Block ciphers and AES

Cryptology, 2016 Autumn

Joan Daemen
Institute for Computing and Information Sciences
Radboud University
September 27, 2016

Outline

Introduction

Block cipher model and security definition

Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

Authentication modes of block ciphers

Currently we are here...

Introduction

Block cipher model and security definition

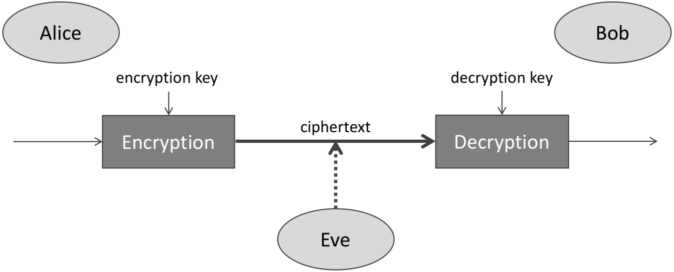
Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

Authentication modes of block ciphers

Encryption



- Alice: sender, enciphers message to cryptogram using key
- Bob: receiver, deciphers cryptogram to message using key
- Eve: eavesdropper, does not have key

The one-time pad

message =	0	1	0	1	1	1	0	1	
keystream =	1	1	1	1	0	0	0	0	$\oplus$
cryptogram =	1	0	1	0	1	1	0	1	

The one-time pad

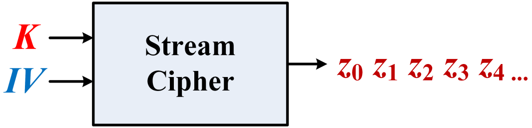
message =	0	1	0	1	1	1	0	1	
keystream =	1	1	1	1	0	0	0	0	$\oplus$
cryptogram =	1	0	1	0	1	1	0	1	

The one-time pad

message =	0	1	0	1	1	1	0	1	
keystream =	1	1	1	1	0	0	0	0	$\oplus$
cryptogram =	1	0	1	0	1	1	0	1	

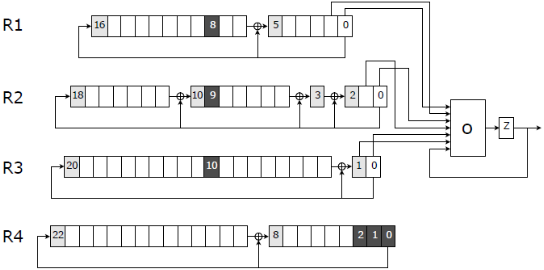
Provably secure if keystream is *fully random*

Stream cipher



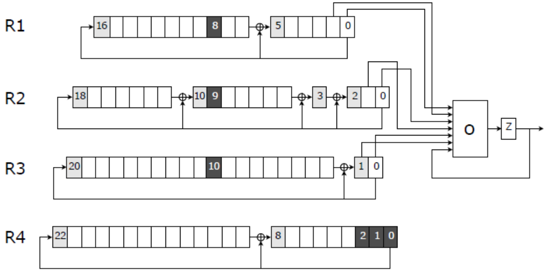
- Generates keystream bits z_t from
  - K: secret, typically 128 or 256 bits
  - IV: initial value, for generating multiple keystreams per key
- z_t can be a bit or a sequences of bits, e.g. a 32-bit word

Example: DECT Stream Cipher



- In use in hundreds of millions of wireless phones
- 4 LFSRs with coprime lengths: large period
- top 3 clocked 2 or 3 times in between time steps t

Example: DECT Stream Cipher



- In use in hundreds of millions of wireless phones
- 4 LFSRs with coprime lengths: large period
- top 3 clocked 2 or 3 times in between time steps t
- practically broken with statistical key recovery attack

Example: RC4 [Ron Rivest] stream cipher

- State is array of 256 bytes
- Simple and elegant update function and output function
- Software-oriented

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```

- Used in TLS and WEP
- Biases in keystream
- Practically broken in several use cases

Currently we are here...

Introduction

Block cipher model and security definition

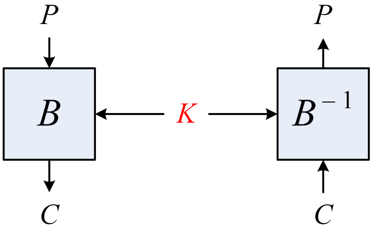
Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

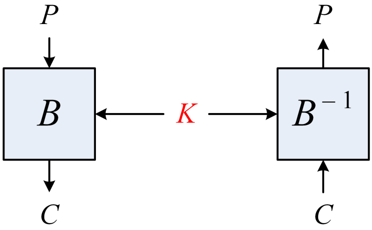
Authentication modes of block ciphers

Block cipher definition



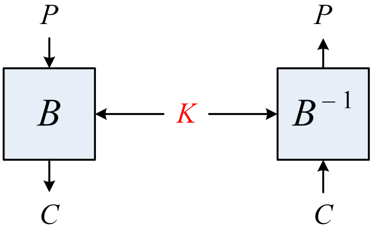
- Permutation B operating on $\mathbb{Z}_2^b$ with b the block length

Block cipher definition



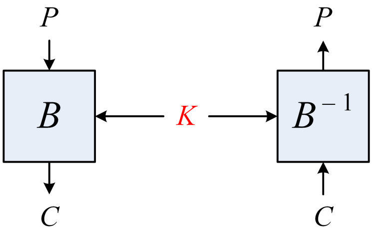
- Permutation B operating on $\mathbb{Z}_2^b$ with b the block length
  - parameterized by a secret key: $B[K]$
  - with an efficient inverse $B^{-1}[K]$

Block cipher definition



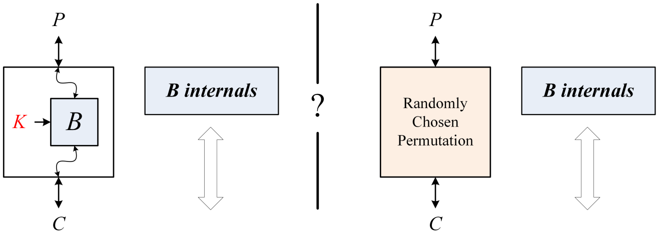
- Permutation B operating on $\mathbb{Z}_2^b$ with b the block length
  - parameterized by a secret key: $B[K]$
  - with an efficient inverse $B^{-1}[K]$
- Computing $C = B[K](P)$ or $P = B^{-1}[K](C)$ should be
  - efficient knowing the secret key K
  - infeasible otherwise

Block cipher definition

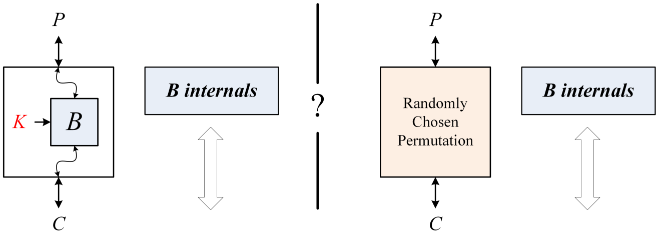


- Permutation B operating on $\mathbb{Z}_2^b$ with b the block length
  - parameterized by a secret key: $B[K]$
  - with an efficient inverse $B^{-1}[K]$
- Computing $C = B[K](P)$ or $P = B^{-1}[K](C)$ should be
  - efficient knowing the secret key K
  - infeasible otherwise
- Dimensions: block length b and key length

Pseudorandom Permutation (PRP) security



Pseudorandom Permutation (PRP) security



- Infeasibility to distinguish $B[K]$ from random permutation
- Distinguishing should have expected effort that is out of reach

iCIS | Digital Security
Radboud University

Currently we are here...

