block.

Stream Ciphers:

- symmetric-key cipher,
- state-driven: operates on arbitrary message length,
- commonly used stream ciphers: A5/1 and ~~A5/2~~ (GSM), ~~RC4~~ (SSL, WEP), eSTREAM Project.

Operate on an *internal state* which is updated after each block.
Compute a *key stream* using the current state and the secret key.

Introduction: Symmetric-Key Crypto

Stream Ciphers:

- symmetric-key cipher,
- state-driven: operates on arbitrary message length,
- commonly used stream ciphers: A5/1 and ~~A5/2~~ (GSM), ~~RC4~~ (SSL, WEP), eSTREAM Project.

Operate on an *internal state* which is updated after each block.
Compute a *key stream* using the current state and the secret key.
Encrypt the *message stream* with the *key stream* to a *cipher stream*.

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state: $\sigma_i = f(\sigma_{i-1}, K)$ with next-state function f

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$h(z_i, m_i)$	with output function h

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$h(z_i, m_i)$	with output function h
decryption:	m_i	$=$	$h^{-1}(z_i, c_i)$	with inverse of h

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Self-Synchronizing Stream Ciphers:

Given key K and initial state $c_{-1} \ldots c_{-t}$:

state:	σ_i	$=$	$(c_{i-1}, c_{i-2}, \ldots c_{i-t})$

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Self-Synchronizing Stream Ciphers:

Given key K and initial state $c_{-1} \ldots c_{-t}$:

state:	σ_i	$=$	$(c_{i-1}, c_{i-2}, \ldots c_{i-t})$	
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Self-Synchronizing Stream Ciphers:

Given key K and initial state $c_{-1} \ldots c_{-t}$:

state:	σ_i	$=$	$(c_{i-1}, c_{i-2}, \ldots c_{i-t})$	
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$h(z_i, m_i)$	with output function h

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Self-Synchronizing Stream Ciphers:

Given key K and initial state $c_{-1} \ldots c_{-t}$:

state:	σ_i	$=$	$(c_{i-1}, c_{i-2}, \ldots c_{i-t})$	
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$h(z_i, m_i)$	with output function h
decryption:	m_i	$=$	$h^{-1}(z_i, c_i)$	with inverse of h

Introduction: Symmetric-Key Crypto

Synchronous Stream Ciphers:

Given key K and initial state σ_{-1}:

state:	σ_i	$=$	$f(\sigma_{i-1}, K)$	with next-state function f
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Self-Synchronizing Stream Ciphers:

Given key K and initial state $c_{-1} \dots c_{-t}$:

state:	σ_i	$=$	$(c_{i-1}, c_{i-2}, \dots c_{i-t})$	
key stream:	z_i	$=$	$g(\sigma_i, K)$	with key-stream function g
cipher stream:	c_i	$=$	$z_i \oplus m_i$	
decryption:	m_i	$=$	$z_i \oplus c_i$	

Introduction: Symmetric-Key Crypto

Block Ciphers:

- symmetric-key cipher,
- memoryless: operates on a fixed-length block size,
- commonly used block ciphers: DES, Triple-DES, AES.

Introduction: Symmetric-Key Crypto

Block Ciphers:

- symmetric-key cipher,
- memoryless: operates on a fixed-length block size,
- commonly used block ciphers: ~~DES~~, Triple-DES, AES.

Introduction: Symmetric-Key Crypto

Block Ciphers:

- symmetric-key cipher,
- memoryless: operates on a fixed-length block size,
- commonly used block ciphers: ~~DES~~, Triple-DES, AES.

An n-bit block cipher is a function $E : \{0,1\}^n \times \mathfrak{K} \to \{0,1\}^n$.
For each fixed key $K \in \mathfrak{K}$ the map

$$E_K : \{0,1\}^n \to \{0,1\}^n, M \mapsto E_K(M)$$

is invertible (bijective) with inverse $E_K^{-1} : \{0,1\}^n \to \{0,1\}^n$.

