Error detection

- Error correcting codes can be used to only detect error.

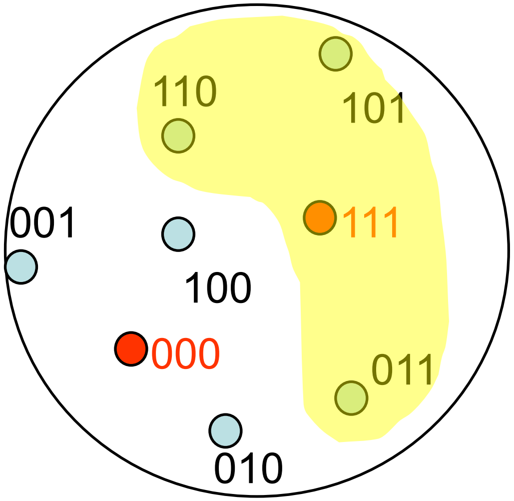
- Error detecting codes need much less redundancy.

- Example: Detect one bit error in a code block
- Message source $M = \{0,1\}^n$

- Parity code
- $m \in \{0, 1\}^n$ $m = m_1 m_2 \ldots . m_n$
- $Enc(m) = (m, p)$
- p is a parity bit; $p = \Sigma_i m_i \pmod 2$

Error detection

- Example: Rep3



- Message source M= {0,1}
- Rep3 code= {000, 111}

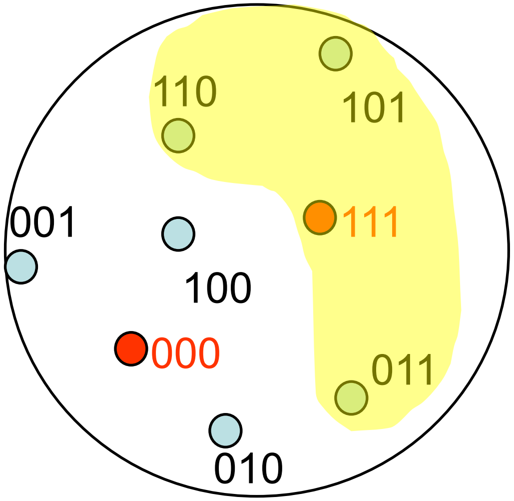
- Two errors in transmission→

000→ 110

→ More errors can be handled

Error detection

- An error correcting code can be used for error detection:



- Rep3: error correction
  - minimum distance decoding
  - 110→decoded→ 1
- Use code for error detection
- 110→ ⊥
  - Decoder outputs because 110 is not a codeword

Minimum distance of a code

- Minimum distance of a code is given by,

  Min $d_H(c,c')$ for all c, c' in C

- A code with minimum distance d,
- Corrects (d-1)/2 errors
- Detects d-1 error

Good codes

Computation:

- Efficient encoding/decoding

Information rate:

- R= k/n   for a code mapping a block of k bits to a codeword of n bits

Error detection: TCP checksum

TCP Header

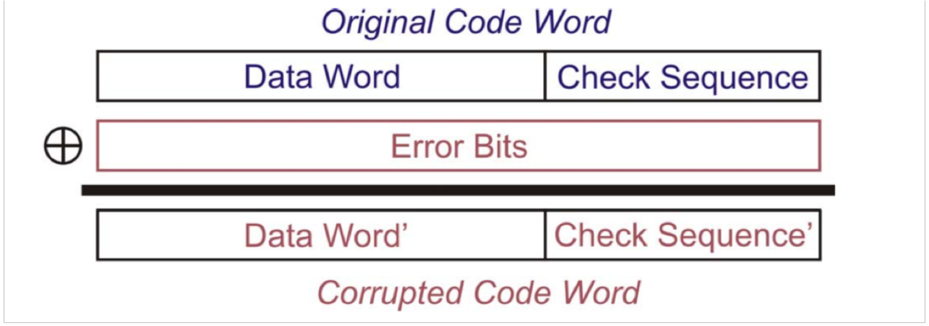
Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
128	Checksum																Urgent pointer															
160 ...	Options (if Data Offset > 5) ...																															

