Cryptography Lecture 5

More block cipher algorithms, modes of operation

LINKÖPING UNIVERSITY

Key length

Table 7.4: Security levels (symmetric equivalent)

Security (bits)	Protection	Comment
32	Real-time, individuals	Only auth. tag size
64	Very short-term, small org	Not for confidentiality in new systems
72	Short-term, medium org Medium-term, small org	
80	Very short-term, agencies Long-term, small org	Smallest general-purpose $<$ 4 years protection (E.g., use of 2-key 3DES, $< 2^{40}$ plaintext/ciphertexts)
96	Legacy standard level	2-key 3DES restricted to 10^6 plaintext/ciphertexts, $\approx$ 10 years protection
112	Medium-term protection	$\approx$ 20 years protection (E.g., 3-key 3DES)
128	Long-term protection	Good, generic application-indep. Recommendation, $\approx$ 30 years
256	”Foreseeable future”	Good protection against quantum computers unless Shor’s algorithm applies.

From “ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)”

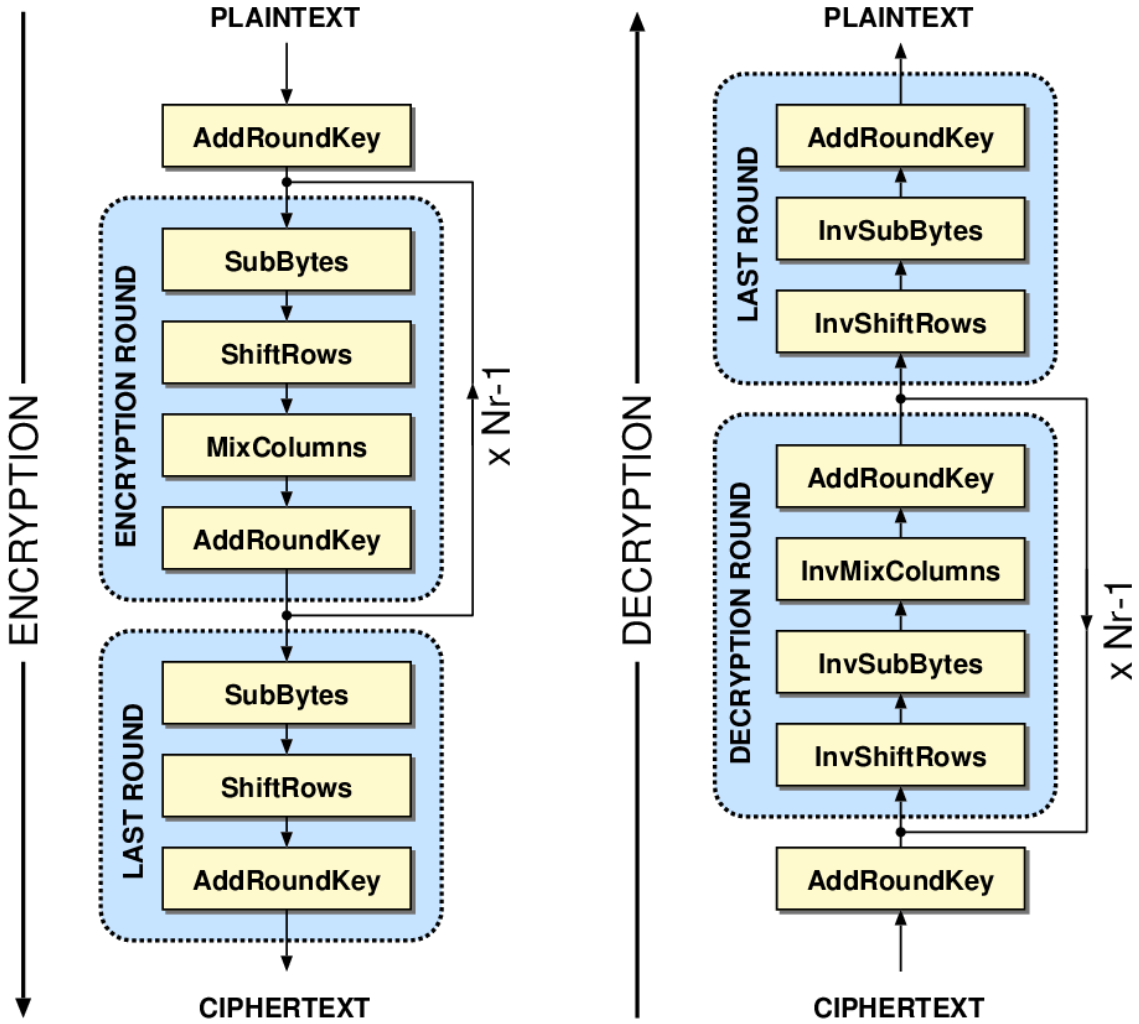
AES competition

- NIST put out a call for new algorithms in 1997, this was the start of the AES competition
- Requirements were: 128 bit blocks, 128, 192 and 256 bit keys, and that it should work well on several kinds of hardware and in software
- Five finalists
  - Rijndael (J. Daemen and V. Rijmen)
  - Serpent (R. Anderson, E. Biham, and L. Knudsen)
  - Twofish (B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson)
  - RC6 (RSA Labs)
  - MARS (IBM)

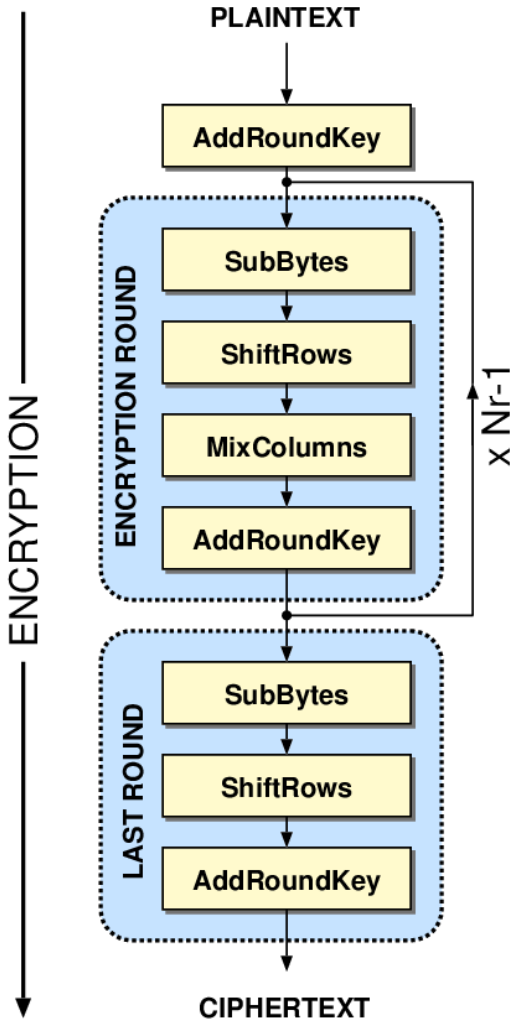
AES competition

- NIST put out a call for new algorithms in 1997, this was the start of the AES competition
- Requirements were: 128 bit blocks, 128, 192 and 256 bit keys, and that it should work well on several kinds of hardware and in software
- Five finalists
  - Rijndael (J. Daemen and V. Rijmen) ← AES (2000)
  - Serpent (R. Anderson, E. Biham, and L. Knudsen)
  - Twofish (B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson)
  - RC6 (RSA Labs)
  - MARS (IBM)

AES structure: 10, 12, or 14 rounds



AES structure: 10, 12, or 14 rounds



- Not a Feistel system
- Addition modulo 2 of the round key
- 8-bit “S-boxes”, this time chosen algebraically as x^{-1} in GF(2^8)
- Shift 8-bit parts between 32-bit sub-blocks, then mix bits in each 32-bit block
- Uses “whitening”
- The key schedule uses the S-box

What does it mean, x^{-1} in GF(2^8)?

A number $y = 1/x$ when we only use integers?

In other words, a number y such that $y * x = 1$?

Let us first look at integers mod 4:
the integers used are 0,1,2,3, and

1+1=2,	1+3=0,	2+3=1,	3+3=2	(mod 4)
1*1=1,	1*3=3,	2*3=2,	3*3=1	(mod 4)

What does it mean, x^{-1} in GF(2^8)?

A number $y = 1/x$ when we only use integers?

In other words, a number y such that $y * x = 1$?

Let us first look at integers mod 4:
the integers used are 0,1,2,3, and

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

What does it mean, x^{-1} in GF(2^8)?

A number $y = 1/x$ when we only use integers?

In other words, a number y such that $y * x = 1$?

Let us first look at integers mod 4:
the integers used are 0,1,2,3, and

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

1*1=1, ?*2=1, 3*3=1

What does it mean, x^{-1} in GF(2^8)?

A number $y = 1/x$ when we only use integers?

In other words, a number y such that $y * x = 1$?

