

IG CS Topic 2.5-2.6 Error Detection and Encryption

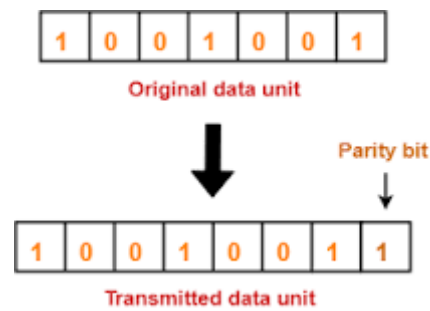
Created by HardyWen

Error Detection

- **The need for error detection**
 - **errors** can occur during data transmission due to **interferences**
 - e.g. data loss, data gain, and data change
 - for example, if an email address is wrongly transmitted, the email might not reach the correct destination
 - hence, the accuracy of data is **vital**, so we need to **detect errors** to make sure the accuracy
- the error detection methods we'll cover include
 - Error detection in **data transmission**
 1. Parity check
 2. Checksum
 3. Echo check
 - 4. Automatic Repeat Request (ARQ)
 - used to establish that data is received without error
 - 5. Check digit
 - used to detect errors in data entry

Parity Check

- a type of error detection method that **adds an additional bit to each byte to create an odd or even sum.**
- **Process**
 1. Check whether it's **odd parity** or **even parity**
 2. Calculate the number of "1"s in 7-bit of data before transmitted
 3. There will be a **parity bit** which acts as the 8th bit of the data so that you can control the parity of all your data
 - if you use an odd parity check, the number of "1"s in all your 8 bits should be odd
 - vice versa for an even parity check
 4. Hence, when the data is obtained, if the sum of "1"s does not obey the parity defined before the transmission, there is an error
 - 通过parity bit控制传过去data的奇偶性，如果接受到的data的1的数量奇偶是错的，就是错的
- consider the scenario below, the situation uses an even parity check so before data transmission, a "1" should be added to the parity bit to make sure that the number of 1s in the transmitted data unit is an even number.



- you should also know about the parity block, which is as below (it uses even parity)
 - the 1st and 2nd columns are wrong because they have odd parities
 - all the other columns and rows are from even parities

	Parity error in 1 st and 2 nd column				Parity bit for each row			
	↓	↓				↓		
Received data	1	0	1	1	0	1	0	1
	0	1	0	0	1	0	0	1
	1	0	0	1	0	1	1	1
	0	0	0	0	1	1	0	0
Parity bit for each column	1	0	1	0	0	1	1	1

Checksum

- a type of error detection method that **performs a calculation on the data to create a checksum value**. Checksum values are **compared after transmission to see if they match**.
- **Process**
 1. before the transmission, a checksum value is calculated from the data using a specific algorithm
 - such as a modulus of 11
 2. the calculated value will be added to the data to be transmitted with it
 3. after the transmission, the receiving device uses the same algorithm to calculate a checksum value from the received data
 4. if the value matches, then there is no error
 5. else, there is an error
 - 算值传过去，接收者同样算法算值，比较值

Echo Check

- a type of data detection method that **sends a copy of the transmitted data back to the sender to be compared with the original data sent**
- **Process**
 1. the sender transmits the data to the receiving device
 2. the receiver then transmits the data it receives back to the sender
 3. the sender compares the data it has sent and the data it has received
 4. if they match, then there is no error
 5. else, there is an error

- 传过去传回来比较

Check Digit

- A type of error detection method that is **used for data entry. A calculation is performed on the data entered to create a value**. Check digit values are **compared** to see if the data entered is correct.
 - data can be wrongly inputted upon entry, such as a barcode scanner might obtain a wrong barcode
- **Process**
 1. A check digit value is previously calculated from the data that will be entered at some point. This value is stored with the data
 2. When the data is entered, the check digit is recalculated from the data entered
 3. If the previously calculated check digit and the stored one match, then there is no error
 4. else, there is an error
 - It works just like checksum but is used in data entry...
 - 预先算好check digit值和数据存储，当数据再一次被输入时再算一遍check digit并和之前的比较
- **Examples**
 - Barcodes
 - International Standard Book Numbers (ISBN)
 - e.g. the ISBN check digit is calculated for every book and stored with the book. When the book's information is inputted, an ISBN check digit will be calculated again to see if it is matched the stored one

