Master Electronique, énergie électrique, automatique
Parcours MiSCIT: Master In Systems, Control and Information Technologies

Security of Networks

Jean-Marc THIRIET
jean-marc.thiriet@univ-grenoble-alpes.fr
http://www.gipsa-lab.grenoble-inp.fr/~jean-marc.thiriet/miscit/miscit_en.html

Security of networks

- 1. Error detections codes
- 2. Introduction to dependability, security principles, risk analysis
- 3. Cyber-Attack
- 4. Cyber-Protection: infrastructures
- 5. Cyber-protection: application of cryptography

Cyber-security

- **Cybernetics** from Greek κῠβερνήτης (*kubernêtês*) => Used 19th and 20 century => ideas of control and communication
- **Cyber** : Greek *kubernân,* to govern
- Today used for everything relative to the « digital » world (internet, web…)
- **Cyber-security**:
  - Cyber security comprises technologies, processes and controls that are designed to protect systems, networks and data from cyber attacks. Effective cyber security reduces the risk of cyber attacks, and protects organisations and individuals from the unauthorised exploitation of systems, networks and technologies (https://www.itgovernance.co.uk/what-is-cybersecurity)
  - État recherché pour un système d'information lui permettant de résister à des événements issus du cyberespace susceptibles de compromettre la disponibilité, l'intégrité ou la confidentialité des données stockées, traitées ou transmises et des services connexes que ces systèmes offrent ou qu'ils rendent accessibles. La cybersécurité fait appel à des techniques de sécurité des systèmes d'information et s'appuie sur la lutte contre la cybercriminalité et sur la mise en place d'une cyberdéfense. (ANSSI, https://www.ssi.gouv.fr/entreprise/glossaire/c/)

Université Grenoble Alpes

2 Errors Detection and Correction

2.1 Error Detection

2.2 Vertical and longitudinal Parity

2.3 Cyclic or polynomial Code (polynomial arithmetics)

2.4 checksums

Ojectives of this chapter

- Review the concepts of error coding and detection
- First level of integrity violation detection
- Revise binary and hexadecimal
- Revise the concept of modulo division (remainder of the integer division) widely used in cryptography…

2.1 Error detection and correction

- Frames Transmissions
  - To ensure the receiver has received correctly information
  - Control of the **integrity** of the received data
- Causes of transmission errors
  - Physical aspects
    - Thermal noise
    - Electromagnetic perturbations
    - Distance between two devices (wireless, mobility)
  - Characteristics of the protocols
    - CSMA/CD
    - Mobility (wireless, mobile telephony, mobile objects)

Type of methods

- BER (**Bit error rate**)
  - Single-bit errors
  - Groups of contiguous strings of bit errors (**burst errors**)
- BER : probability P of a single bit being corrupted in a defined time interval
- BER of $P=10^{-3}$ means that, on average, 1 bit in 10^3 => corrupted
- For a string of N bits => $1-(1-P)^N$
  - With $P=10^{-3}$ and N=10 =>the probability of the string to be corrupted => 10^{-2}

Choice of the method

- Based on
  - the size of blocks (packets, frames)
  - Probability of errors
- Ex :
  - Block of 1250 bytes with $P=10^{-3}$
    - $1-(1-P)^N = 1-(1-10^{-3})^{1250*8}$= 100 % errors !!
  - Block of 125 bytes with $P=10^{-3}$
    - $1-(1-P)^N = 1-(1-10^{-3})^{125*8}$= 63 % errors !

      (more than 1 block among 2 !)
- The length of the block (frame) is too long
- The maximal size of a packet to be transmitted depends on the quality of communications (error rates)

2.1 Hamming distance

- Number of positions having different bit values between two words (the words have the same length)
  - Ex : 1 0 1 1 1 0 1
    1 **1** **0** 1 1 **1** 1

  Here the Hamming distance is 3
  - If the distance between two words is d, d errors can transform one word into another one
- Generally, to detect "d" errors, a code with a "d+1" distance is necessary

Example of parity

- Parity
- A=01
- B=10
- ABBA
- 01101001

A=10

B=01

- 10101000
- BBBA

2.2 Detector codes

Example: parity control

Even parity : 10110111

parity bit

Odd parity: 10110110

- Vertical Parity (VRC)
- Longitudinal or Column Parity (LRC)

	0	1	0	0	0	1	1	**0**
	1	0	1	1	1	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

(Odd parity)

	0	1	0	0	0	1	1	**0**
	1	0	1	1	**0**	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

Parity error in one line and one column

Homework 2, due 23rd September 2022

- -Thursday 10th September 8pm at last
- 10110
- 11011
- 11001
- Provide the parity (even parity) bits both horizontically and vertically
- The receiver will receive (even parity)
- 111001
- 101110
- 001111
- 001000
- Are there transmission errors? If yes, am I able to correct the transmission errors? What is the corrected message ? Make an extra comment.