Introduction

Block cipher model and security definition

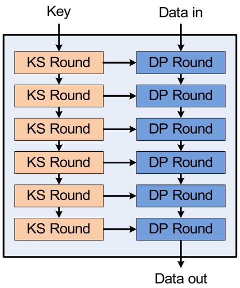
Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

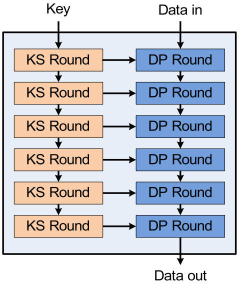
Encryption modes of block ciphers

Authentication modes of block ciphers

Iterative block ciphers

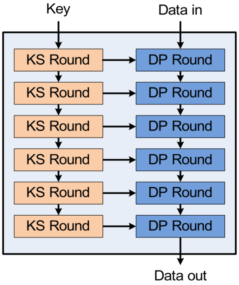


Iterative block ciphers



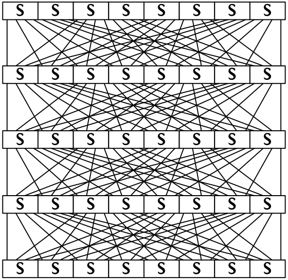
- Data path (right): transforms P to C
  - iteration of a non-linear round function
  - ...that depends on a round key

Iterative block ciphers

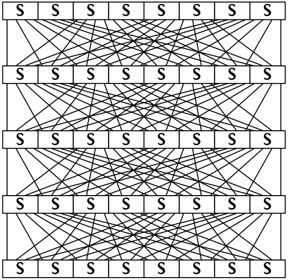


- Data path (right): transforms P to C
  - iteration of a non-linear round function
  - ...that depends on a round key
- Key schedule (left)
  - generates round keys from cipher key K

Substitution-permutation network (SPN)



Substitution-permutation network (SPN)



Round function in data path with two (or three) layers

- Non-linear substitution layer: S-boxes applied in parallel
- permutation layer: moves bits to different S-box positions
- either key-dependent S-boxes or third layer of *key addition*

Data encryption standard (DES)

- Standard by and for US government
- By National Institute for Standardization and Technology (NIST)
- Designed by IBM in collaboration with NSA
- 1977: Federal Information Processing Standard (FIPS) 46

Data encryption standard (DES)

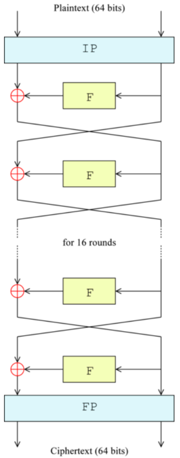
- Standard by and for US government
- By National Institute for Standardization and Technology (NIST)
- Designed by IBM in collaboration with NSA
- 1977: Federal Information Processing Standard (FIPS) 46
  - complete block cipher specification
  - block length: 64 bits, key length: 56 bits
  - no design rationale
  - freely usable

Data encryption standard (DES)

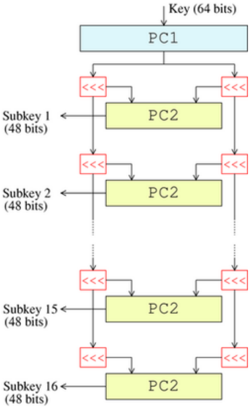
- Standard by and for US government
- By National Institute for Standardization and Technology (NIST)
- Designed by IBM in collaboration with NSA
- 1977: Federal Information Processing Standard (FIPS) 46
  - complete block cipher specification
  - block length: 64 bits, key length: 56 bits
  - no design rationale
  - freely usable
- Massively adopted by banks and industry worldwide
- Dominated symmetric crypto for more than 20 years

Data encryption standard: overview

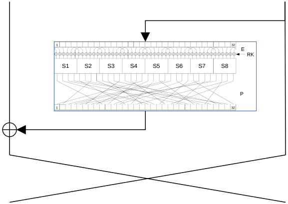
Feistel data path



Linear key schedule

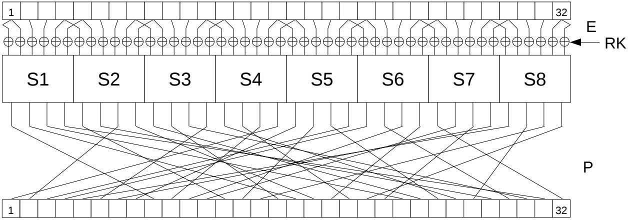


Data encryption standard: F-function



- Variant of SPN with 4 layers:

Data encryption standard: F-function



- Variant of SPN with 4 layers:
  - expansion E: from 32 to 48 bits
  - bitwise round key addition
  - substitution: 8 different 6-to-4 bit non-linear S-boxes
  - permutation P: moving nearby bits to remote positions
- clearly hardware-oriented

Non-ideal DES property: Weak Keys

- What happens if the cipher key is all-zero?
  - all round keys are all-zero
  - all rounds are the same
  - cipher and its inverse are the same
- Same is true for an all-one cipher key
- And two more keys due to symmetry in key schedule
- Weak key K_w:

$$\mathsf{DES}[K_w] \circ \mathsf{DES}[K_w] = \mathrm{I}$$

- Also 6 semi-weak key pairs (K_1, K_2)

$$\mathsf{DES}[K_1] \circ \mathsf{DES}[K_2] = \mathrm{I}$$

- Mostly of academic interest

Non-ideality in DES: Complementation Property

- What happens if we complement the cipher input?
  - flip all bits in key
  - flip all bits in plaintext
- In first round
  - input to F complemented so output of E complemented
  - round key also complemented so input to S-boxes unaffected
  - output of F unaffected
- Output of first round is simply complemented
- Repeat this until you reach the ciphertext
- Complementation property:

$$\mathrm{DES}[K](P) = C \Leftrightarrow= \mathrm{DES}[\overline{K}](\overline{P}) = \overline{C}$$