Let us first look at integers mod 4:
the integers used are 0,1,2,3, and

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

1*1=1, ?*2=1, 3*3=1

Integers mod 4 do not have division.

What does it mean, x^{-1} in GF(2^8)?

A number $y = 1/x$ when we only use integers?

In other words, a number y such that $y * x = 1$?

Integers mod 4 does not have division

But you can arrange the arithmetics so that division does work:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

1*1=1, 3*2=1, 2*3=1

That is, 1/2=3

This is a number field (“talkropp”)

- A number field has addition, subtraction, multiplication and division. The rational numbers, reals, and complex numbers are examples of fields. Integers mod n is not a field, except when n is a prime number.
- Finite fields (Galois Fields, GF) only exist when the number of elements is a prime power p^n, so 4 and 256 are fine.
- Construction is not via integers, but rather via polynomials in a variable X with coefficients that are integers mod p.
- Choose a “primitive polynomial”, and construct the field by using polynomials modulo the chosen polynomial.

The finite field GF(4)

The primitive polynomial is $X^2 + X + 1$, and all polynomials mod this polynomial can be denoted 0, 1, X, and X+1

There are four elements, and addition and multiplication is that of polynomials (with coefficients mod 2) mod $X^2 + X + 1$

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

1*1=1, 3*2=1, 2*3=1

That is, 1/2=3

The finite field GF(4)

The primitive polynomial is $X^2 + X + 1$, and all polynomials mod this polynomial can be denoted 0, 1, X, and X+1

There are four elements, and addition and multiplication is that of polynomials (with coefficients mod 2) mod $X^2 + X + 1$

+	0	1	X	X+1
0	0	1	X	X+1
1	1	0	X+1	X
X	X	X+1	0	1
X+1	X+1	X	1	0

*	0	1	X	X+1
0	0	0	0	0
1	0	1	X	X+1
X	0	X	X+1	1
X+1	0	X+1	1	X

1*1=1, $(X+1) * X = X^2 + X = 1$, $X * (X+1) = 1$

That is, $1/X = X + 1$

The finite field GF(256)

The primitive polynomial is $X^8 + X^4 + X^3 + X + 1$.
This is not the only alternative, but it is the one used by AES

There are 256 elements, and addition and multiplication are as before, mod $X^8 + X^4 + X^3 + X + 1$

Example:

$$\begin{aligned} X^2 * (X^7 + X^6 + X^3 + X + 1) &= X^9 + X^8 + X^5 + X^3 + X^2 \\ &= X * (X^8 + X^7 + X^4 + X^2 + X) \\ &= X * (X^8 + X^7 + X^4 + X^2 + X + X^8 + X^4 + X^3 + X + 1) \\ &= X * (X^7 + X^3 + X^2 + 1) \\ &= X^8 + X^4 + X^3 + X \\ &= 1 \end{aligned}$$

That is, $1/X^2 = X^7 + X^6 + X^3 + X + 1$

The finite field GF(256)

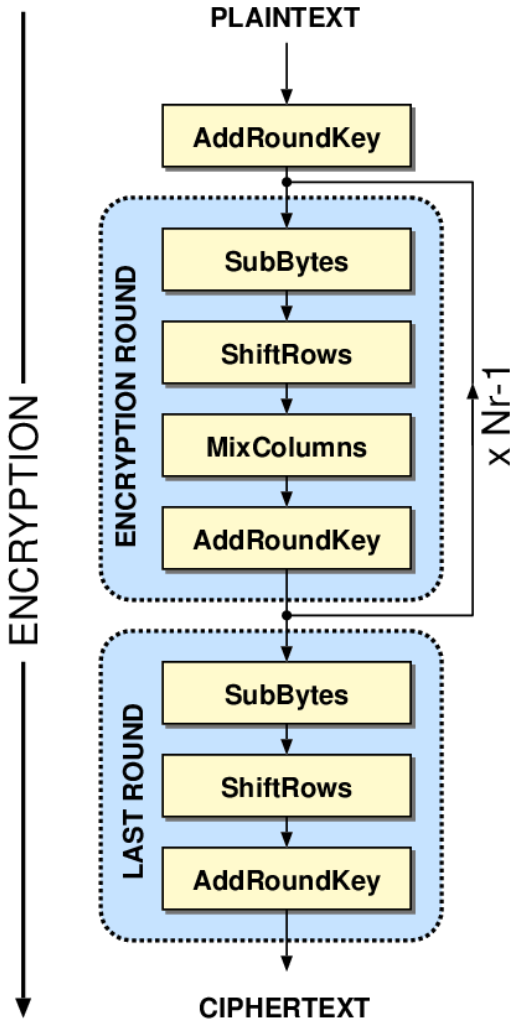
Example:

$$
\begin{aligned}
X^2 * (X^7 + X^6 + X^3 + X + 1) &= X^9 + X^8 + X^5 + X^3 + X^2 \\
&= X * (X^8 + X^7 + X^4 + X^2 + X) \\
&= X * (X^8 + X^7 + X^4 + X^2 + X + X^8 + X^4 + X^3 + X + 1) \\
&= X * (X^7 + X^3 + X^2 + 1) \\
&= X^8 + X^4 + X^3 + X \\
&= 1
\end{aligned}
$$

In binary: $(X^7X^6X^5X^4X^3X^2X^1X^0)$

$$
\begin{aligned}
100 * (11001011) &= 1100101100 \\
&= 10 * (110010110) \\
&= 10 * (110010110 \oplus 100011011) \\
&= 10 * (10001101) \\
&= 100011010
\end{aligned}
$$

AES structure: 10, 12, or 14 rounds



- Not a Feistel system
- Addition modulo 2 of the round key
- 8-bit “S-boxes”, this time chosen algebraically as x^{-1} in GF(2^8)
- Shift 8-bit parts between 32-bit sub-blocks, then mix bits in each 32-bit block
- Uses “whitening”
- The key schedule uses the S-box

AES design considerations

- Not a Feistel system, but treats parts uniformly, so reaches full diffusion faster, in two rounds
- The S-boxes are chosen algebraically to avoid controversy, and $x \rightarrow x^{-1}$ is manifestly nonlinear. MixColumn causes diffusion among the bytes
- ShiftRow was added to prohibit two recent attacks: “truncated differentials” and the “Square attack”
- The key schedule uses the S-box to avoid attacks using partial knowledge of the key bits, and avoid that two distinct keys share many round keys
- Ten rounds was chosen to give a margin: at construction, the known attacks were better than brute force at six rounds

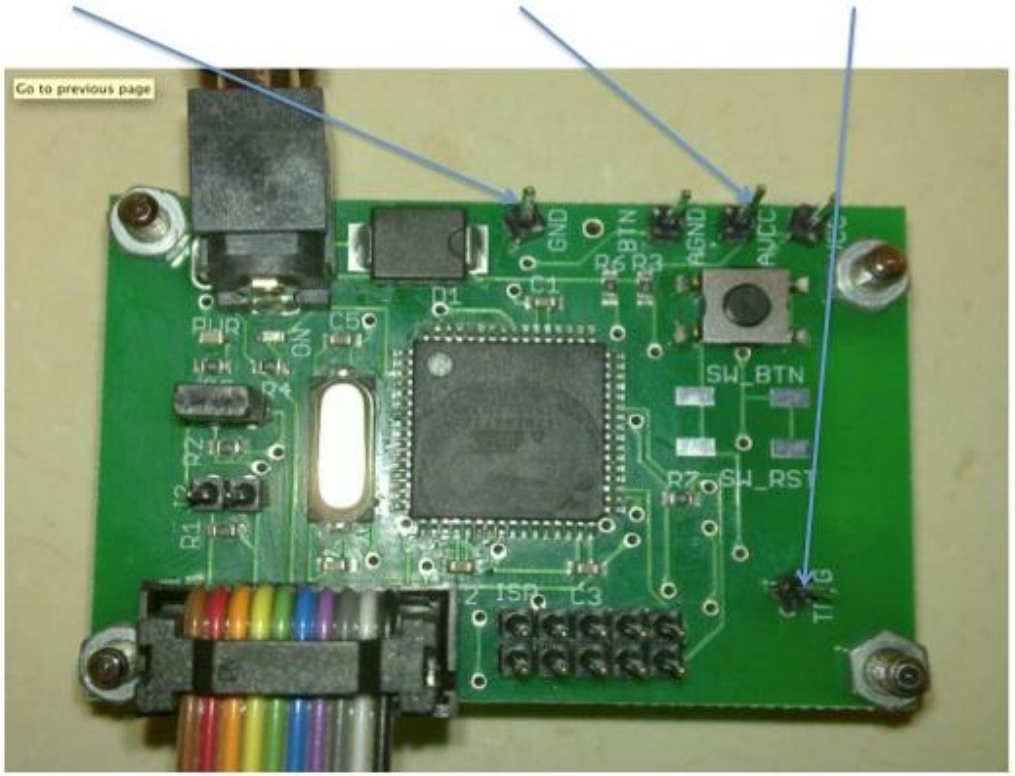
AES known weaknesses

- There are (used to only be) known attacks for 7-round AES-128, 8-round AES-196 and 9-round AES-256
- A “related-key” attack can break the full 12-round AES-196 using 2^{176} operations, and 14-round AES-256 in 2^{119} operations
- The problem is the key schedule, which seems weaker in AES-196 and AES-256 than in AES-128
- The first proper break of full AES appeared 2011. The attack is based on “bicliques”, and recovers the key from AES-128 in $2^{126.1}$ operations, from AES-192 in $2^{189.7}$ op., and from AES-256 in $2^{254.4}$ op.