- IP packets can be corrupted (lost, out of order ..) . TCP detects these and requests re-transmission.

- CRC 16 (Cyclic redundancy check)
- Checksum (16 bits) –used for error-detection of the header and data.

CRC 16

- Adds a fixed length checksum to a string of arbitrary length.



- Intuition:
- Using integers (example): divide by a number and keep the remainder

- 7634153 mod 251 = 239
- 7634153 238
- 8-bit check sequence

- Not good divisors
- 7634153 mod 100 = 53 will be affected by the last two digits

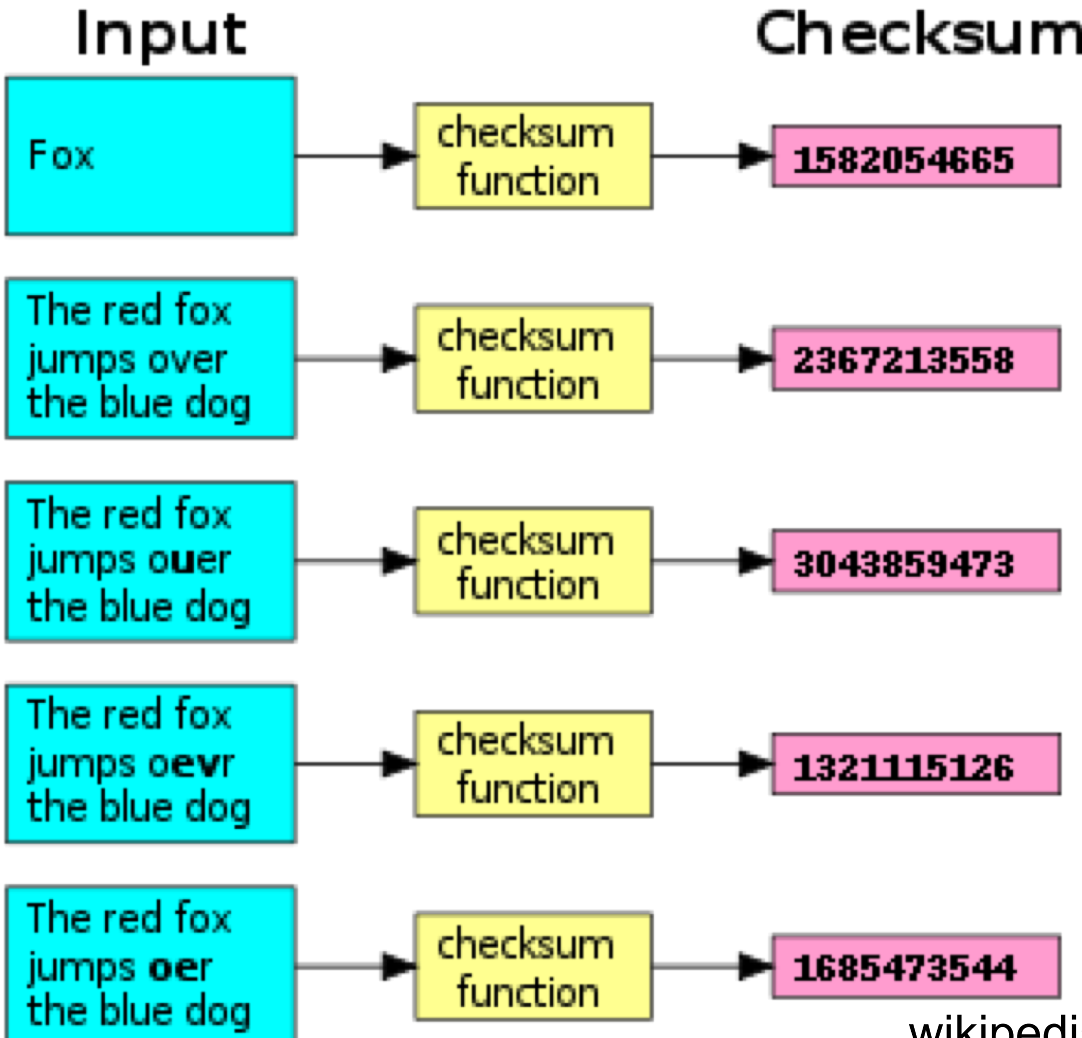
CRC 16

- Adds a fixed length checksum to a string of arbitrary length.
- It is calculated by dividing the string (appended by 16 zeros) to a specially chosen polynomial.

```
11010011101100 000 <--- Data Word left shifted by 3 bits
1011               <--- 4-bit divisor is 1011  x^3 + x + 1
01100011101100 000 <--- result of first conditional subtraction
 1011              <--- divisor
00111011101100 000 <--- result of second conditional subtraction
  1011             <--- continue shift-and-subtract ...
00010111101100 000
   1011
00000001101100 000
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------   Remainder is the Check Sequence
00000000000000 100 <--- Remainder (3 bits)
```

[Wikipedia]

Error detection: cksum command in Unix



wikipedia