- Reduces complexity of exhaustive key search from 2^{55} to 2^{54}

iCIS | Digital Security
Radboud University

Non-ideal DES properties: statistical attacks

- Two specific key-recovery attacks:
  - differential cryptanalysis: exploits difference propagation
  - linear cryptanalysis: exploits large P-to-C correlations
- Differential cryptanalysis [Biham and Shamir, 1990]
  - propag. of plaintext difference Δ_p to ciphertext difference Δ_c
  - $\mathrm{DP}(\Delta_p, \Delta_a)$: probability that Δ_p results in Δ_c
  - $\exists \Delta_p, \Delta_c$ with $\mathrm{DP}(\Delta_p, \Delta_a)$ relatively high for all keys
  - requires $|Q_s| \approx 2^{47}$ (1000 TeraByte) chosen plaintexts
- Linear cryptanalysis [Matsui, 1992]
  - correlation between bits in plaintext $u_p^{\mathrm{T}} p$ and ciphertext $u_c^{\mathrm{T}} c$
  - $\mathsf{Corr}(u_p, u_a)$: correlation between $u_p^{\mathrm{T}} p$ and $u_c^{\mathrm{T}} c$
  - $\exists u_p, u_c$ with $\mathsf{Corr}(u_p, u_c)$ relatively high for all keys
  - requires about $|Q_s| \approx 2^{43}$ (64 TeraByte) known plaintexts
- Both *break* DES but still non-trivial to exploit in the field

The real problem of DES: the short key

The real problem of DES: the short key

- Exhaustive key search: about 3.6×10^{14} trials

iCIS | Digital Security
Radboud University

The real problem of DES: the short key

- Exhaustive key search: about 3.6×10^{14} trials
- More than 15 years ago: "software" cracking
  - about 10.000 workstations
  - 500.000 trials per second per workstation
  - expected time: 7.200.000 seconds: 2,5 months
  - applied in cracking RSA labs DES challenge, June 97

The real problem of DES: the short key

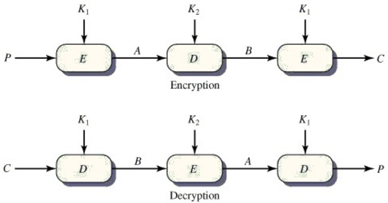
- Exhaustive key search: about 3.6×10^{14} trials
- More than 15 years ago: "software" cracking
  - about 10.000 workstations
  - 500.000 trials per second per workstation
  - expected time: 7.200.000 seconds: 2,5 months
  - applied in cracking RSA labs DES challenge, June 97
- Cracking using dedicated hardware
  - COPACOBANA RIVYERA (2008)
  - costs about 10.000$
  - board with 128 Spartan-3 5000 FPGAs.
  - finds a DES key in less than a day

The real problem of DES: the short key

- Exhaustive key search: about 3.6×10^{14} trials
- More than 15 years ago: "software" cracking
  - about 10.000 workstations
  - 500.000 trials per second per workstation
  - expected time: 7.200.000 seconds: 2,5 months
  - applied in cracking RSA labs DES challenge, June 97
- Cracking using dedicated hardware
  - COPACOBANA RIVYERA (2008)
  - costs about 10.000$
  - board with 128 Spartan-3 5000 FPGAs.
  - finds a DES key in less than a day
- Short DES key is real-world concern!

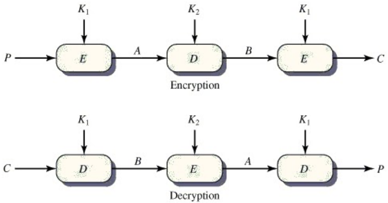
iCIS | Digital Security
Radboud University

The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

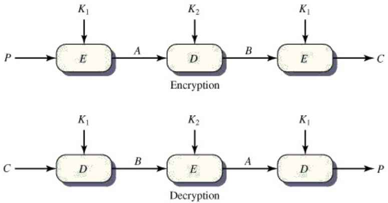
The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

- Double DES allows meet-in-the-middle attacks

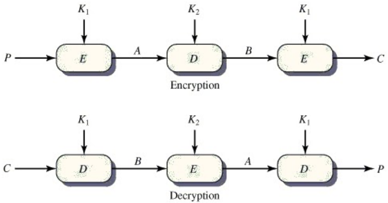
The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

- Double DES allows meet-in-the-middle attacks
- Three variants of Triple-DES

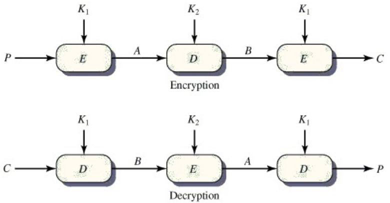
The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

- Double DES allows meet-in-the-middle attacks
- Three variants of Triple-DES
  - 3-key: 168-bit key, only option allowed by NIST

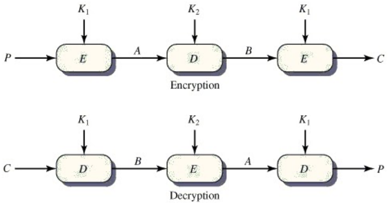
The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

- Double DES allows meet-in-the-middle attacks
- Three variants of Triple-DES
  - 3-key: 168-bit key, only option allowed by NIST
  - 2-key: 112-bit key by taking $K_3 = K_1$
    - still massively deployed by banks worldwide

The solution: Triple DES (FIPS 46-2 and 46-3)



(b) Triple encryption

- Double DES allows meet-in-the-middle attacks
- Three variants of Triple-DES
  - 3-key: 168-bit key, only option allowed by NIST
  - 2-key: 112-bit key by taking $K_3 = K_1$
    - still massively deployed by banks worldwide
  - 1-key: 56-bit key by taking $K_3 = K_2 = K_1$
    - falls back to single DES thanks to inverse in middle

Currently we are here...

Introduction

Block cipher model and security definition

Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

Authentication modes of block ciphers

AES: the result of a competition

- January 1997: NIST announces the AES initiative
  - replacement of DES
  - open call for block cipher proposals
  - ...and for analysis, comparisons, etc.
- September 1997: official request for proposals
  - *faster than Triple-DES*
  - 128-bit blocks, 128-, 192- and 256-bit keys
  - specs, reference and optimized code, test vectors
  - design rationale and preliminary analysis
  - patent waiver
- Vincent Rijmen and I decided to submit a variant of Square
  - Most important change: multiple key and block lengths
  - We call it Rijndael

The AES competition

- First round: August 1998 to August 1999
  - 15 candidates at 1st AES conference in Ventura, California
  - analysis presented at 2nd AES conf. in Rome, March 1999
  - NIST narrowed down to 5 finalists using this analysis
- Second round: August 1999 to summer 2000
  - analysis presented at 3rd AES conf. in New York, April 2000
  - NIST selected winner using this analysis
- Criteria
  - security margin
  - efficiency in software and hardware
  - key agility
  - simplicity
- NIST motivated their choice in two reports