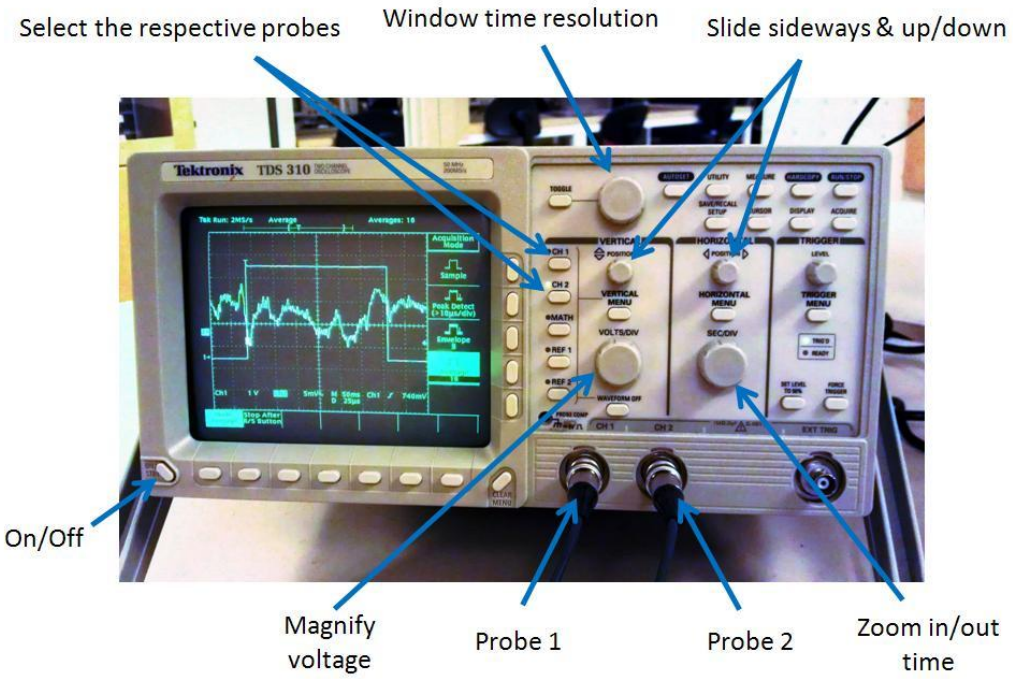
Caveat emptor: possible implementation weaknesses

- There could be side channels (röjande signaler, RÖS), depending on your implementation in software *and hardware*
- A “timing attack” measures the time it takes to encrypt (decrypt, authenticate). If the system delay depends on the key, this may leak information. AES is vulnerable in some implementations (as is RSA, . . .)
- Power consumption can also be used. Smart cards used to be vulnerable to this (and the arms race continues)
- Magnetic fields, RF emissions, interference on other data channels, deliberately slowing the system, driving it at low voltage, . . .

DES chip

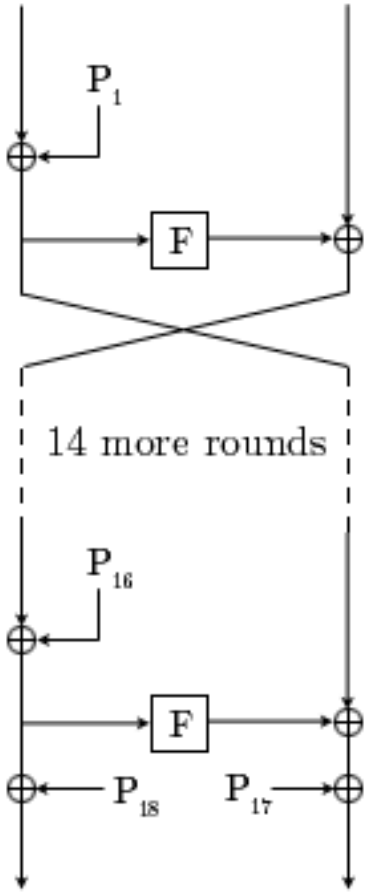

Go to previous page
GND
BTN
AGND
AVCC
R6
R3
C1
R1
C5
PWR
ON
R4
SW_BTN
SW_RST
R7
R2
R1
C3

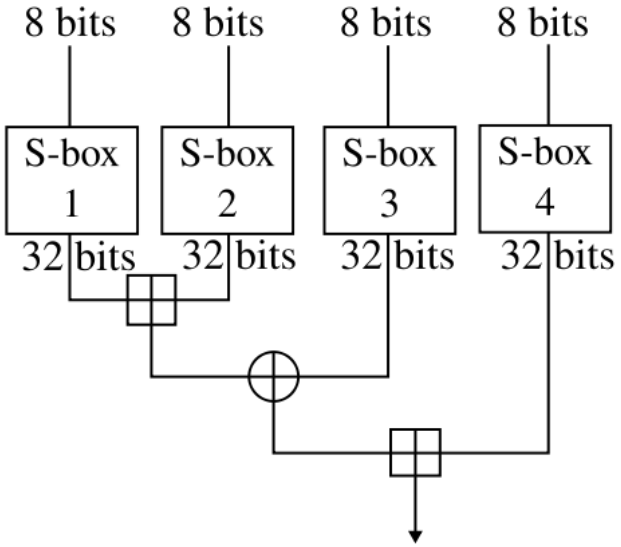
DES chip analysis



- One problem in particular is RÖS
- Used key leaking out as electromagnetic radiation
- or as in Laboration 2, as variations in power consumption

Blowfish

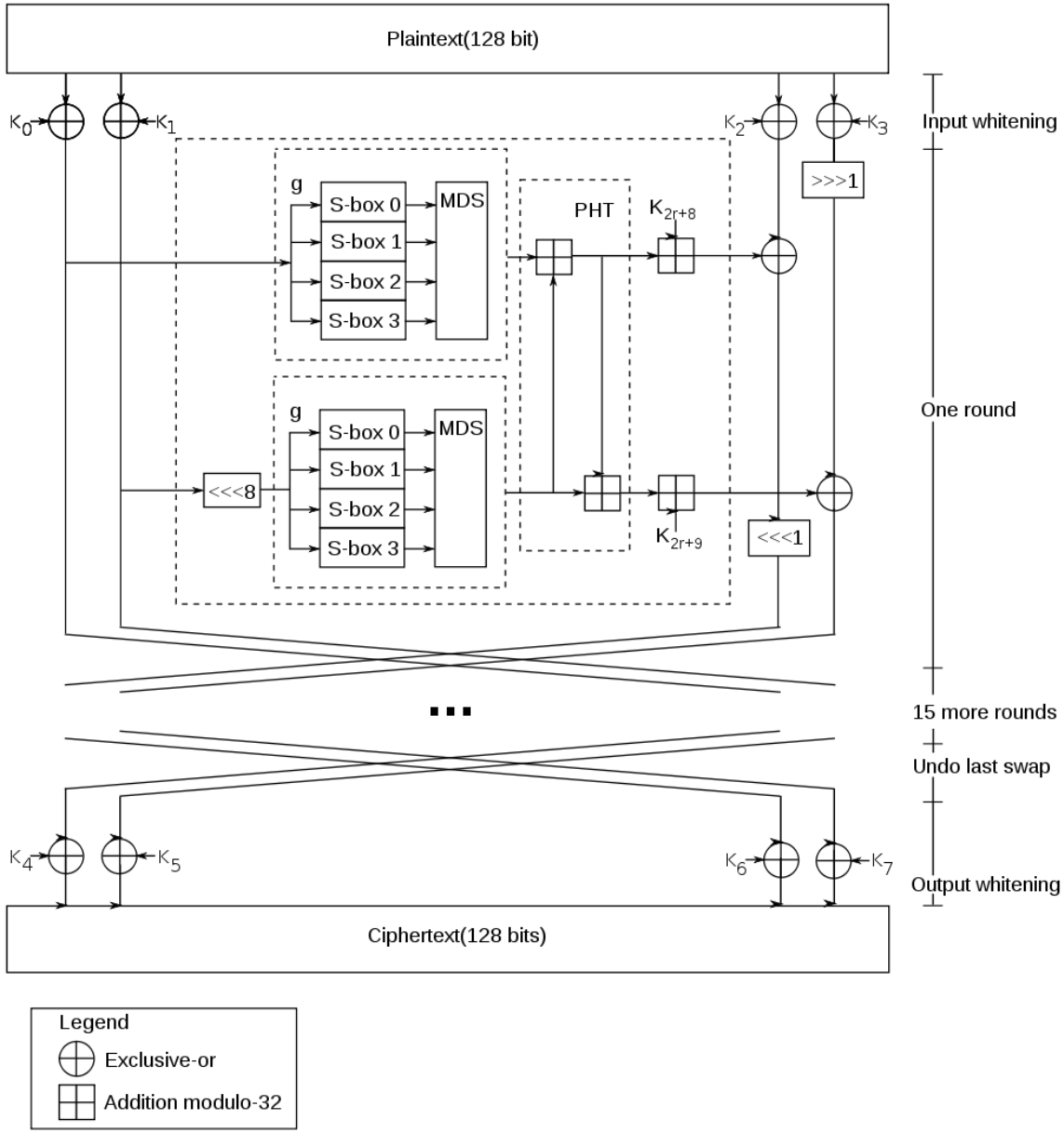




- Nonlinearity is achieved with addition mod 2 and mod 2^{32}
- The secret key is used to calculate Parray and S-boxes, using Blowfish itself 521 times
- The procedure is initialized with digits from the hexadecimal expansion of π
- Blowfish is popular, fast, and secure, but it is expensive to change the key