From Block Ciphers to Stream Ciphers

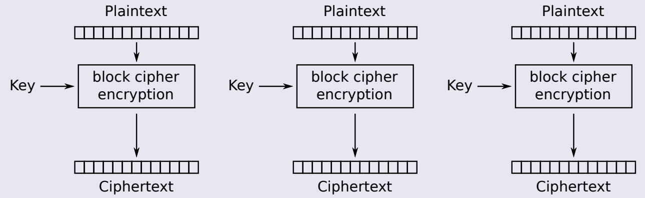
Modes of Operation:

- electronic codebook (ECB) mode,
- cipher-block chaining (CBC) mode,
- cipher feedback (CFB) mode,
- output feedback (OFB) mode,
- counter (CTR) mode.

Modes of Operation

Electronic Codebook (ECB) Mode:

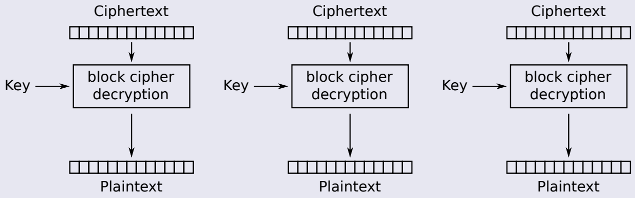
- Encryption:
  obtain ciphertext $C_1, \ldots, C_t$ as
  $C_i = E_K(M_i), \; i = 1 \ldots t$



Electronic Codebook (ECB) mode encryption

Electronic Codebook (ECB) Mode:

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as
  $M_i = E_K^{-1}(C_i),\; i = 1 \ldots t$



Electronic Codebook (ECB) mode decryption

Electronic Codebook (ECB) Mode:

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as

Plaintext Plaintext Plaintext

Electronic Codebook (ECB) mode decryption

Modes of Operation

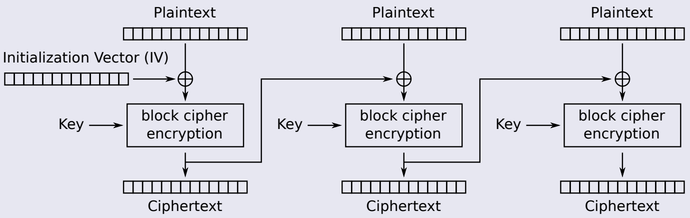
Properties of ECB:

- Considered insecure if applied to more than one block!
- Each block of ciphertext C_i depends only on message block M_i,
- encryption and decryption can be performed in parallel,
- allows random read access for decryption,
- requires padding of input to a multiple of block size.

Cipher-Block Chaining (CBC) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Encryption:
  obtain ciphertext $C_1, \ldots, C_t$ as
  $C_i = E_K(M_i \oplus C_{i-1}),\ i = 1 \ldots t,\ C_0 = IV$

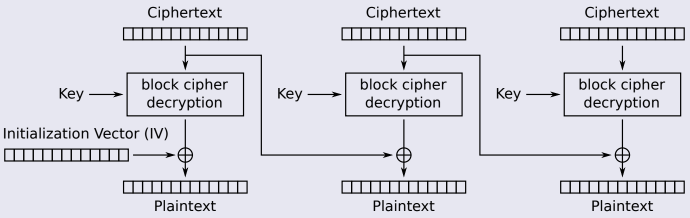


Cipher Block Chaining (CBC) mode encryption

Cipher-Block Chaining (CBC) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as
  $M_i = E_K^{-1}(C_i) \oplus C_{i-1},\ i = 1 \ldots t,\ C_0 = IV$



Cipher Block Chaining (CBC) mode decryption

Properties of CBC:

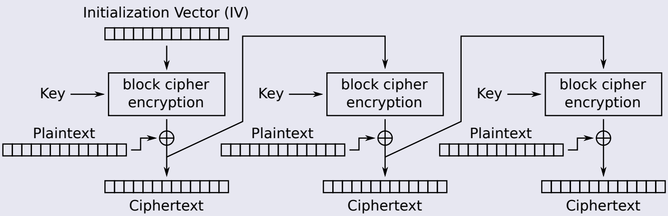
- The last ciphertext block C_t depends all message blocks $M_1, \ldots, M_t$,
- encryption can *not* be performed in parallel but decryption can,
- *no* random read access for decryption,
- requires padding of input to a multiple of block size.