Rijndael design approach: the wide trail strategy

- Round function with four layers, each with separate goal:
  - nonlinear layer: S-boxes with high non-linearity
  - dispersion layer: like P in DES F-function
  - mixing layer (absent in DES): linear local mixing
  - round key addition

Rijndael design approach: the wide trail strategy

- Round function with four layers, each with separate goal:
  - nonlinear layer: S-boxes with high non-linearity
  - dispersion layer: like P in DES F-function
  - mixing layer (absent in DES): linear local mixing
  - round key addition
- Mixing layer goals:
  - each output bit depends on multiple input bits
  - each small input difference propagates to multiple output bits

Rijndael design approach: the wide trail strategy

- Round function with four layers, each with separate goal:
  - nonlinear layer: S-boxes with high non-linearity
  - dispersion layer: like P in DES F-function
  - mixing layer (absent in DES): linear local mixing
  - round key addition
- Mixing layer goals:
  - each output bit depends on multiple input bits
  - each small input difference propagates to multiple output bits
- Quality of mixing layer quantitied its branch number $\mathcal{B}$
  - allows proving bounds related to resistance against LC/DC
  - in combination with S-box layer and transposition layer
  - link with theory of error-correcting codes
  - optimum mix layer = maximum-distance-separable (MDS) code

Rijndael

- Block cipher with block and key lengths $\in \{128, 160, 192, 224, 256\}$
  - set of 25 block ciphers
  - AES limits block length to 128 and key length to multiples of 64

Rijndael

- Block cipher with block and key lengths $\in \{128, 160, 192, 224, 256\}$
  - set of 25 block ciphers
  - AES limits block length to 128 and key length to multiples of 64
- Round function with four steps
  - all rounds are identical
  - ...except for the round keys
  - ...and omission of mixing layer in last round
  - parallel and symmetric

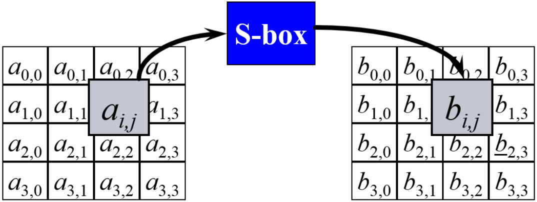
Rijndael

- Block cipher with block and key lengths $\in \{128, 160, 192, 224, 256\}$
  - set of 25 block ciphers
  - AES limits block length to 128 and key length to multiples of 64
- Round function with four steps
  - all rounds are identical
  - ...except for the round keys
  - ...and omission of mixing layer in last round
  - parallel and symmetric
- Key schedule
  - Expansion of cipher key to round key sequence
  - Recursive procedure that can be done in-place

Rijndael

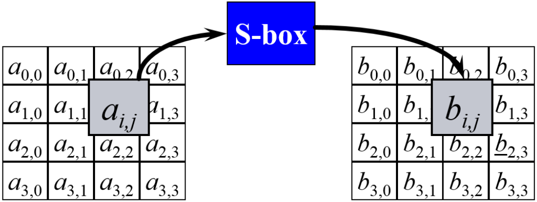
- Block cipher with block and key lengths $\in \{128, 160, 192, 224, 256\}$
  - set of 25 block ciphers
  - AES limits block length to 128 and key length to multiples of 64
- Round function with four steps
  - all rounds are identical
  - ...except for the round keys
  - ...and omission of mixing layer in last round
  - parallel and symmetric
- Key schedule
  - Expansion of cipher key to round key sequence
  - Recursive procedure that can be done in-place
- Manipulates bytes with simple operations in $GF(2^8)$

The non-linear layer: SubBytes



Single S-box with two layers:

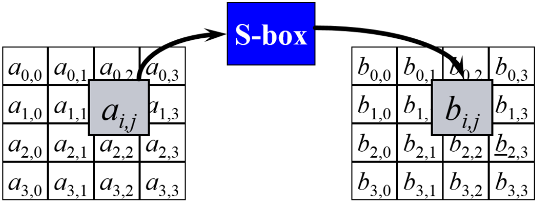
The non-linear layer: SubBytes



Single S-box with two layers:

- $y = x^{254}$ in GF(2^8)
  - $x^{\#x} = 1$ (Lagrange) so $y = x^{-1}$ for $x \neq 0$
  - optimal non-linearity [Nyberg, Eurocrypt 1993]

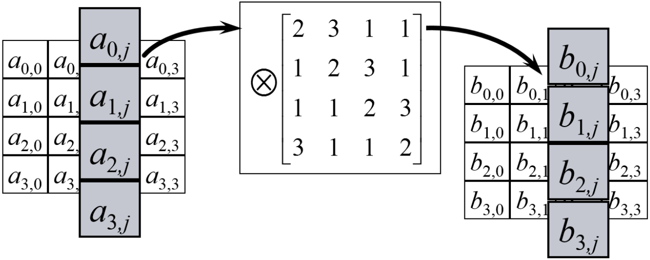
The non-linear layer: SubBytes



Single S-box with two layers:

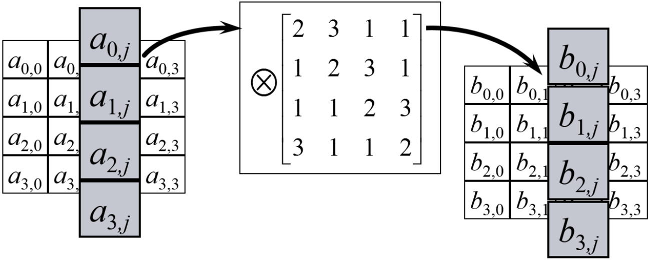
- $y = x^{254}$ in GF(2^8)
  - $x^{\#x} = 1$ (Lagrange) so $y = x^{-1}$ for $x \neq 0$
  - optimal non-linearity [Nyberg, Eurocrypt 1993]
- Affine mapping: multiplication by 8×8 matrix in GF(2)
  - to have algebraic complexity, without it: $xy = 1$ for $x \neq 0$

The mixing layer: MixColumns



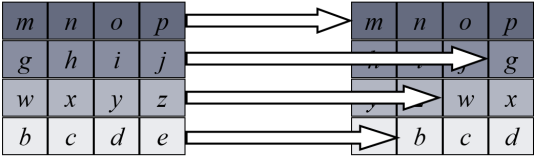
- Same mapping applied to all 4 columns

The mixing layer: MixColumns

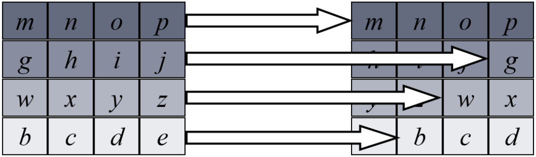


- Same mapping applied to all 4 columns
- Multiplication by a 4×4 circulant matrix in GF(2^8)
  - Elements: 1, 1, x and $x + 1$
  - *circulant MDS ($\mathcal{B} = 5$) matrix with the simplest elements*
  - Inverse has more complex elements