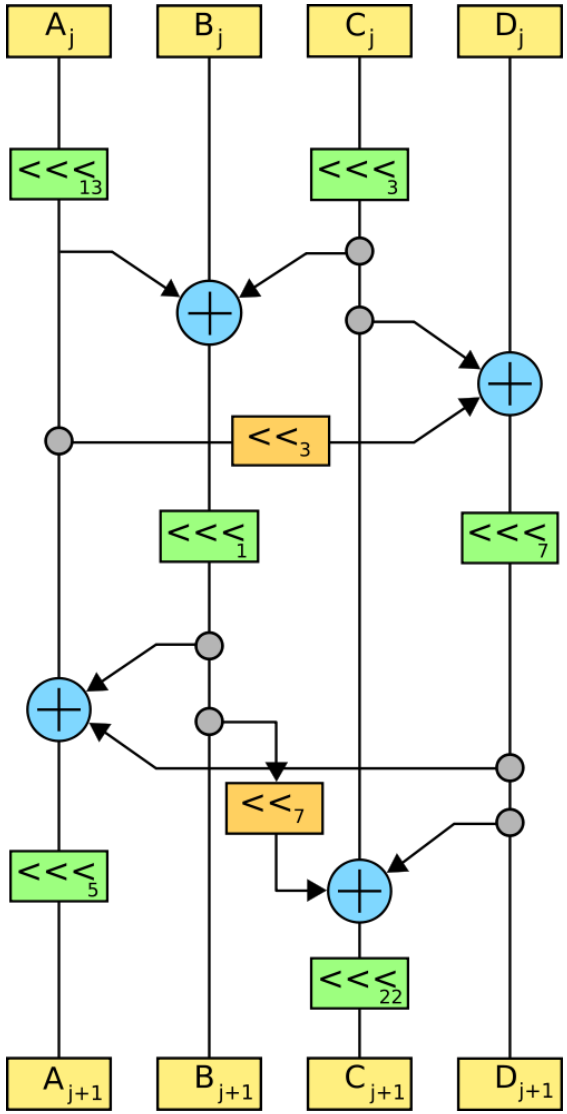
Twofish

- AES third place
- Feistel
- S-boxes depend on key
- Slower than AES
- Bigger security margin

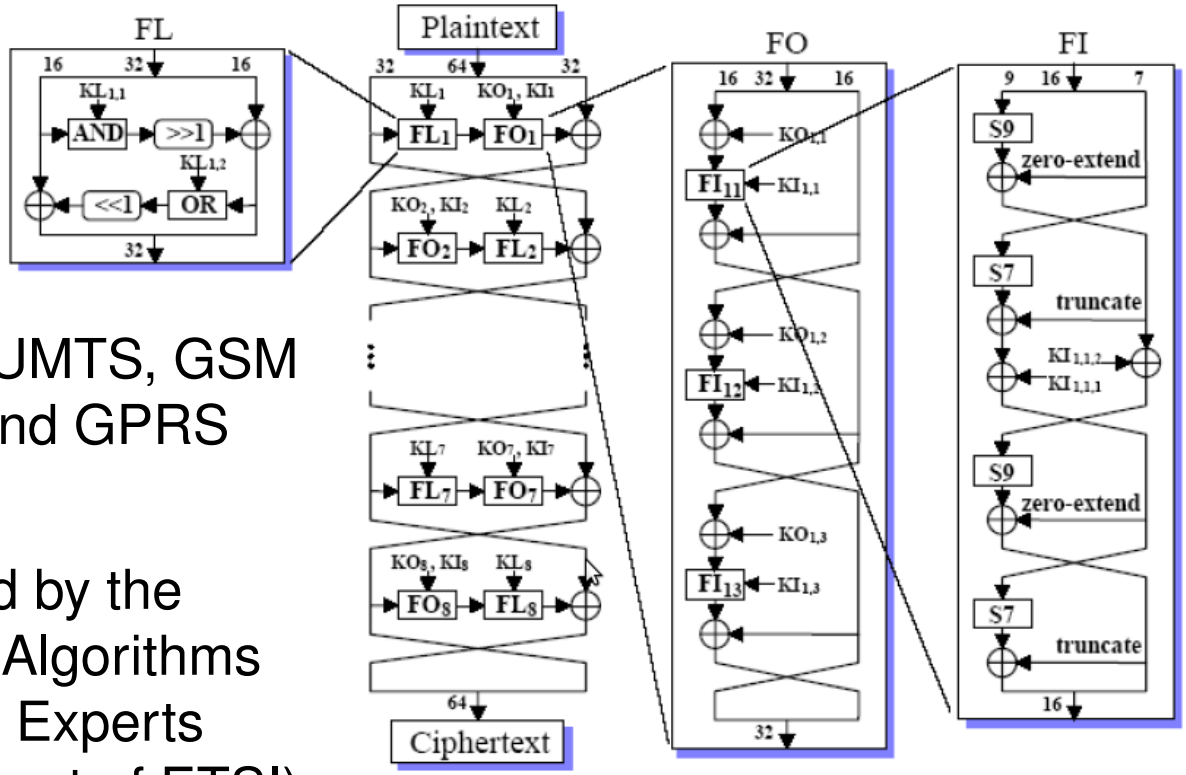


Serpent

- AES second place
- Constructed for security, not speed
- Feistel system, 32 rounds
- Four-bit S-boxes
- Adapted for parallel calculation
- Same speed as DES, 1/3 the speed of AES
- No known attacks



KASUMI



- Used in UMTS, GSM (A5/3), and GPRS (GEA3)
- Designed by the Security Algorithms Group of Experts (SAGE, part of ETSI)
- A related key attack breaks Kasumi with very modest computational resources
- Attack does not work on the designs used in UMTS, GSM, and GPRS

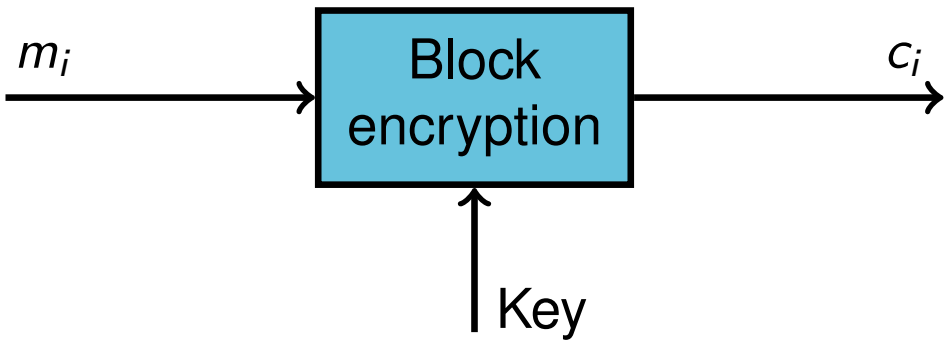
Modes of operation for block ciphers

- ECB, Electronic Codebook Mode
- CBC, Cipher Block Chaining
- CFB, Cipher Feedback
- OFB, Output Feedback
- CTR, Counter Mode

All (except ECB) use an initial value, an IV, as ”feedback” for the first block. This value should be changed between sequences for good security