Command is for error detection in files.

cksum test.py
2365581169 34 test.py

Error correction: Hard Drive

Error rates and handling [edit]

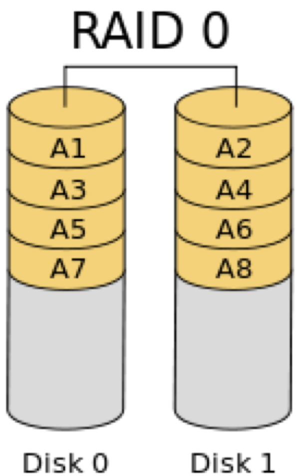
Modern drives make extensive use of error correction codes (ECCs), particularly Reed–Solomon error correction. These techniques store extra bits, determined by mathematical formulas, for each block of data; the extra bits allow many errors to be corrected invisibly. The extra bits themselves take up space on the HDD, but allow higher recording densities to be employed without causing uncorrectable errors, resulting in much larger storage capacity.[38] For example, a typical 1 TB hard disk with 512-byte sectors provides additional capacity of about 93 GB for the ECC data.[39]

In the newest drives, as of 2009, low-density parity-check codes (LDPC) were supplanting Reed-Solomon; LDPC codes enable performance close to the Shannon Limit and thus provide the highest storage density available.[40]

wikipedia

Communication systems commonly use error detection.
Storage systems commonly use error correction.

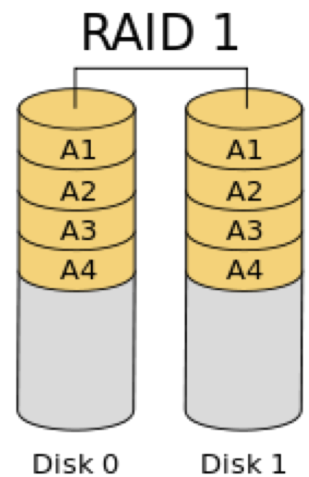
Error Correction:
RAID (Redundant Array of Independent Disks)



RAID-0

- No redundancy

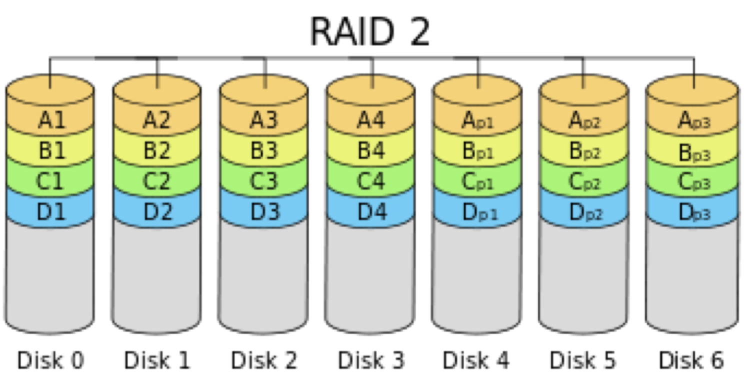
- data “striped” across different drives