The dispersion layer: ShiftRows



The dispersion layer: ShiftRows



- Each row is shifted by a different amount
- Different shift offsets for higher block lengths
- Together with MixColumns and SubBytes:
- Together with MixColumns and SubBytes:
  - full diffusion in two rounds
  - $\mathcal{B}^2 = 25$ *active S-boxes in 4 rounds*

Round key addition: AddRoundKey

$$\begin{array}{|c|c|c|c|}\hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} + \begin{array}{|c|c|c|c|}\hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Key schedule: 192-bit key, 128-bit block example

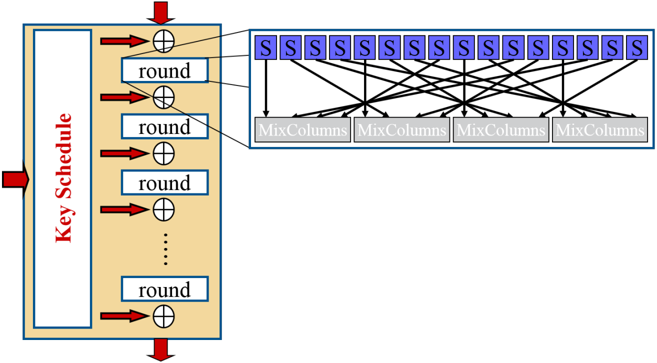
k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}	$\cdots$

Round key 0	Round key 1	Round key 2	$\cdots$

$$
\begin{aligned}
k_{6n} &= k_{6n-6} \oplus f(k_{6n-1}) \\
k_i &= k_{i-6} \oplus k_{i-1}, \quad i \neq 6n
\end{aligned}
$$

f: AES S-box in parallel to 4 bytes followed by cyclic shift over 1 byte

Rijndael: summary



- $\#$ rounds: $6 + \max(\ell_k, \ell_b)$ with ℓ_k key and ℓ_b block length in 32-bit words
- last round has no MixColumns to make inverse similar to cipher

Rijndael symmetry

- Highly symmetric round function (as opposed DES)
  - SubBytes: 1 S-box instead of different ones
  - MixColumns: 1 MDS matrix with circulant symmetry
  - ShiftRows: bytes relative movement independent of position
  - round function minus key addition is shift-invariant

Rijndael symmetry

- Highly symmetric round function (as opposed DES)
  - SubBytes: 1 S-box instead of different ones
  - MixColumns: 1 MDS matrix with circulant symmetry
  - ShiftRows: bytes relative movement independent of position
  - round function minus key addition is shift-invariant
- Very high symmetry in nonlinear part of S-box: $y = x^{-1}$
  - representation of elements of GF(2^8): choice of basis
  - elements as degree$<$ 2 polynomials with coeff. in GF(2^4)
  - can be done recursively
  - called tower fields

Rijndael symmetry

- Highly symmetric round function (as opposed DES)
  - SubBytes: 1 S-box instead of different ones
  - MixColumns: 1 MDS matrix with circulant symmetry
  - ShiftRows: bytes relative movement independent of position
  - round function minus key addition is shift-invariant
- Very high symmetry in nonlinear part of S-box: $y = x^{-1}$
  - representation of elements of $GF(2^8)$: choice of basis
  - elements as degree$<$ 2 polynomials with coeff. in $GF(2^4)$
  - can be done recursively
  - called tower fields
- Asymmetry:
  - inverse is different and slightly more expensive
  - key schedule has some symmetry, but much less

Rijndael implementation aspects

- Implementations can exploit symmetry

iCIS | Digital Security
Radboud University

Rijndael implementation aspects

- Implementations can exploit symmetry
- Software with table-lookups:
  - 4 Kbytes of table
  - 16 table-lookup + 16 XORs per round

Rijndael implementation aspects

- Implementations can exploit symmetry
- Software with table-lookups:
  - 4 Kbytes of table
  - 16 table-lookup + 16 XORs per round
- Software in bitslice:
  - rearrangement of the bits
  - only bitwise Boolean instructions and shifts

iCIS | Digital Security
Radboud University

Rijndael implementation aspects

- Implementations can exploit symmetry
- Software with table-lookups:
  - 4 Kbytes of table
  - 16 table-lookup + 16 XORs per round
- Software in bitslice:
  - rearrangement of the bits
  - only bitwise Boolean instructions and shifts
- Hardware:
  - very suitable thanks to arithmetic in GF(2^n) instead of $(\mathbb{Z}_{2^n}, +)$
  - fully parallel: combinatorial logic with full round
  - serial: logic for 1 S-box and 1 MixColumns matrix column
  - S-box area/circuit depth trade-off by using tower fields

Rijndael security status

- Cryptanalysis (in public domain)
  - all attacks, also on reduced-round, have huge data complexity
  - there is an (academic) attack against full-round AES:
    - biclique attacks [Bogdanov, Khovratovich, Rechberger, 2011]
    - $|Q_c| \approx 2^{126}$: factor 2 gain compared to exhaustive key search
    - gain evaporates when looking at complete picture
  - solid security status thanks to public scrutiny
- Implementation attacks: exploiting implementation weaknesses
  - timing attacks: cache misses in table-lookups
  - power analysis: exploiting dependence of current on data
  - electromagnetic analysis: same for EM emanations
  - fault attacks: exploiting forced faults
- Implementation attacks are the ones that matter in practice!

Currently we are here...

Introduction

Block cipher model and security definition

Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

Authentication modes of block ciphers

Block cipher modes for encryption

- DES can encipher 8-byte messages, AES of 16-byte messages
  - what about longer and shorter messages?
  - two approaches: block encryption and stream encryption

Block cipher modes for encryption

- DES can encipher 8-byte messages, AES of 16-byte messages
  - what about longer and shorter messages?
  - two approaches: block encryption and stream encryption
- Block encryption modes
  - split the message in blocks
  - after padding last *incomplete* block if needed
  - apply permutation $B[K]$ (keyed block cipher) to blocks in some way

Block cipher modes for encryption

- DES can encipher 8-byte messages, AES of 16-byte messages
  - what about longer and shorter messages?
  - two approaches: block encryption and stream encryption
- Block encryption modes
  - split the message in blocks
  - after padding last *incomplete* block if needed
  - apply permutation $B[K]$ (keyed block cipher) to blocks in some way
- Stream encryption modes
  - build a stream cipher with a block cipher as building block

Block encryption modes

- Ideal: wide block encryption
  - each cryptogram bit depends on each message bit and vice versa
  - hard to build using a fixed-length block cipher
  - not online: cannot encipher long messages on the fly