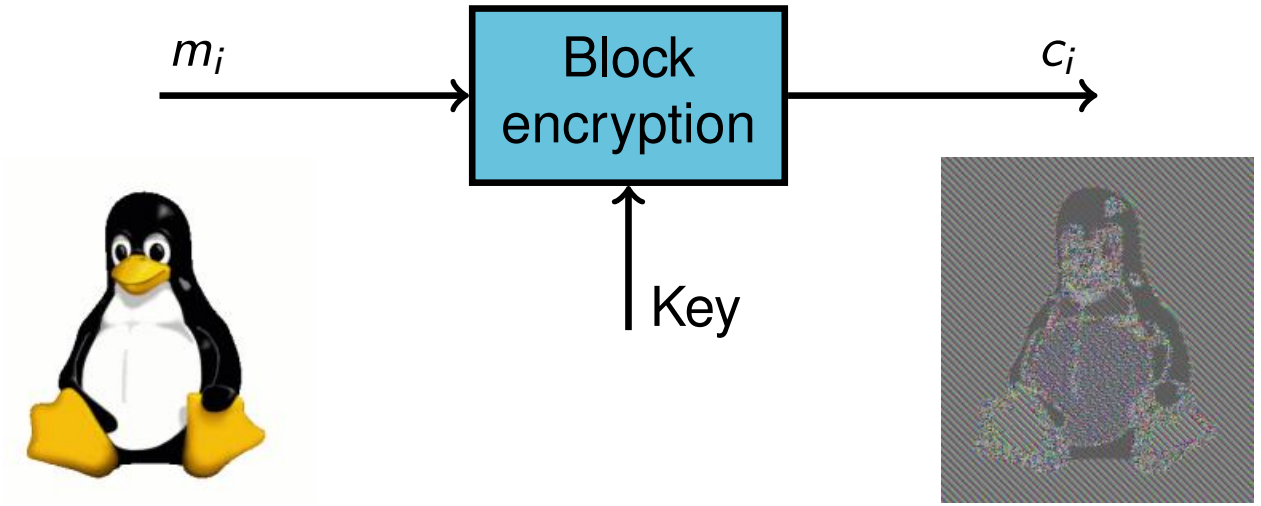
Modes of operation: ECB, Electronic Code Book

- One block at a time
- No feedback, no preserved state
- Repeated plaintext blocks produce repeated cipher blocks



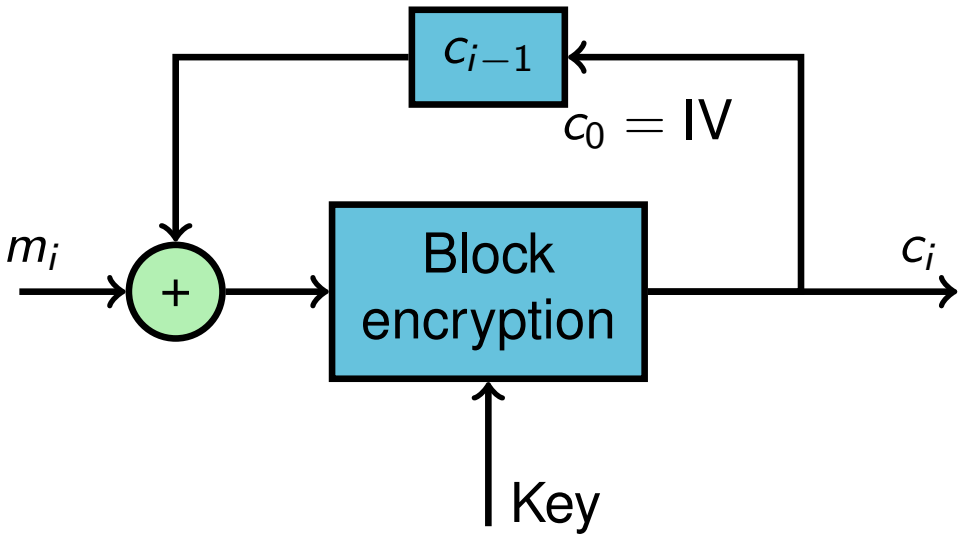
Modes of operation: ECB, Electronic Code Book

- One block at a time
- No feedback, no preserved state
- Repeated plaintext blocks produce repeated cipher blocks
- Don’t use ECB

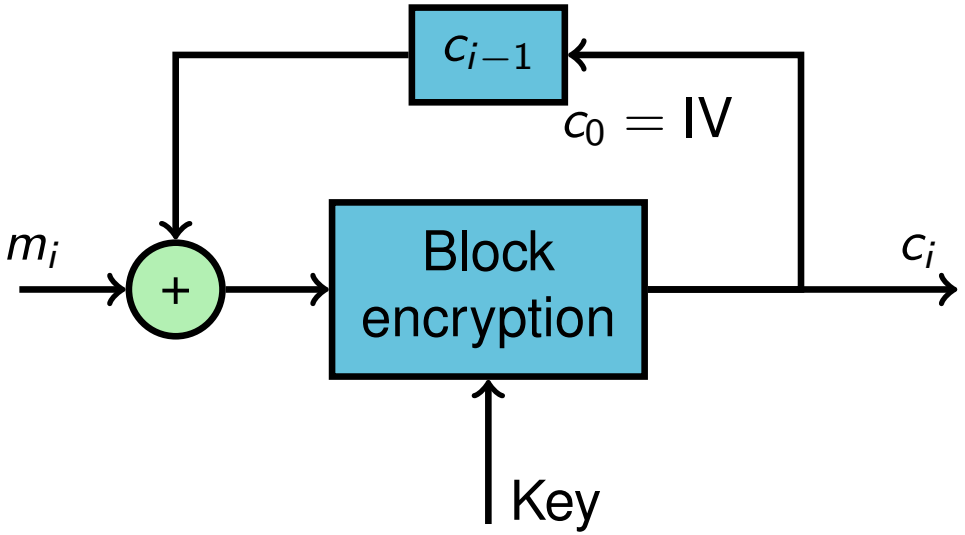


Modes of operation: CBC, Cipher Block Chaining

- Most common mode to preserve data integrity
- One whole block at a time
- Feedback of previous cipher block
- Propagates transmission errors (one bit transmission error destroys one whole block plus one more bit)



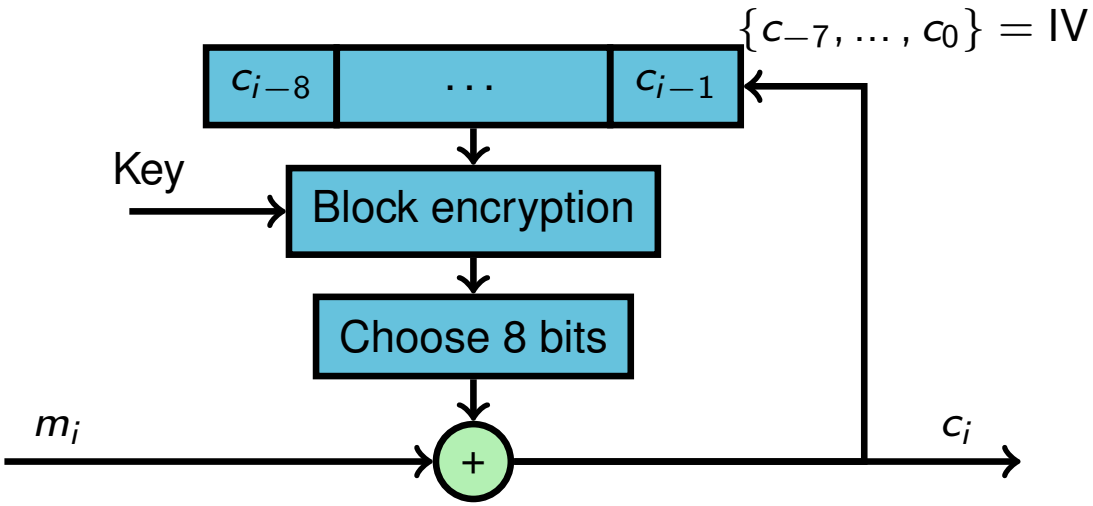
Changing the initialization vector, IV



- Don't use a fixed IV
- A counter as an IV is also usually a bad idea
- A random IV is good, but must be sent to the receiver (as a nonce)
- A nonce+counter IV has the security of a random IV and needs less communication

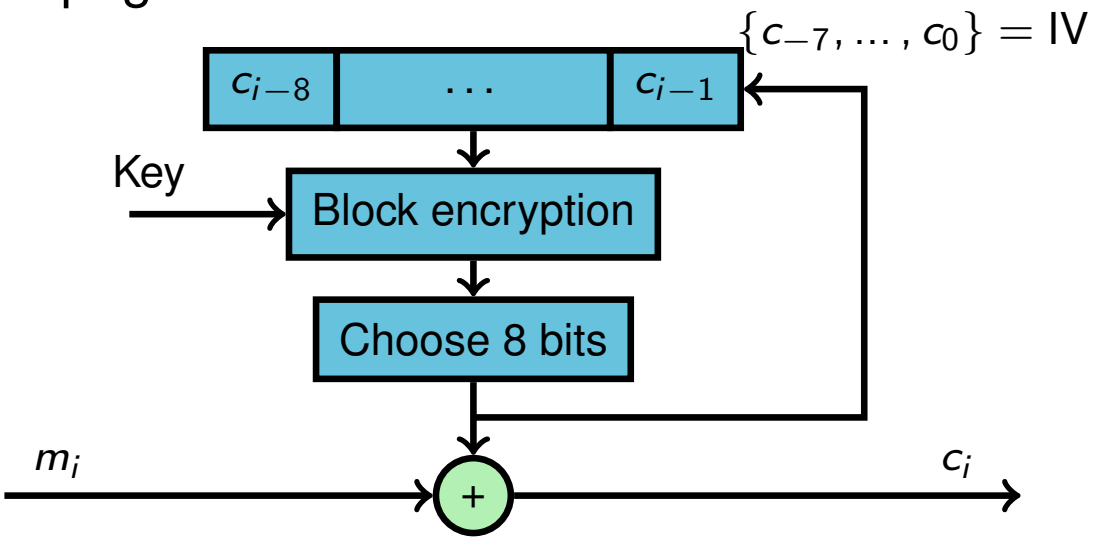
Modes of operation: CFB, Cipher FeedBack