2.3 Modulo 2 polynomial arithmetics

- A polynomial represents a bit suite:

  110001 ------- > x^5+x^4+1

- Addition and soustraction **without any carry**: EXCLUSIVE OR

	1	0	0	1	1	0	1	1	$x^7 + x^4 + x^3 + x + 1$
+	1	1	0	0	1	0	1	0	$+ x^7 + x^6 + x^3 + x$
=	0	1	0	1	0	0	0	1	$= x^6 + x^4 + 1$

  - **Cyclic codes: CRC (Cyclic Redundandy Check)**

- **Transmitter**
  - **Data (message): bits suite represented by M(x)**
  - **The transmitter divides x^r.M(x) by G(x) with G(x) (degree r, r+1 bits), generator polynomial**
  - **x^r.M(x) = G(x).Q(x) + R(x), the size (number of bits) of R(x) is exactly r bits**
  - **Let's transmit the frame T(x) = x^r.M(x) + R(x)**
- **Receiver**
  - **The receiver divides T(x) by G(x) and the remainder should be 0**

Example of CRC

- 10101 =>M(x)=X4+x2+1
- Let's consider G(x)=X2+1
- r=2 ; x^r.M(x) = x6+x4+x2
- x6+x4+x2 ___x2+1_____
- -(x6+x4) x4+1
- _____x2
- -(x2+1)
- -1
- Remainder is -1 =>Remainder is 1
- This remainder will be transmitted together with the message, it is the CRC
- Size of the remainder : size is 2 , the degree of the remainder is 1
- The value of the remainder is 01 => R(x)=01
- The frame is obtained from **T(x) = x^r.M(x) + R(x) =>** 1010101

2.3 Normalised generator polynomials

- **CRC12 generator polynomial = $x^{12} + x^{11} + x^3 + x^2 + x + 1$**
- **CRC16 generator polynomial = $x^{16} + x^{15} + x^2 + 1$**
- **CRC-CCITT generator polynomial = $x^{16} + x^{12} + x^5 + 1$ (V41 recommendation used in HDLC protocol)**
- **CRC-32 (Ethernet) generator polynomial : = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$**

2.3 Example of a CRC calculation by the transmitter

- M : 1101011011
- Generator polynomial G : 10011
- What is the size of the remainder?
- Demonstrate, by using the polynomial division, that the remainder is ????
- The transmitted message is so:
  - 1101011011????
- What is the size of the remainder and why?

2.3 Example of a CRC calculation by the transmitter

- M : 1101011011
- Generator polynomial G : 10011, so the degree is 4=> the degree of the remainder will be 3 (4-1) so the size will be 4 bits
- Demonstrate, by using the polynomial division, that the remainder is 1110
- The transmitted message is so:
  - 11010110111110

2.3 Role of the receiver ?

- It receives 1101011011**1110** and divides by the generator polynomial 10011
  - ⇒The remainder should be 0,
  - ⇒If it is not the case, each bit which is at 1 in the remainder corresponds to an erroneous bit
    - ! Under the condition to respect the rule ! If the remainder contains n bits, it is possible at best to detect and correct n errors, here 4 !
- Ex : I receive 1101001011**1110** with G(x)=10011 => Is there an error?

2.4 Checksums

- IP and TCP headers used in Internet networks uses a simpler method (easier to implement): CHECKSUM (*somme de contrôle*).
- RFC 791:
  - *The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header.*
- From RFC 1071 (RFC (« standards ») which defines calculation methods for checksum in IP environment)

  - Take a 16-bit words
  - Calculate the *one's complement sum* (! sum to which the carry is directly added to the result !)
  - At the end, determine the *one's complement* of the result (! Inversion of all the bits of the result !)
  - Then this result should be placed in the checksum field

Example

- Let's calculate a 4bit-CS on this set of data
  - 1011101101100
  - 1011
  - 0110
  - 1100
  - 11101 <= Sum
  - 1
  - 1110 <= Sum with the carry
  - 0001 => 4-bit CS

Example

- The receiver will receive
  - 1011011011000001

  - 1011
  - 0110
  - 1100
  - 0001
  - 11110 <= Sum
  - 1
  - 1111 <= Sum with the carry
  - 0000 => The result => no transmission error