Block encryption modes

- Ideal: wide block encryption
  - each cryptogram bit depends on each message bit and vice versa
  - hard to build using a fixed-length block cipher
  - not online: cannot encipher long messages on the fly
- Tolerate some level of degeneracy

Block encryption modes

- Ideal: wide block encryption
  - each cryptogram bit depends on each message bit and vice versa
  - hard to build using a fixed-length block cipher
  - not online: cannot encipher long messages on the fly
- Tolerate some level of degeneracy
- Electronic Code Book (ECB) mode
  - *we consider only 16-byte messages*
  - longer messages are split in 16-byte blocks
  - shorter messages padded to 16 bytes
  - same for *last incomplete block*

Block encryption modes

- Ideal: wide block encryption
  - each cryptogram bit depends on each message bit and vice versa
  - hard to build using a fixed-length block cipher
  - not online: cannot encipher long messages on the fly
- Tolerate some level of degeneracy
- Electronic Code Book (ECB) mode
  - *we consider only 16-byte messages*
  - longer messages are split in 16-byte blocks
  - shorter messages padded to 16 bytes
  - same for *last incomplete block*
- Cipher Block Chaining (CBC) mode
  - *ECB randomized with what's available*
  - requires also split in 16-byte blocks and padding

Block encryption modes

- Ideal: wide block encryption
  - each cryptogram bit depends on each message bit and vice versa
  - hard to build using a fixed-length block cipher
  - not online: cannot encipher long messages on the fly
- Tolerate some level of degeneracy
- Electronic Code Book (ECB) mode
  - *we consider only 16-byte messages*
  - longer messages are split in 16-byte blocks
  - shorter messages padded to 16 bytes
  - same for *last incomplete block*
- Cipher Block Chaining (CBC) mode
  - *ECB randomized with what's available*
  - requires also split in 16-byte blocks and padding
- Due to padding, cryptogram is longer than message

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding
  - send along message or padding length with cryptogram
  - impose padding is injective (or reversible)

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding
  - send along message or padding length with cryptogram
  - impose padding is injective (or reversible)
- simplest reversible padding: a single 1 and then zeroes
  - extends message in all cases
  - turns 16-byte message into 32-byte string

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding
  - send along message or padding length with cryptogram
  - impose padding is injective (or reversible)
- simplest reversible padding: a single 1 and then zeroes
  - extends message in all cases
  - turns 16-byte message into 32-byte string
- padding with exotic requirements
  - random-length padding: to hide message length
  - random padding: to add entropy

Intermezzo: padding

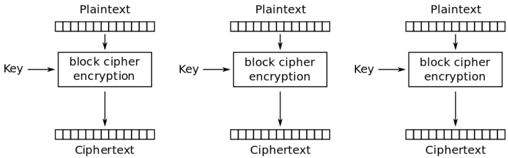
- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding
  - send along message or padding length with cryptogram
  - impose padding is injective (or reversible)
- simplest reversible padding: a single 1 and then zeroes
  - extends message in all cases
  - turns 16-byte message into 32-byte string
- padding with exotic requirements
  - random-length padding: to hide message length
  - random padding: to add entropy
- Badly designed padding is often source of security problems

Intermezzo: padding

- Simplest padding: append zeroes
  - up to length multiple of block length (e.g. 16 bytes)
  - shortest possible padding
  - as such not usable for our purposes
- Decryption of cryptogram gives *padded* message
- Recovering message requires removing padding
  - send along message or padding length with cryptogram
  - impose padding is injective (or reversible)
- simplest reversible padding: a single 1 and then zeroes
  - extends message in all cases
  - turns 16-byte message into 32-byte string
- padding with exotic requirements
  - random-length padding: to hide message length
  - random padding: to add entropy
- Badly designed padding is often source of security problems

iCIS | Digital Security
Radboud University

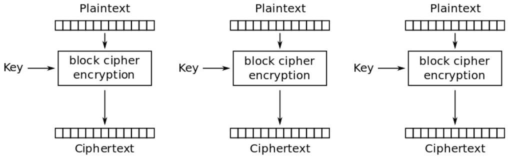
Electronic CodeBook Mode (ECB)



Electronic Codebook (ECB) mode encryption

- Advantages
  - simple
  - parallelizable

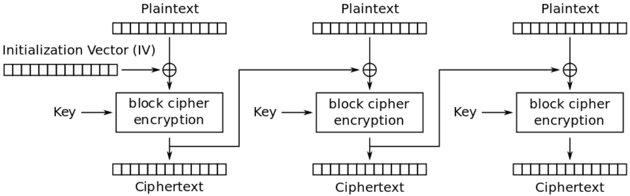
Electronic CodeBook Mode (ECB)



Electronic Codebook (ECB) mode encryption

- Advantages
  - simple
  - parallelizable
- Limitation: equal plaintext blocks $\rightarrow$ equal ciphertext blocks:
  - likely to happen in low-entropy messages
  - problem in padded last block

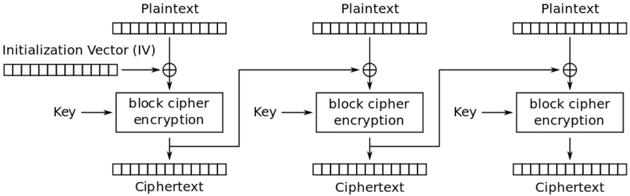
Cipher Block Chaining mode (CBC)



Cipher Block Chaining (CBC) mode encryption

- *ECB with plaintext block randomized by previous ciphertext block*

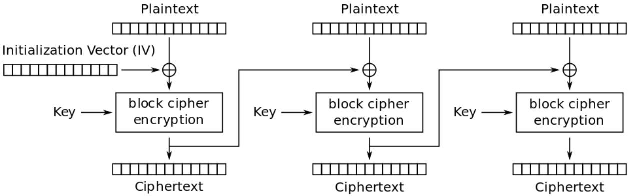
Cipher Block Chaining mode (CBC)



Cipher Block Chaining (CBC) mode encryption

- *ECB with plaintext block randomized by previous ciphertext block*
- First plaintext block randomized with Initial Value (IV)

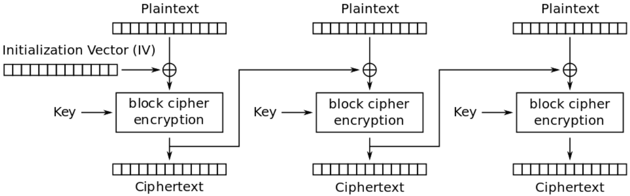
Cipher Block Chaining mode (CBC)



Cipher Block Chaining (CBC) mode encryption

- *ECB with plaintext block randomized by previous ciphertext block*
- First plaintext block randomized with Initial Value (IV)
- Solves leakage in ECB (partially):
  - equal plaintext blocks do not lead to equal ciphertext blocks