- Useful for sending small amounts of data preserving data integrity
- Encrypts k ($<$ block length) bits at a time, usually a byte
- Picks k bits from the output and adds them to the plaintext bits to form the next cryptogram
- Uses previous cryptograms as input block
- Propagates transmission errors like CBC



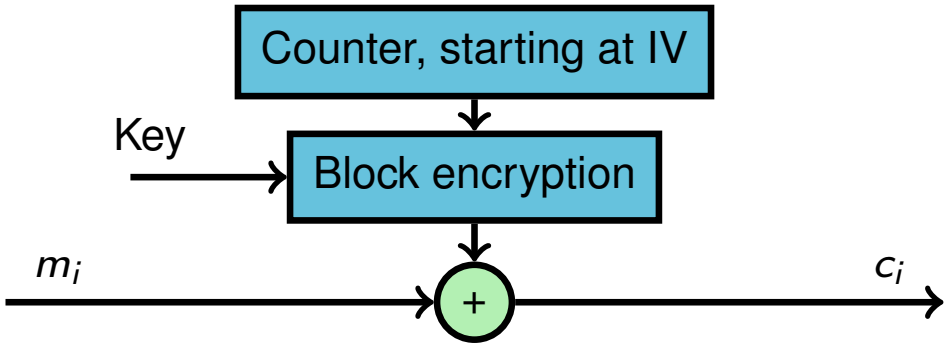
Modes of operation: OFB, Output FeedBack

- Useful for small amounts of data not preserving data integrity
- Encrypts k bits at a time, usually a byte
- Picks k bits from the output and adds them to the plaintext bits to form the next cryptogram
- Uses the same k output bits to form input in next stage, so block cipher is a pseudorandom generator
- No error propagation



Modes of operation: CTR, Counter Mode

- Useful for sending secret data without preserved data integrity
- Encrypts one block at a time
- Adds this output to the plaintext to form the next cryptogram
- Uses a counter as input, so uses block cipher as pseudorandom generator
- No error propagation



Modes of operation

- ECB, Electronic Code Book (not recommended)
- CBC, Cipher Block Chaining (common)
- CFB, Cipher FeedBack
- OFB, Output FeedBack (stream cipher, fast)
- CTR, Counter mode (stream cipher, fast, gaining popularity)

All (except ECB) use an initial value, an IV, as ”feedback” for the first block. This value should be changed between sequences for good security

Message Authentication Code

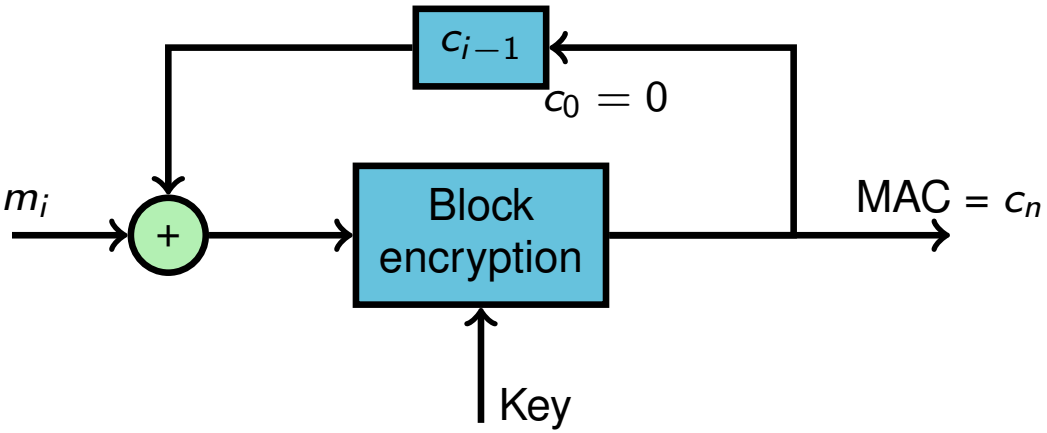
- A MAC is a function that takes two arguments, a message and a key, and generates a short tag
- These are generated and verified with the same key
- If no secret key is involved, you have a simple checksum, not a MAC (nor a signature)

Examples

- HMAC: Hash-function-based MACs
- CBC-MAC: Use (part of) the last ciphertext block of a block cipher in CBC mode
- Many many more

CBC-MAC, Cipher Block Chaining-MAC

- Run through CBC, use (part of) last block
- Vulnerable to collision (birthday) attacks
- Also vulnerable if used with variable-length messages
- Don't use the same key for encryption and authentication



MACs achieve the following:

- They ensure data integrity
- They ensure data origin down to the level of owners of the (symmetric, secret) key, but not down to one single individual
- If you are one member of a pair that owns the key, the MAC is correct, and you know that you have not created this message, then the originator must be the other member of your pair

Requirements on a MAC

- An ideal MAC behaves (from the point of view of Eve) as a random mapping from all possible inputs to all possible tags
- This requires using a secret key

- Note that the security is related to the tag length rather than the key length
- “Related” is used because the security is often given by half the tag length, rather than the entire tag length, because of *the birthday paradox*

Requirements on a MAC

- An ideal MAC behaves (from the point of view of Eve) as a random mapping from all possible inputs to all possible tags
- This requires using a secret key

- Note that the security is related to the tag length rather than the key length
- “Related” is used because the security is often given by half the tag length, rather than the entire tag length, because of *the birthday paradox*

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

One tends to be selfish in these cases and think: ”The chance that another person has the same birthday as me is 1/365. The chance that two other person has the same birthday as me is (almost) 2/365. So, close to 183.”

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

One tends to be selfish in these cases and think: ”The chance that another person has the same birthday as me is 1/365. The chance that two other person has the same birthday as me is (almost) 2/365. So, close to 183.”

But this is wrong! This calculation is correct when looking for matches *to one specific person*.

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

The correct calculation is the following: The chance that a second person *does not* have the same birthday as the first is 364/365.

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

The correct calculation is the following: The chance that a second person *does not* have the same birthday as the first is 364/365.

If the two first do not have the same birthday, the chance that a third person does not have the same birthday as the two first is 363/365.

The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

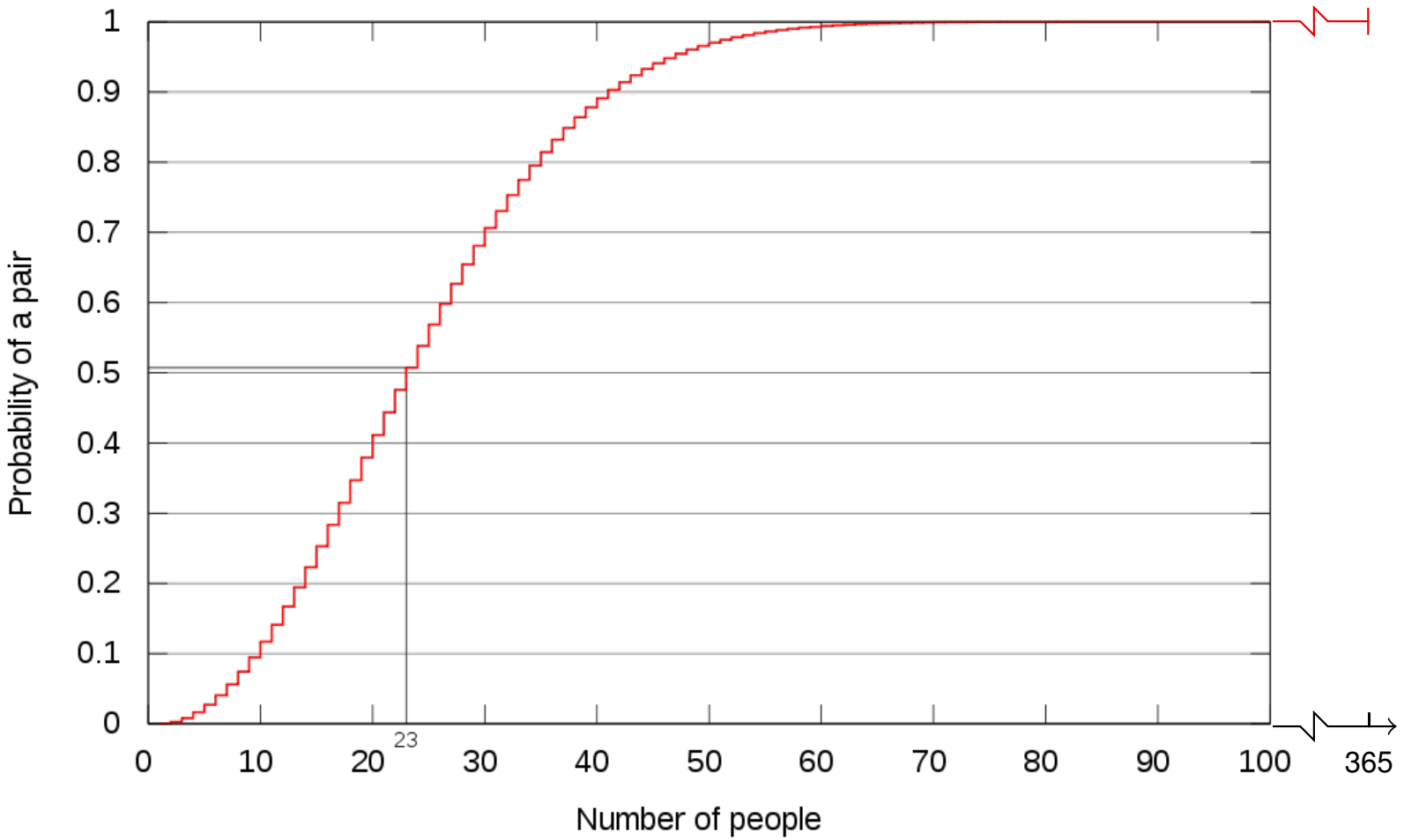
The correct calculation is the following: The chance that a second person *does not* have the same birthday as the first is 364/365.