Modes of Operation

Cipher Feedback (CFB) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Encryption:
  obtain ciphertext $C_1, \ldots, C_t$ as
  $C_i = E_K(C_{i-1}) \oplus M_i, \; i = 1 \ldots t, \; C_0 = IV$



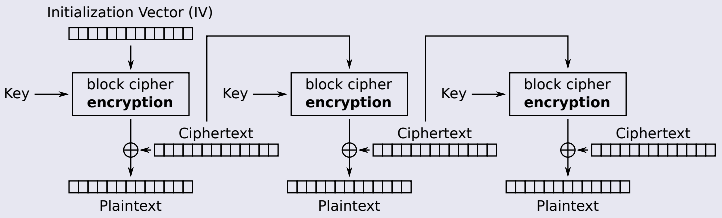
Cipher Feedback (CFB) mode encryption

Modes of Operation

Cipher Feedback (CFB) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as
  $M_i = E_K(C_{i-1}) \oplus C_i,\ i = 1 \ldots t,\ C_0 = IV$



Cipher Feedback (CFB) mode decryption

Modes of Operation

Properties of CFB:

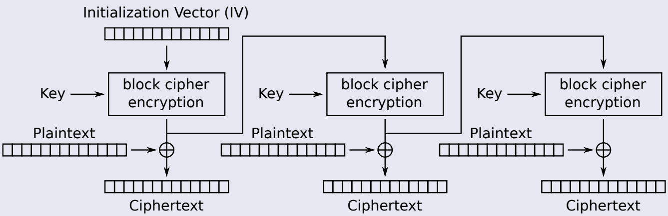
- The last ciphertext block C_t depends all message blocks $M_1, \ldots, M_i$,
- encryption can *not* be performed in parallel but decryption can,
- *no* random access for decryption,
- does *not* require padding of plaintext.
- Two messages encrypted with the same key must use a different IV!

Modes of Operation

Output Feedback (OFB) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Encryption:
  obtain ciphertext $C_1, \ldots, C_t$ as
  $C_i = O_i \oplus M_i,\ i = 1 \ldots t,\ O_i = E_K(O_{i-1}),\ O_0 = IV$

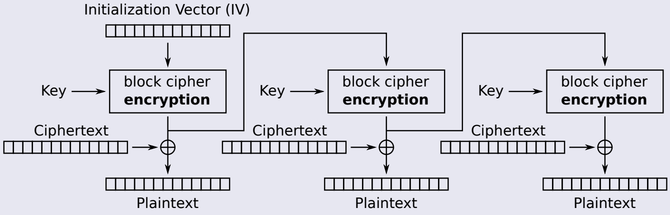


Output Feedback (OFB) mode encryption

Output Feedback (OFB) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as
  $M_i = O_i \oplus C_i,\ i = 1 \ldots t,\ O_i = E_K(O_{i-1}),\ O_0 = IV$



Output Feedback (OFB) mode decryption

Modes of Operation

Properties of OFB:

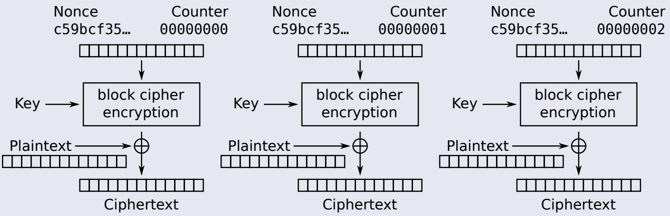
- Each block of ciphertext C_i depends only on message block M_i,
- encryption and decryption can *not* be performed in parallel,
- *no* random access for decryption,
- does *not* require padding of plaintext.
- Two messages encrypted with the same key must use a different IV!

Modes of Operation

Counter (CTR) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Encryption:
  obtain ciphertext $C_1, \ldots, C_t$ as
  $C_i = E_K(N_i) \oplus M_i,\ i = 1 \ldots t,\ N_i = N_{i-1} + 1 \mod 2^n,\ N_0 = IV$



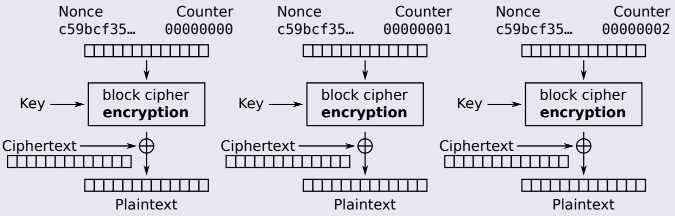
Counter (CTR) mode encryption

Modes of Operation

Counter (CTR) Mode:

Use a (non-secret) initialization vector (IV) of length n bits.