2.4 Checksums

- The receiver achieves the same operations (checksum received included) => the result should be 0000 after the final inversion, which means No Errors!
- Ex : Calculation of the 4-bit checksum of 1110 0011

- 0xE + 0x3
  - Normal calculation 0xE + 0x3 = 0x11 (0b10001)
  - Calculate the 4-bits *one's complement sum* of 0xE + 0x3 = 0x2 = $(0010)_2$
- Ex : The *one's complement* of this result is $(1101)_2$, 0xD

2.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00
  (00 00 is the checksum field, 0000 at this stage, it will be filled after)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7

- 0100
- F203
- F303
- F4F5
- 1E7F8

- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7

2.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00
  (00 00 is the checksum field)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the *one's complement sum* :
- DEEF + 002 = DEF1

2.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00
  (00 00 is the checksum field)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the *one's complement sum* :
- DEEF + 002 = DEF1
- The checksum is the *one's complement* of the result:
- ~DEF1 = 210E
- The sent packet includes the checksum:
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- For the receiver:
- 0100+F203+ F4F5+ F6F7 + 210E =

2.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00
  (00 00 is the checksum field)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the *one's complement sum* :
- DEEF + 002 = DEF1
- The checksum is the *one's complement* of the result:
- ~DEF1 = 210E
- The sent packet includes the checksum:
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- For the receiver:
- 0100+F203+ F4F5+ F6F7 + 210E = 0002 FFFD
- FFFD + 0002 = FFFF
- It is correct !
- Reverse FFFF => 0000

2.5 Conclusion

- BER
- Size of blocks (frames)
- Parity better for random distribution of errors
- CRC better for bursts of errors
- Estimation that 999 errors out of 1000 are corrected thanks to CRC
  - If the BER on the physical medium is 10^{-6}, it becomes 10^{-9} thanks to the CRC algorithm

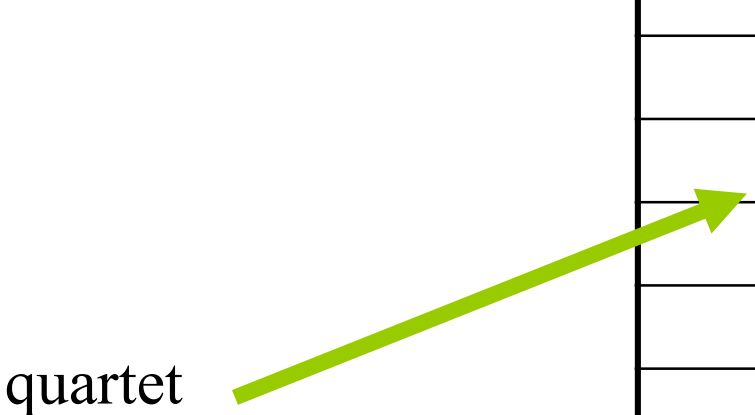
Conversion rapide décimal/binaire/hexadecimal

- Décimal/binaire
  - 125 en binaire

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	1	1	1	0	1

Conversion rapide décimal/binaire /hexadecimal

Base 2	Base 16	Base 10
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15



Références

- Transmissions et réseaux, S. Lohier & D. Présent, Dunod, Paris, 2003.

Exercises

- 1. A receiver receives (2D-odd parity) this message
- 110011
- 101100
- 001011
- 001011
- Are there transmission errors? If yes, is it possible to correct the transmission errors? What is the corrected message ?

- 2. We receive the following message: 110101011010 using the following generator suite: 1101.
- Explain how the frame is built: which bits are representative of the data and which of the CRC?
- Is the message correctly received?
- If not, what can/should we do?