If the two first do not have the same birthday, the chance that a third person does not have the same birthday as the two first is 363/365.

If these three do not share a birthday, . . .

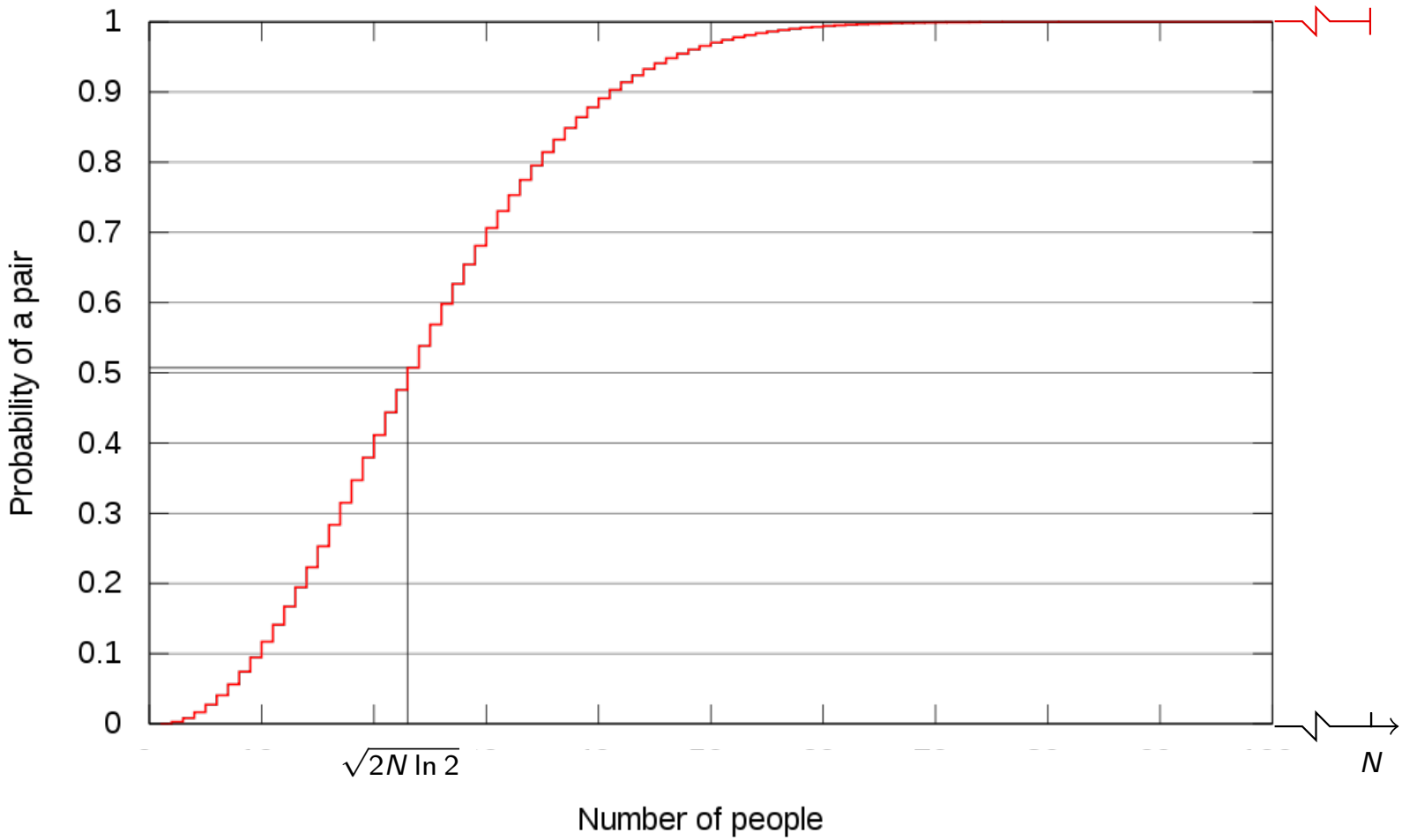
The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?



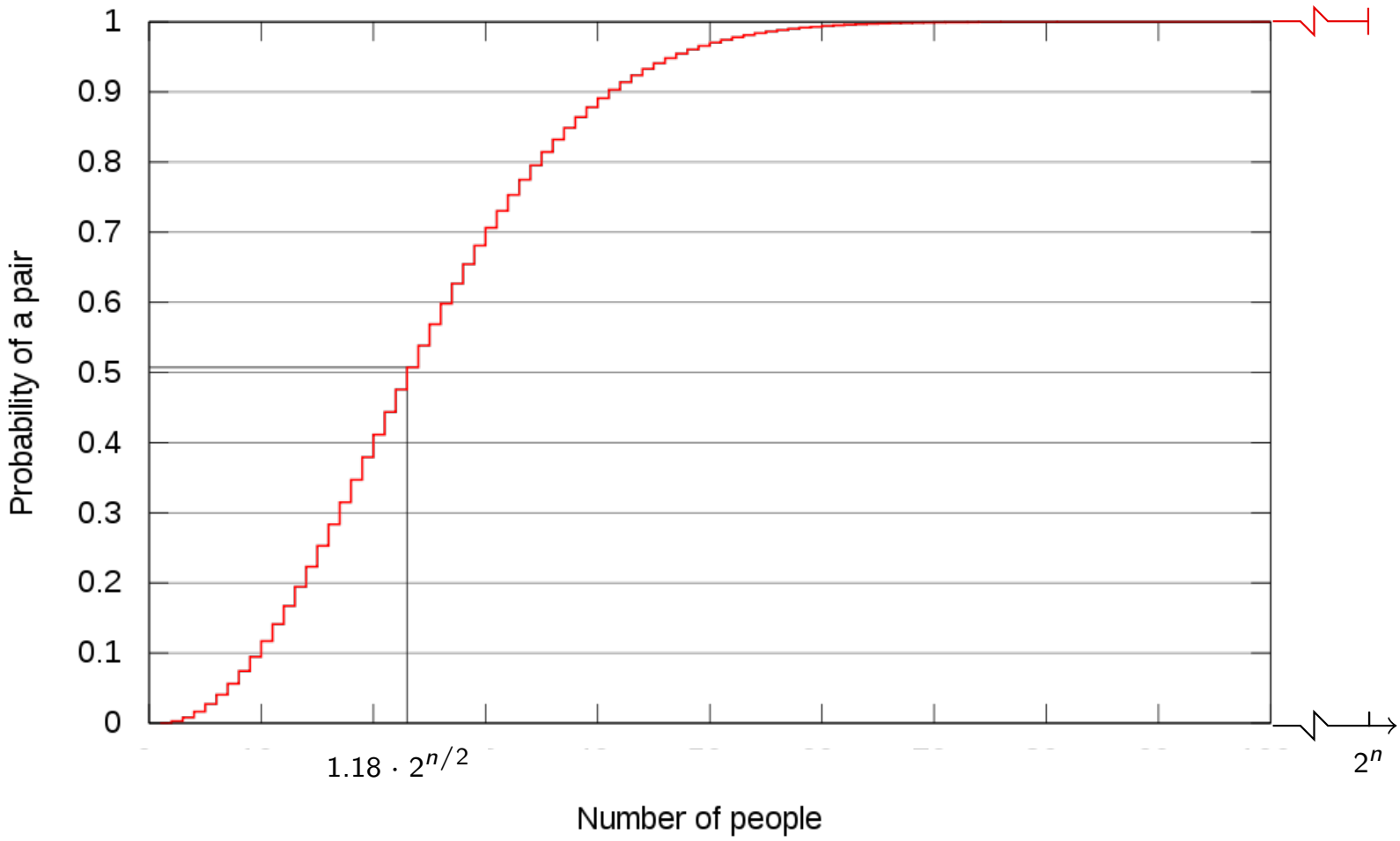
The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