- Decryption:
  obtain plaintext $M_1, \ldots, M_t$ as
  $M_i = E_K(N_i) \oplus C_i,\ i = 1 \ldots t,\ N_i = N_{i-1} + 1 \mod 2^n,\ N_0 = IV$



Counter (CTR) mode decryption

Properties of CTR:

- Each block of ciphertext C_i depends only on message block M_i,
- both encryption and decryption *can* be performed in parallel,
- *allows* random access for decryption,
- does *not* require padding of plaintext.
- Two messages encrypted with the same key must use a different IV!

Properties of CTR:

- Each block of ciphertext C_i depends only on message block M_i,
- both encryption and decryption *can* be performed in parallel,
- *allows* random access for decryption,
- does *not* require padding of plaintext.
- Two messages encrypted with the same key must use a different IV!

Most widely used modes are CBC and CTR.

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms.

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms. IBM submits a candidate based on Horst Feistel's Lucifer cipher.

Technische Universiteit Eindhoven University of Technology

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms. IBM submits a candidate based on Horst Feistel's Lucifer cipher.
- **March 1975:** DES proposal is published in the Federal Register. Public comments are requested.

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms. IBM submits a candidate based on Horst Feistel's Lucifer cipher.
- **March 1975:** DES proposal is published in the Federal Register. Public comments are requested. Criticism from various researchers (e.g., Hellman and Diffie) about modifications in respect to the submission:
  - shorter keylength (56bit instead of 64bit),
  - modified "S-boxes".

Technische Universiteit Eindhoven University of Technology

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms. IBM submits a candidate based on Horst Feistel's Lucifer cipher.
- **March 1975:** DES proposal is published in the Federal Register. Public comments are requested. Criticism from various researchers (e.g., Hellman and Diffie) about modifications in respect to the submission:
  - shorter keylength (56bit instead of 64bit),
  - modified "S-boxes".
- **July 1991:** Biham and Shamir (re-)discover differential cryptanalysis that requires 2^{47} chosen plaintexts.

An Example for Block Ciphers: DES

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **August 1974:** NBS publishes a second request algorithms. IBM submits a candidate based on Horst Feistel's Lucifer cipher.
- **March 1975:** DES proposal is published in the Federal Register. Public comments are requested. Criticism from various researchers (e.g., Hellman and Diffie) about modifications in respect to the submission:
  - shorter keylength (56bit instead of 64bit),
  - modified "S-boxes".
- **July 1991:** Biham and Shamir (re-)discover differential cryptanalysis that requires 2^{47} chosen plaintexts. New S-boxes are stronger than the original S-boxes.

History:

- **May 1973:** NBS (NIST) publishes a first request for a standard encryption algorithm.
- **Aug** IBM … her.
- **Ma** … ster. Pub Crit … about mod
  - •
  - modified "S-boxes".

"NSA worked closely with IBM to strengthen the algorithm against all except brute force attacks and to strengthen substitution tables, called S-boxes. Conversely, NSA tried to convince IBM to reduce the length of the key from 64 to 48 bits. Ultimately they compromised on a 56-bit key."

- **July 1991:** Biham and Shamir (re-)discover differential cryptanalysis that requires 2^{47} chosen plaintexts.
  New S-boxes are stronger than the original S-boxes.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.
- **Oct. 1999:** DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the use of Triple DES (DES only for legacy systems).

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.
- **Oct. 1999:** DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the use of Triple DES (DES only for legacy systems).
- **May 2002:** Advanced Encryption Standard (AES) becomes effective.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.
- **Oct. 1999:** DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the use of Triple DES (DES only for legacy systems).
- **May 2002:** Advanced Encryption Standard (AES) becomes effective.
- **May 2005:** NIST withdraws FIPS 46-3.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.
- **Oct. 1999:** DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the use of Triple DES (DES only for legacy systems).
- **May 2002:** Advanced Encryption Standard (AES) becomes effective.
- **May 2005:** NIST withdraws FIPS 46-3.
- **April 2006:** The FPGA based COPACOBANA of the Universities of Bochum and Kiel breaks DES in 9 days at $10,000 hardware cost.