RAID-1

- Mirroring

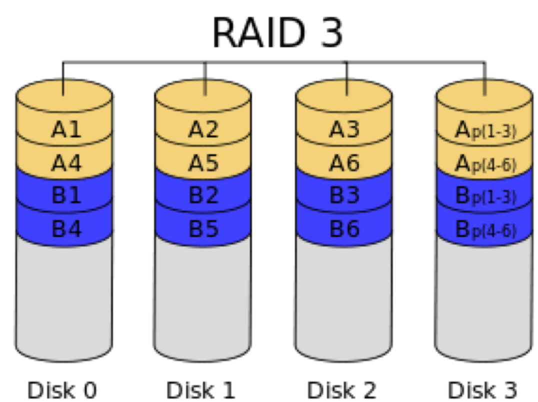
RAID combines multiple physical disk drive into a single logical unit



RAID-2

- ECC (Hamming code)

- not practically used



RAID-3

- Parity disk

- Bit-interleaved parity

Wikipedia

Summary

- Message integrity
  - Noise
  - Adversarial

- Protection goals:
  - Detection
  - Correction

Noise corruption

•Error detecting/correcting codes

•Block coding
  - Hamming code
  - CRC 16

•Error control strategies
  - Error correction
  - Detection (and retransmission)

Message integrity: adversarial corruption

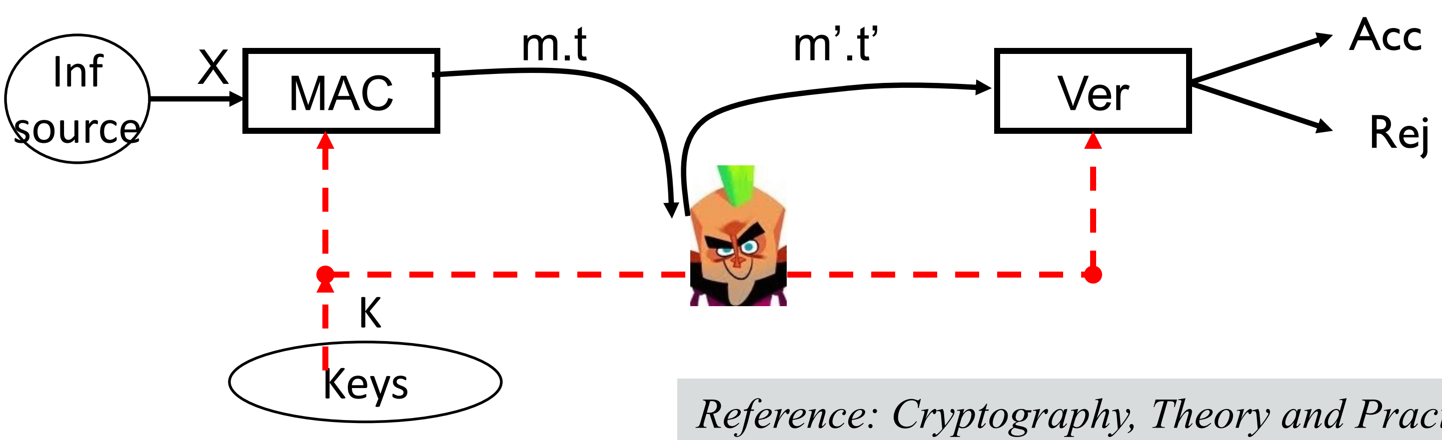
- Error correcting/detecting codes, checksums, CRC 16…

do not protect against tampering.

Message Authentication Code

- Message authentication codes (MAC) provide protection against message tampering.

- Need shared secret key.



Reference: Cryptography, Theory and Practice, Douglas Stinson

Secrecy does not imply integrity

- Secrecy and authentication are different properties.
- Secrecy $\not\Rightarrow$ integrity