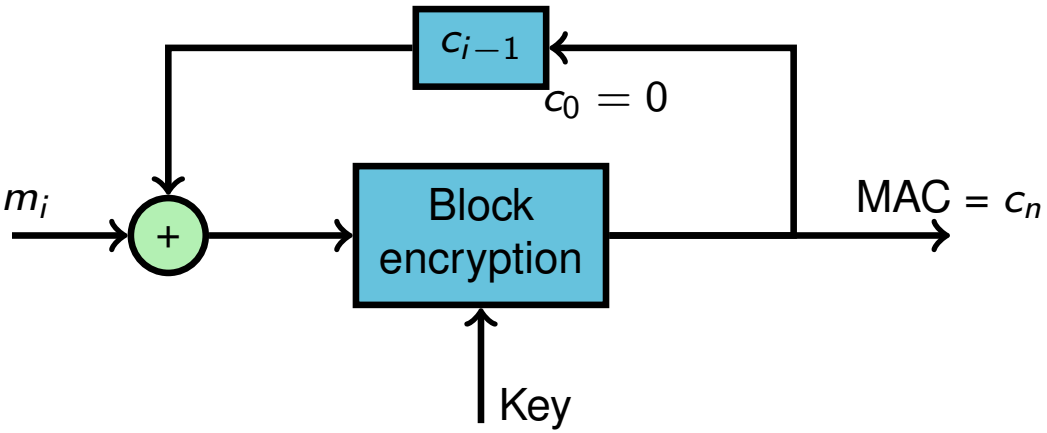


The birthday paradox

How many people must there be in a room so that the probability of two of them having the same birthday is larger than 50%?

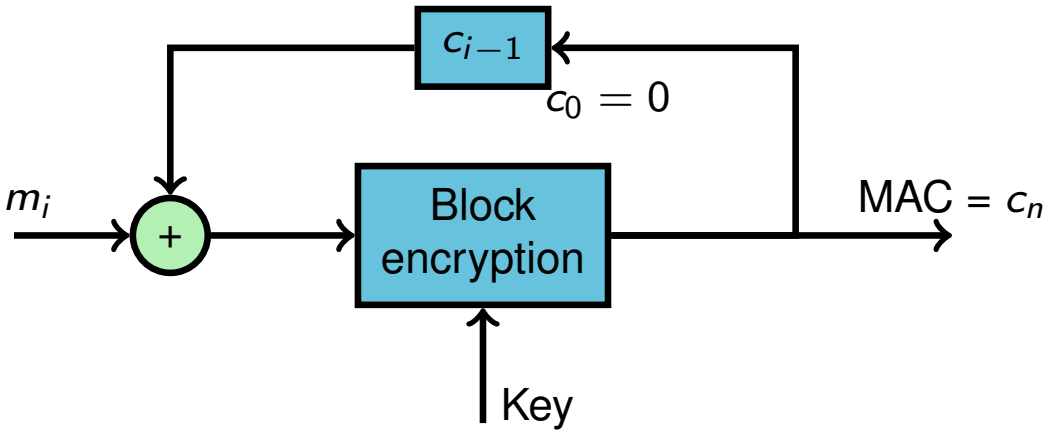


Birthday attack on CBC-MAC



- Accumulate many message+MAC pairs, wait for collision
- This will take 2^{64} steps for a 128-bit block cipher because of the birthday paradox (to find a collision, not to match a specific value)

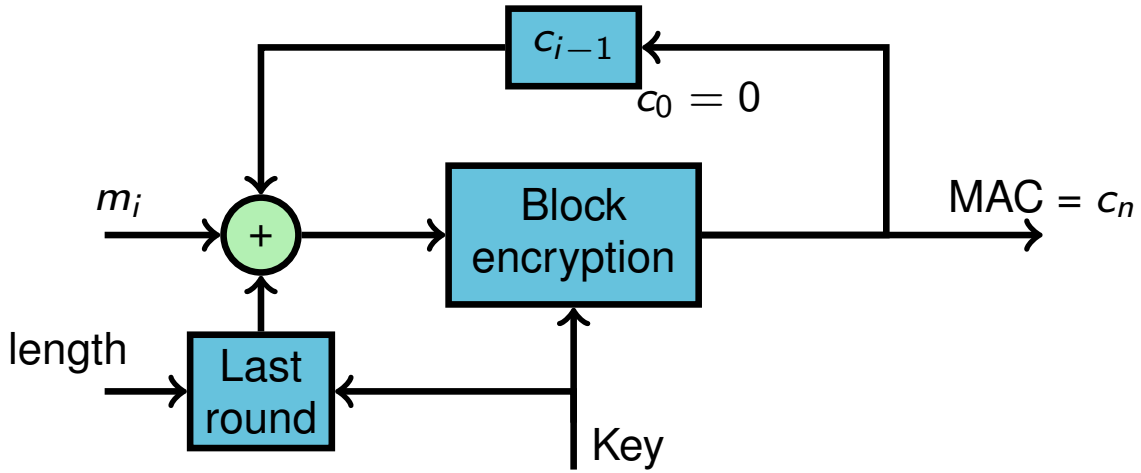
Length attack on CBC-MAC



- For one-block messages a and b, the attacker obtains $M(a)$, $M(b)$, and $M(a||b)$
- The attacker creates the message $b||(b \oplus M(a) \oplus M(b))$
- The MAC for this is also $M(a||b)$
- (exercise: check this)

CMAC

- Just as CBC-MAC, except in the last iteration where a length- and key-dependent value is XORed in
- Standardized by NIST
- Still vulnerable to the birthday attack



The Secure Channel

- If we want to do both Encryption and Authentication, which order is best?
  - Encrypt-then-Authenticate
  - Authenticate-then-Encrypt
  - Encrypt-and-Authenticate
- Use different keys for encryption and authentication in the first two cases

The Horton principle

- Authenticate what is *meant*, not what is said
- For example message content *and length*

The Horton Principle

- Authenticate what is *meant*, not what is said

- If the recipient needs extra information on how to interpret the message, that information needs to be authenticated as well
- An example is varying data field lengths in internet protocols, another example is protocol version
- In fact, in SSL 3.0, the protocol version field (in the record layer) is not authenticated, neither are (some) message boundaries; these are minor problems

Message Authentication Code

- A MAC is a function that takes two arguments, a message and a key, and generates a short tag
- These are generated and verified with the same key
- If no secret key is involved, you have a simple checksum, not a MAC (nor a signature)

Examples

- HMAC: Hash-function-based MACs
- CBC-MAC, CMAC (UMAC, GMAC, SGMAC, . . .)

Authenticated encryption

- CCM (Counter-CBC-MAC), OCB (Offset Code Book), EAX (Encrypted Authenticated X?), CWC (Carter-Wegman-Counter), GCM (Galois Counter Mode), SGCM,...

Example: Unix passwords

- Passwords are assumed to be 8 character 7-bit ASCII
- In total 56 bits
- Encrypt the message 0 using this DES key

Weaknesses

- Dictionary attacks
- DES is fast, and there is dedicated hardware

Example: Unix password weaknesses

- Dictionary attacks
  - Use a salt to make dictionary attacks more difficult
  - the book mentions 12-bit salts
- DES is fast, and there is dedicated hardware
  - Don’t have the salt in the message
  - Instead use it to change the DES algorithm (see the book)
  - This makes hardware attacks more difficult
  - . . . and slows down an attack

Example: Modern unixes

- Stronger algorithms (proper hash functions)
- Whole password significant
- Longer salts (up to 16 characters)
- Use “key stretching”: many many rounds
- Only root can read the passwd file

mkpasswd -m SHA-512 test
6EUEoZKNThDNKmfdb$3g5AuZFmWHCaDJDJq2GVPdLQ8CAOPdDUG

ID	Method
1	MD5
2a	Blowfish (not in mainline glibc; only some Linuxes, and NetBSD)
5	SHA-256 (since glibc 2.7)
6	SHA-512 (since glibc 2.7)

Next lecture

- Public key principles
- One-way functions
- Mathematics: modular arithmetic (Euclidean algorithm, Euler's totient function, Chinese remainder theorem)
- Rivest Shamir Adleman, RSA