Exercise

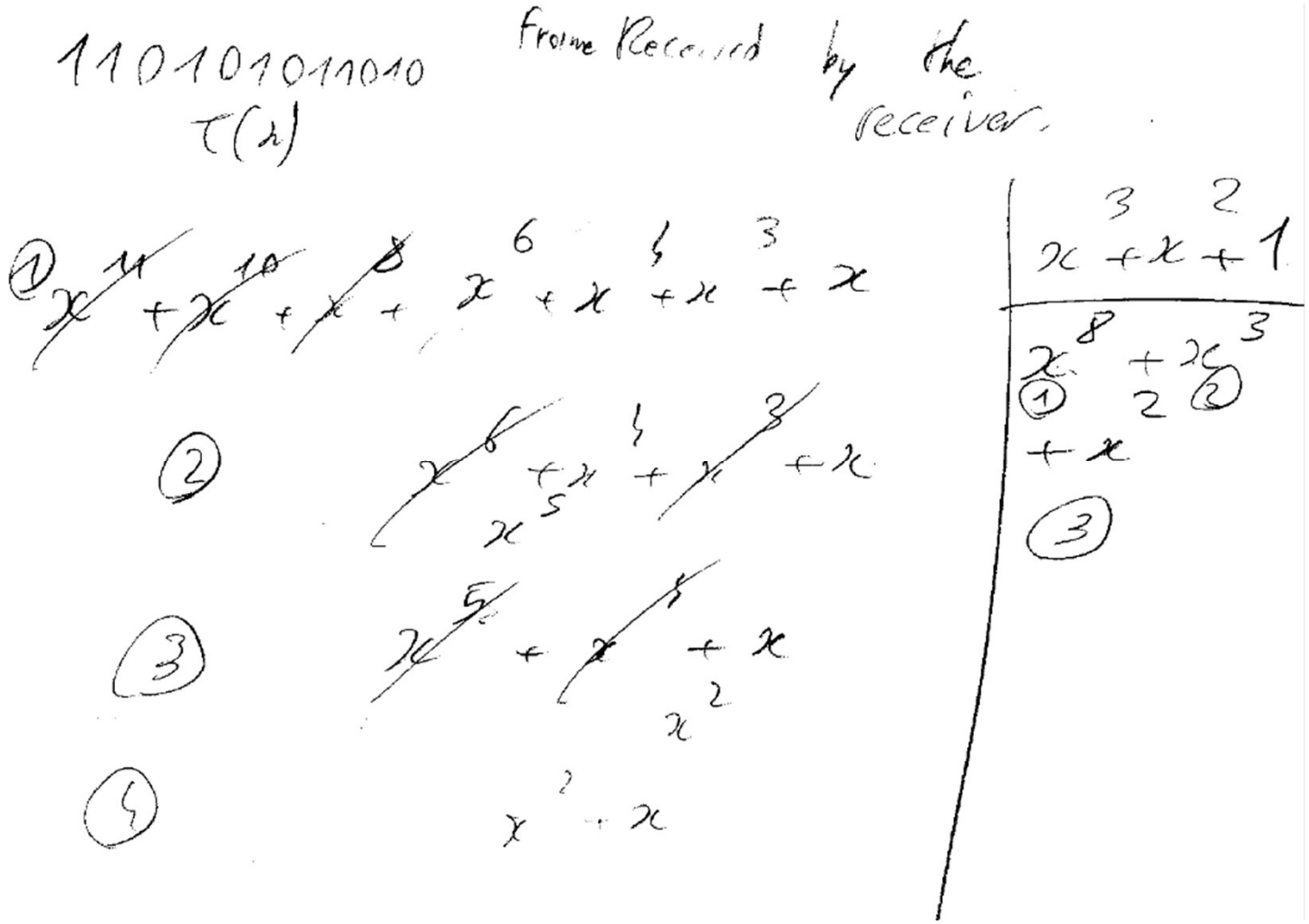
- 1.2 A receiver receives (odd parity) this message
- 1 1 0 0 1 1 => not odd
- 1 0 1 1 0 0 => odd
- 0 0 1 0 1 1 => odd
- 0 0 1 0 1 1 => odd
- ERR OK OK OK OK OK
- Are there transmission errors?
- Yes, line 1 and column 1
- If yes, is it possible to correct the transmission errors?
- Yes
- What is the corrected message ?

Exercise

- 1.2 A receiver receives (odd parity) this message
- **0** 1 0 0 1 1
- 1 0 1 1 0 0
- 0 0 1 0 1 1
- 0 0 1 0 1 1
- Are there transmission errors?
- Yes, line 1 and column 1
- If yes, is it possible to correct the transmission errors?
- Yes
- What is the corrected message ?

Exercises

- 2. We receive the following message: 110101011010 using the following generator suite: 1101.
- Explain how the frame is built: which bits are representative of the data and which of the CRC?
- **We know that $T(x) = x^r.M(x) + R(x)$**
- **We know that $G(x) = x^3 + x^2 + 1$, degree 3**
  - **So the highest possible degree of the remainder is 3 - 1 = 2**
  - **OR so the size of the remainder is 3.**
- **Consequently 110101011 are the data and 010 is the remainder.**
- Is the message correctly received?
- We have to divide **$x^{11} + x^{10} + x^8 + x^6 + x^4 + x^3 + x$ / $x^3 + x^2 + 1$**
- **The result is $x^2 + x$ which is <> 0, so the frame has not well been received.**
- If not, what can/should we do?
- Ask the transmitter to send the message again


1101010011010
T(x)
Frame Received by the receiver.
x^3 + x^2 + 1