- One-time pad: perfect secrecy

Encryption and Decryption:

$$Enc(k,x) = k_i \oplus x_i = y_i, \qquad i = 1, \cdots n$$

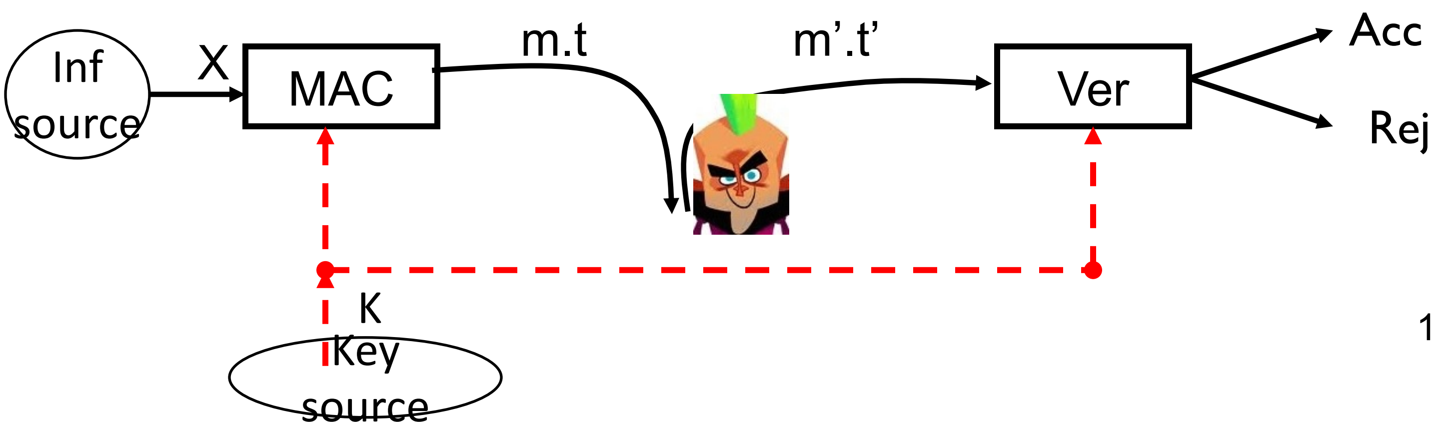
$$Dec(k,y) = k_i \oplus y_i = x_i, \qquad i = 1, \cdots n$$

- Message bits can be flipped without receiver detecting
- Enc(k,x) $\oplus$ **1** = ($k_i \oplus x_i$) $\oplus$ 1 = y'_i
- Dec(k,y) = $k_i \oplus$ ($k_i \oplus x_i$) $\oplus$ 1 = ($k_i \oplus k_i$) $\oplus$ $x_i \oplus$ 1 = $x_i \oplus$ 1

Message Authentication Code

- Two algorithms MAC(m,k) ,Ver ((m,t),k)
- Message space : M
- Tag space: T
- Key space: K

- MAC(m, k)= t
- Ver((m,t), k) =1 if (m,t) is valid for k ;
  0, otherwise.



Example

- $M=\{m_1,m_2,m_3\}$, $K=\{k_1,k_2,k_3,k_4\}$
  $T=\{a,b\}$

- (m_1 , b) is valid for k_2 and k_4
- (m_1 , a) is valid for k_1 and k_3

- (m_2 , b) is valid for k_1 and k_2
- (m_2 , a) is valid for k_3 and k_4

- (m_3 , b) is valid for k_3
- (m_3 , a) is valid for k_1, k_3 and k_4

	m_1	m_2	m_3
k_1	a	b	a
k_2	b	b	a
k_3	a	a	b
k_4	b	a	a

Encoding matrix

M x T

	m_1,b	m_1,a	m_2,b	m_2,a	m_3,b	m_3,a
k_1	0	1	1	0	0	1
k_2	1	0	1	0	0	1
k_3	0	1	0	1	1	0
k_4	1	0	0	1	0	1

Two representations of encoding matrix

Example

- $M=T=Z_3$, $K=Z_3 \times Z_3$,
- $MAC(m; (i,j)) = m.i + j \bmod 3$
- $Ver((m,t), (i,j)) = 1$, if $t= m.i+j$

k=(i,j) \ m	0	1	2
$K_1=(0,0)$	0	0	0
$K_2=(0,1)$	1	1	1
$K_3=(0,2)$	2	2	2
$K_4=(1,0)$	0	1	2
$K_5=(1,1)$	1	2	0
$K_6=(1,2)$	0	1	2
$K_7=(2,0)$	0	2	1
$K_8=(2,1)$	1	0	2
$K_9=(2,2)$	2	1	0