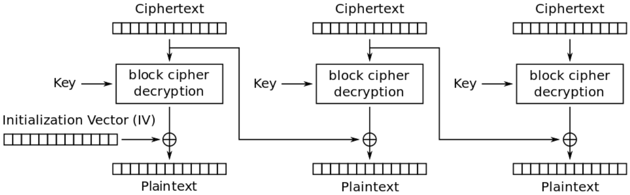
Cipher Block Chaining mode (CBC)



Cipher Block Chaining (CBC) mode encryption

- *ECB with plaintext block randomized by previous ciphertext block*
- First plaintext block randomized with Initial Value (*IV*)
- Solves leakage in ECB (partially):
  - equal plaintext blocks do not lead to equal ciphertext blocks
  - requires randomly generating and transferring *IV*

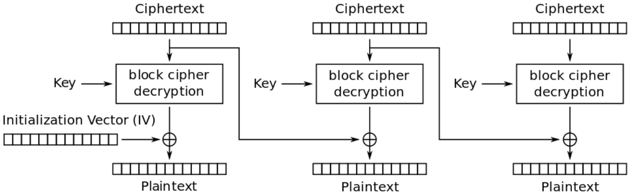
Cipher Block Chaining mode (cont'd)



Cipher Block Chaining (CBC) mode decryption

- Replacing IV randomness by N nonce requirement: $IV = B[K](N)$

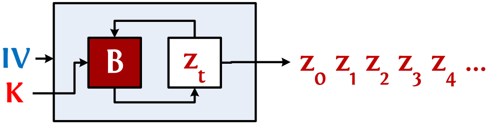
Cipher Block Chaining mode (cont'd)



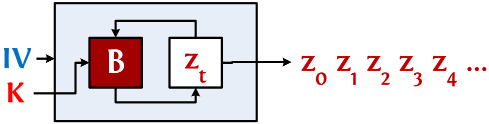
Cipher Block Chaining (CBC) mode decryption

- Replacing IV randomness by N nonce requirement: $IV = B[K](N)$
- Properties of CBC
  - encryption strictly serial, decryption can be parallel
  - IV must be managed and transferred
  - security less than what one would think

Stream encryption: Output FeedBack mode (OFB)

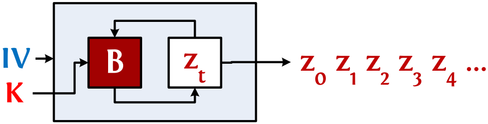


Stream encryption: Output FeedBack mode (OFB)



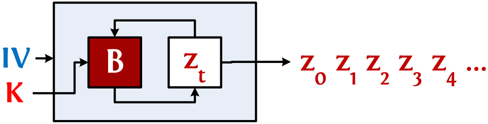
- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and output z_t
  - initialization: $z_{-1} = IV$
  - state update: $z_t = B[K](z_{t-1})$
  - output: just take z_t

Stream encryption: Output FeedBack mode (OFB)



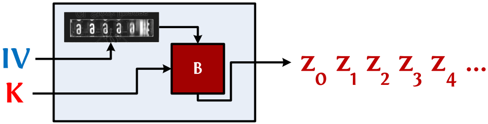
- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and output z_t
  - initialization: $z_{-1} = IV$
  - state update: $z_t = B[K](z_{t-1})$
  - output: just take z_t
- Properties

Stream encryption: Output FeedBack mode (OFB)

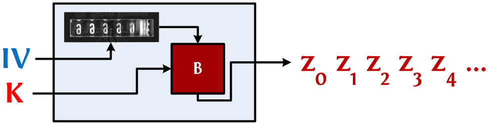


- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and output z_t
  - initialization: $z_{-1} = IV$
  - state update: $z_t = B[K](z_{t-1})$
  - output: just take z_t
- Properties
  - strictly serial
  - cycle lengths not known in advance
  - no need for B^{-1} (valid for all stream encryption)

Stream encryption: Counter mode

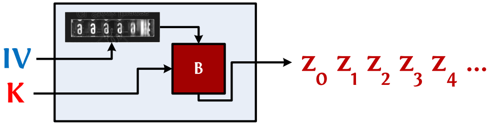


Stream encryption: Counter mode



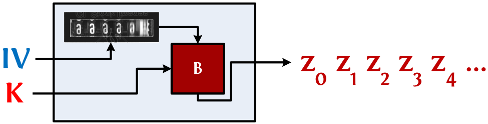
- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and counter c
  - initialization: $c^0 = IV$
  - state update: $c^t = c^{t-1} + 1$, with c interpreted as an integer
  - output: $z_t = B[K](c^t)$

Stream encryption: Counter mode



- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and counter c
  - initialization: $c^0 = IV$
  - state update: $c^t = c^{t-1} + 1$, with c interpreted as an integer
  - output: $z_t = B[K](c^t)$
- Properties

Stream encryption: Counter mode



- Stream cipher with:
  - State a^t consisting of two parts: fixed key K and counter c
  - initialization: $c^0 = IV$
  - state update: $c^t = c^{t-1} + 1$, with c interpreted as an integer
  - output: $z_t = B[K](c^t)$
- Properties
  - fully parallelizable
  - cycle length 2^b with b the block length

Encryption modes: overview

	ECB	CBC	OFB	Counter
parallel encryption	y	n	n	y
parallel decryption	y	y	n	y
random access	y	y	n	y
requires B^{-1}	y	y	n	n
requires padding	y	y	n	n
full collapse if nonce violation	n	n	y	y
error propagation $C \to P$	y	y	n	n

Legend:

- random access: fast decryption of bits anywhere in the message
- error propagation: single-bit error in C expands to b bits in P

Currently we are here...

Introduction

Block cipher model and security definition

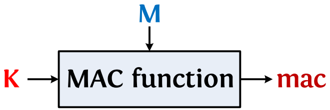
Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

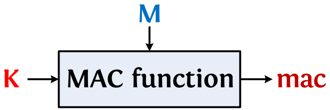
Encryption modes of block ciphers

Authentication modes of block ciphers

Message authentication code (MAC) functions

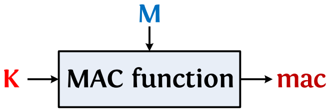


Message authentication code (MAC) functions



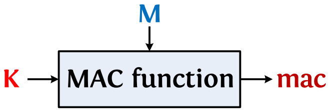
- MAC: cryptographic checksum
  - input: key K and arbitrary-length message M
  - output: ℓ-bit MAC or tag T with ℓ some length

Message authentication code (MAC) functions



- MAC: cryptographic checksum
  - input: key K and arbitrary-length message M
  - output: ℓ-bit MAC or tag T with ℓ some length
- Applications:
  - message authentication: append MAC to message
  - entity authentication: compute MAC over challenge