An Example for Block Ciphers: DES

History:

- **in 1994:** First experimental cryptanalysis using linear cryptanalysis.
- **June 1997:** The DESCHALL Project breaks a DES key in 96 days.
- **July 1998:** The EFF's DES cracker breaks a DES key in 56 hours.
- **Oct. 1999:** DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the use of Triple DES (DES only for legacy systems).
- **May 2002:** Advanced Encryption Standard (AES) becomes effective.
- **May 2005:** NIST withdraws FIPS 46-3.
- **April 2006:** The FPGA based COPACOBANA of the Universities of Bochum and Kiel breaks DES in 9 days at $10,000 hardware cost.
- **Nov. 2008:** The successor of COPACOBANA, the RIVYERA machine reduces the average time to less than one single day.

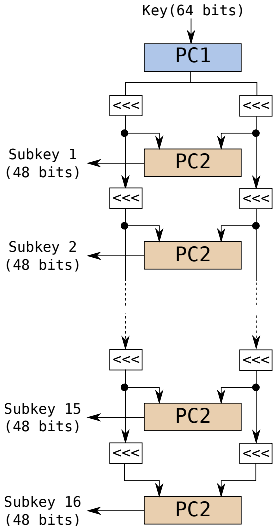
An Example for Block Ciphers: DES

Overall structure:

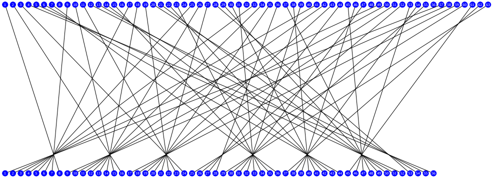
DES uses a 64-bit key with 8 parity bits, hence effectively 56-bits.

- Key schedule: Expand 56-bit key into 16 subkeys.
- Message processing: en-/decode message in 16 rounds.

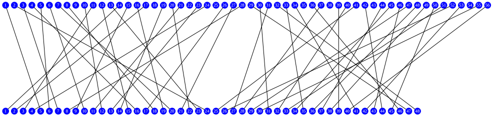
DES: Key Schedule



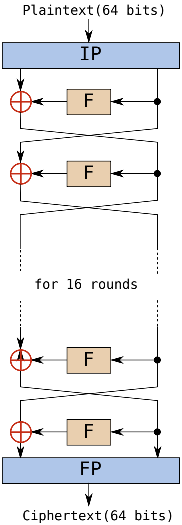
PC1: 64-bit $\longrightarrow 2 \times$ 28-bit



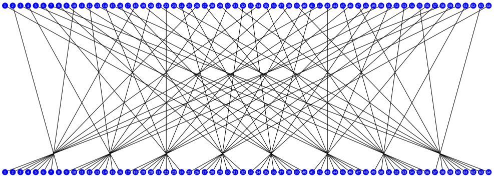
PC2: $2 \times$ 28-bit $\longrightarrow$ 48-bit



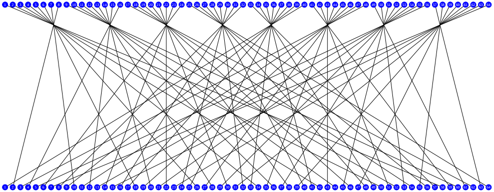
DES: Message En-/decoding



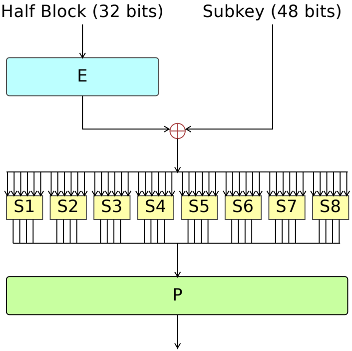
Initial permutation:



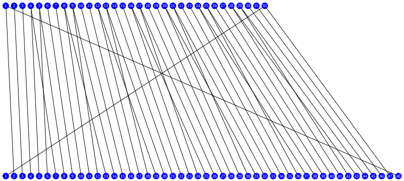
Final permutation:



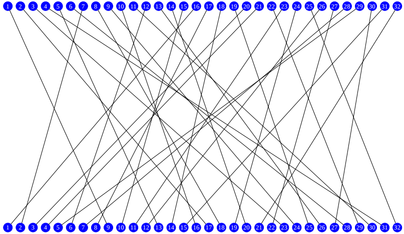
DES: Message En-/decoding



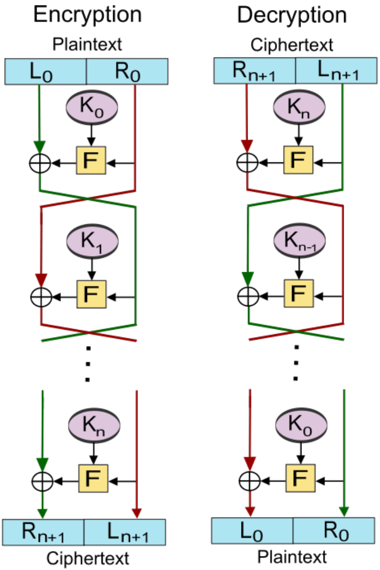
Expansion function:



Permutation:



DES: Message En-/decoding



An Example for Block Ciphers: DES

DES is broken!

DES provides only 56-bits of security and can be easily broken by a brute-force attack!

An Example for Block Ciphers: DES

DES is broken!

DES provides only 56-bits of security and can be easily broken by a brute-force attack!

DES is not fully broken mathematically...

There are attacks with lower complexity than brute force but those require a large amount of known plaintext-ciphertext pairs.

An Example for Block Ciphers: DES

DES is broken!

DES provides only 56-bits of security and can be easily broken by a brute-force attack!

DES is not fully broken mathematically...

There are attacks with lower complexity than brute force but those require a large amount of known plaintext-ciphertext pairs.

Still, DES is broken!

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$

Technische Universiteit Eindhoven University of Technology

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,
- $k_0 = k_1 = k_2$ (fallback to DES for backward compatibility).

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,
- $k_0 = k_1 = k_2$ (fallback to DES for backward compatibility).

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,
- $k_0 = k_1 = k_2$ (fallback to DES for backward compatibility).

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,
- $k_0 = k_1 = k_2$ (fallback to DES for backward compatibility).

An Example for Block Ciphers: Triple DES

Algorithm of Triple DES:

Use three 56-bit keys k_0, k_1, and k_2.

- Encryption: $C = E_{k_2}(D_{k_1}(E_{k_0}(M)))$
- Decryption: $M = D_{k_0}(E_{k_1}(D_{k_2}(C)))$

Keying Options:

- All three keys are independent and different,
- $k_0 = k_2$, and k_1 is different,
- $k_0 = k_1 = k_2$ (fallback to DES for backward compatibility).

Limited Security!

- Option 1 provides about 112-bits of security.
- Option 2 provides about 80-bits of security.
- Option 3 does *not* provide security.

An Example for Block Ciphers: AES

History:

- **September 1997:** NIST issued a public call for a new block cipher: block length of 128 bits; key lengths of 128, 192, and 256 bits.

An Example for Block Ciphers: AES

History:

- **September 1997:** NIST issued a public call for a new block cipher: block length of 128 bits; key lengths of 128, 192, and 256 bits.
- **August 1998 and March 1999:** AES1 and AES2 conferences organized by NIST.

An Example for Block Ciphers: AES

History:

- **September 1997:** NIST issued a public call for a new block cipher: block length of 128 bits; key lengths of 128, 192, and 256 bits.
- **August 1998 and March 1999:** AES1 and AES2 conferences organized by NIST.
- **August 1999:** NIST announces 5 finalists:
  - MARS (IBM),
  - RCG (Rivest, Robshaw, Sidney, Yin),
  - Rijndael (Daemen, Rijmen),
  - Serpent (Anderson, Biham, Knudsen),
  - Twofish (Schneier).

An Example for Block Ciphers: AES

History:

- **September 1997:** NIST issued a public call for a new block cipher: block length of 128 bits; key lengths of 128, 192, and 256 bits.
- **August 1998 and March 1999:** AES1 and AES2 conferences organized by NIST.
- **August 1999:** NIST announces 5 finalists:
  - MARS (IBM),
  - RCG (Rivest, Robshaw, Sidney, Yin),
  - Rijndael (Daemen, Rijmen),
  - Serpent (Anderson, Biham, Knudsen),
  - Twofish (Schneier).
- **April 2000:** AES3 conference.