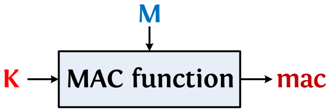
Message authentication code (MAC) functions



- MAC: cryptographic checksum
  - input: key K and arbitrary-length message M
  - output: ℓ-bit MAC or tag T with ℓ some length
- Applications:
  - message authentication: append MAC to message
  - entity authentication: compute MAC over challenge
- Ideal behaviour: pseudorandom function (PRF)
  - returns fully uncorrelated responses for different inuts

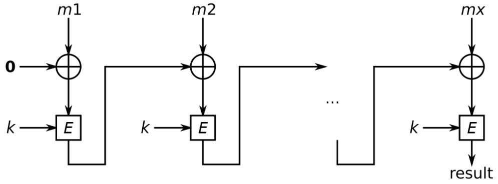
iCIS | Digital Security
Radboud University

Message authentication code (MAC) functions

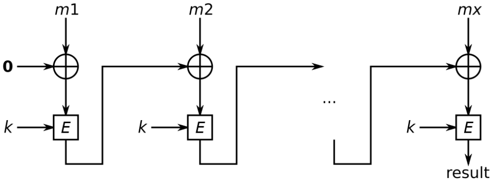


- MAC: cryptographic checksum
  - input: key K and arbitrary-length message M
  - output: ℓ-bit MAC or tag T with ℓ some length
- Applications:
  - message authentication: append MAC to message
  - entity authentication: compute MAC over challenge
- Ideal behaviour: pseudorandom function (PRF)
  - returns fully uncorrelated responses for different inuts
- If ideal, Pr(success) of forging a pair $M, T = \mathrm{MAC}(K, M)$ is $2^{-\ell}$

Cipher Block Chaining MAC mode (CBC-MAC)

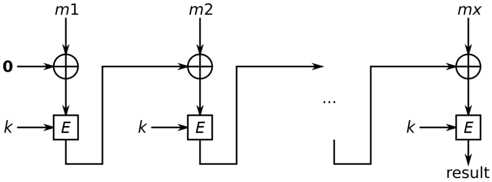


Cipher Block Chaining MAC mode (CBC-MAC)



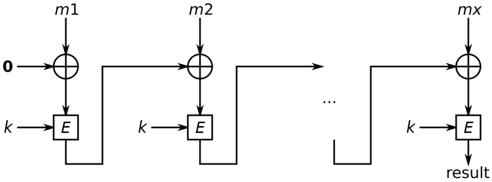
- Observation: in CBC encryption Ci depends on $m1$ to mi

Cipher Block Chaining MAC mode (CBC-MAC)



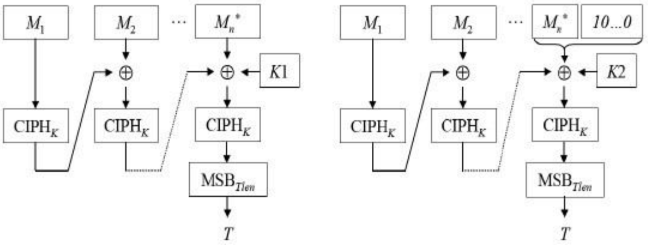
- Observation: in CBC encryption Ci depends on $m1$ to mi
- Idea:
  - Apply CBC encryption to (padded) message
  - take T equal to last ciphertext block
  - throw away other blocks (essential for security)

Cipher Block Chaining MAC mode (CBC-MAC)



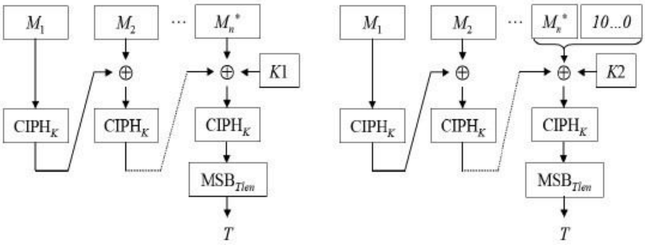
- Observation: in CBC encryption Ci depends on $m1$ to mi
- Idea:
  - Apply CBC encryption to (padded) message
  - take T equal to last ciphertext block
  - throw away other blocks (essential for security)
- Broken for arbitrary-length messages
  - length-extension weakness

A fix of CBC-MAC: C-MAC



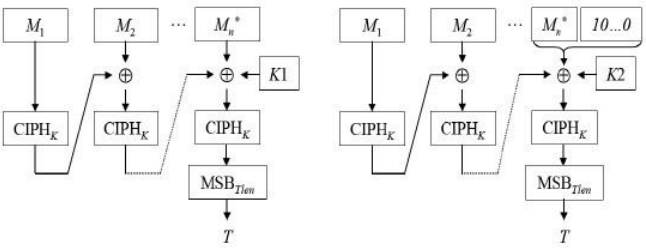
- NIST standard: Special Publication 800-38B

A fix of CBC-MAC: C-MAC



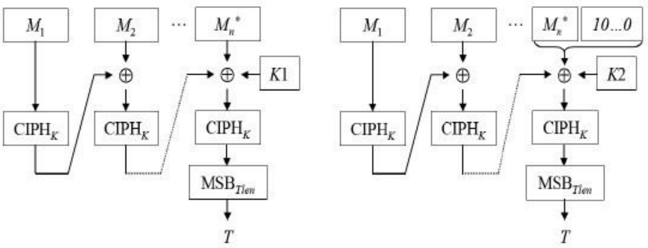
- NIST standard: Special Publication 800-38B
- avoid length-extension by *doing something different at the end*

A fix of CBC-MAC: C-MAC



- NIST standard: Special Publication 800-38B
- avoid length-extension by *doing something different at the end*
- addition of a constant before last application of $B[K]$

A fix of CBC-MAC: C-MAC



- NIST standard: Special Publication 800-38B
- avoid length-extension by *doing something different at the end*
- addition of a constant before last application of $B[K]$
- Secure for arbitrary-length messages

Currently we are here...

Introduction

Block cipher model and security definition

Data Encryption Standard (DES)

Advanced Encryption Standard (AES)

Encryption modes of block ciphers

Authentication modes of block ciphers

Summary

- Block ciphers are keyed b-bit permutations
  - a different permutation $B[K]$ per key K (and tweak w)
  - with an efficient inverse $B[K]^{-1}$
  - exhaustive keysearch should be best attack (complexity $2^{|K|-1}$)
- DES and AES are the most widespread block ciphers
  - constructed by iterating a simple round function
  - round has steps for non-linearity, mixing and transposition
- Block ciphers are versatile:
  - block encryption modes: e.g., ECB and CBC
  - stream encryption modes: e.g., OFB, counter and CFB
  - MAC computation modes: e.g., CBC-MAC and C-MAC
- Inverse permutation only used in block encryption modes