An Example for Block Ciphers: AES

History:

- **September 1997:** NIST issued a public call for a new block cipher: block length of 128 bits; key lengths of 128, 192, and 256 bits.
- **August 1998 and March 1999:** AES1 and AES2 conferences organized by NIST.
- **August 1999:** NIST announces 5 finalists:
  - MARS (IBM),
  - RCG (Rivest, Robshaw, Sidney, Yin),
  - Rijndael (Daemen, Rijmen),
  - Serpent (Anderson, Biham, Knudsen),
  - Twofish (Schneier).
- **April 2000:** AES3 conference.
- **October 2nd, 2000:** NIST announces Rijndael as winner.

An Example for Block Ciphers: AES

Parameters:

- fixed block size of 128bit,
- variable key size (in bits): AES-128, AES-192, AES-256.

Animation:

http://poincare.matf.bg.ac.rs/~ezivkovm/nastava/rijndael_animacija.swf

Rijndael S-box:

For y in $GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ compute

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

with $z = y^{-1}$.

An Example for Block Ciphers: AES

Rijndael S-box:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

An Example for Block Ciphers: AES

Optimizations for 32-bit Architectures:

- Lookup tables $T_0, \ldots, T_3$ combining all steps.

An Example for Block Ciphers: AES

Optimizations for 32-bit Architectures:

- Lookup tables $T_0, \ldots, T_3$ combining all steps.

Security Concerns:

- Theoretical attacks reduce security of AES-128 to $2^{126.1}$.
- Cache-timing attacks are practical attacks but require precise timing measurements.
  $\rightarrow$ AES implementations must be resistant to timing attacks!

An Example for Block Ciphers: AES

Optimizations for 32-bit Architectures:

- Lookup tables $T_0, \dots, T_3$ combining all steps. ⚠

Security Concerns:

- Theoretical attacks reduce security of AES-128 to $2^{126.1}$.
- Cache-timing attacks are practical attacks but require precise timing measurements.
  $\rightarrow$ AES implementations must be resistant to timing attacks!

An Example for Block Ciphers: AES

Optimizations for 32-bit Architectures:

- Lookup tables $T_0, \ldots, T_3$ combining all steps. ⚠

Security Concerns:

- Theoretical attacks reduce security of AES-128 to $2^{126.1}$.
- Cache-timing attacks are practical attacks but require precise timing measurements.
  $\rightarrow$ AES implementations must be resistant to timing attacks!

High-Speed Implementations:

- NaCl: `http://nacl.cr.yp.to/features.html`
- `http://cryptojedi.org/crypto/index.shtml#aesbs`

Image Credits

The figures are from Wikipedia and have been published with share-alike licenses or have been put into public domain:

- The shematics for the "Modes of Operation" on slides 8 to 12, the DES permutation figures on slides 21 to 24 (right column), and the picture of the DES F-function on slide 23 have been released into public domain.
- The DES flow diagrams on slides 21 and 22 have been published CC BY-SA by by Francisco Menendez (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons:
  https://commons.wikimedia.org/wiki/File:DES-main-network.svg
  https://commons.wikimedia.org/wiki/File:DES_key_schedule.svg
- "Feistel cipher diagram en" (slide 24) by Amirkiderivative, derived from Amirki (talk). Licensed under CC BY-SA 3.0 via Commons -
  https://commons.wikimedia.org/wiki/File:
  Feistel_cipher_diagram_en.svg#/media/File:Feistel_cipher_diagram_en.svg

Technische Universiteit Eindhoven University of Technology

Image Credits (continued)

- The pictures for the "Tux ECB example" on slide 8 are released under Attribution via Commons:
  - "TuX" by Larry Ewing (lewing@isc.tamu.edu) using The Gimp. Licensed under Attribution via Commons: https://commons.wikimedia.org/wiki/File:Tux.jpg#/media/File:Tux.jpg
  - "Tux ecb" by en:User:Lunkwill, dervied from "Tux" by Larry Ewing (see above). Licensed under Attribution via Commons: https://commons.wikimedia.org/wiki/File:Tux_ecb.jpg#/media/File:Tux_ecb.jpg
  - "Tux secure" by en:User:Lunkwill, dervied from "Tux" by Larry Ewing (see above). Licensed under Attribution via Commons: https://commons.wikimedia.org/wiki/File:Tux_secure.jpg#/media/File:Tux_secure.jpg