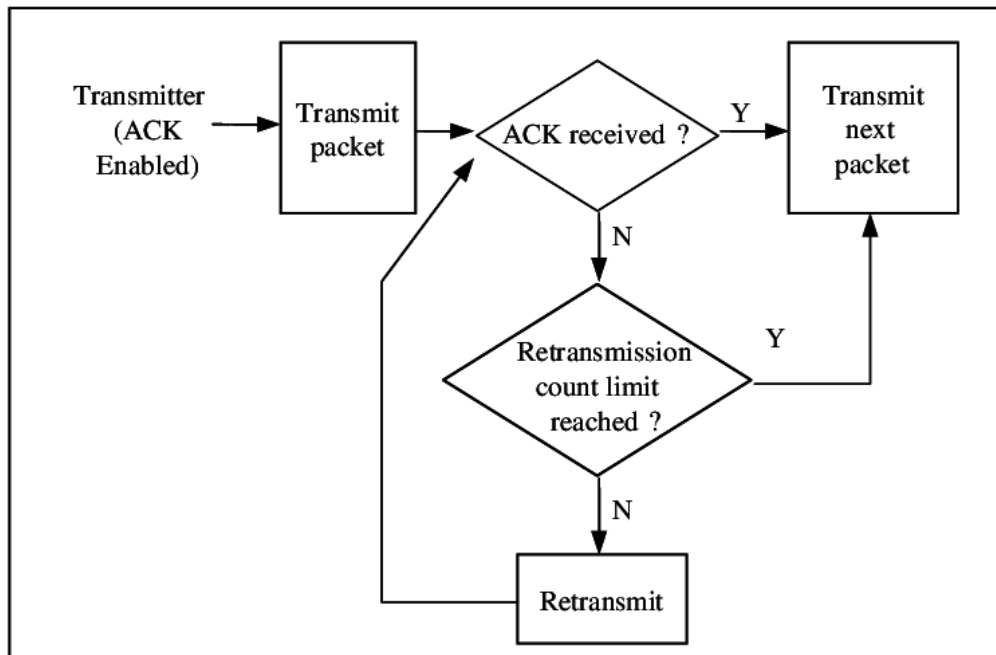
Automatic Repeat Query

- A type of error detection method that uses acknowledgement and **timeout** to **see if data has arrived correctly after transmission**
- ARQ is used to ensure that when an error is detected in the data, either the sending device will send the data again, or the receiving device will tell the sending device to send the data again
- there are both **positive** and **negative** ARQ methods

Positive ARQ

1. the sender transmits the first data packet
 2. the receiver receives the data and checks it for errors
 3. if the data has no errors, the receiver sends a **positive acknowledgement** back to the sender
 - when the sender receives this positive acknowledgement, it sends the next package (as it knows that the previous data is correctly sent)
 4. if the sender **does not** receive a positive acknowledgement within a set timeframe, a **timeout** occurs
 - when the timeout occurs, the sender will resend the data packet
 - it will keep resending until it receives a positive acknowledgement or a time limit (such as 20 times) is reached
- To conclude in Chinese

- 发包 -> 接受者检查无误发positive acknowledgement -> 发下一个包
- 发包 -> 有误 -> 发送者在设定时间内没收到positive acknowledgement -> 一直重新发同样的包直到它收到了acknowledgement或者超过了重新发送设定次数



Negative ARQ

1. the sender transmits the first data packet
 2. the receiver receives the data and checks it for errors
 3. if the data has no errors, no further actions are taken
 - if the sender receives no response within the timeout, it will send the next packet
 4. if the data has errors, the receiver sends a **negative acknowledgement** to the sender
 - the sender will resend the data seeing a negative acknowledgement
- To conclude in Chinese
 - 发包 -> 接收者检查无误则不动 -> 发送者很久没收到回复就发下一个包
 - 发包 -> 有误 -> 接收者发送negative acknowledgement -> 发送者重发
 - Can't find a flowchart for this one and I'm too tired to produce one (: ... Sorry :(

Encryption

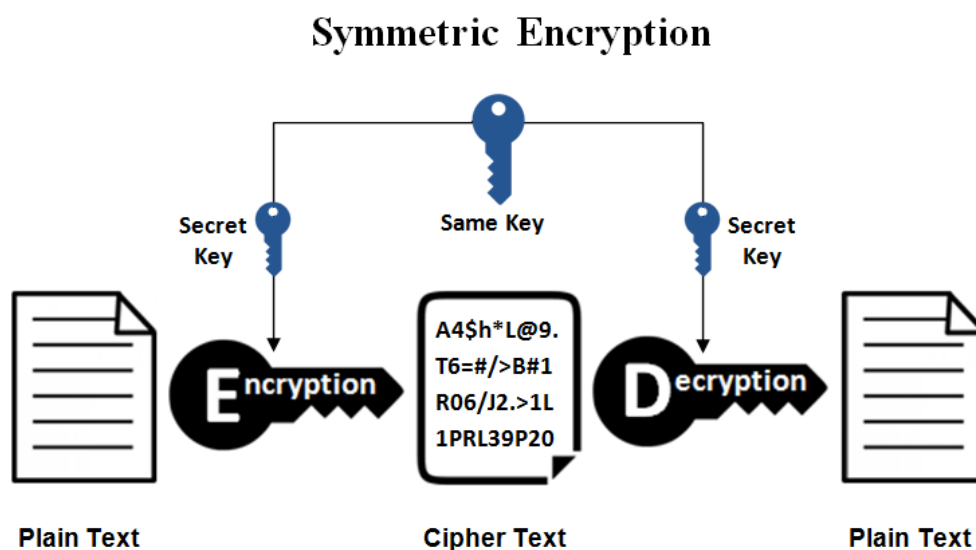
- a method of securing data for storage or transmission that scrambles it and makes it meaningless
- **The need for encryption**
 - data that is transmitted often needs to be protected during transmission
 - because data is valuable (imagine your bank account and password)
 - so, encryption protects the data from being accessed to and understood by a 3rd-party (such as a hacker) when it is being transmitted

Keywords

- **Plain text**
 - the name given to the data before encryption
 - e.g. "I love her".
- **Encryption key**
 - a type of algorithm that is used to encrypt data
- **Cipher text**
 - the name given to data after encryption
 - e.g. "fjosaifjoasifjoi"
- The plain text turns to cipher text through the **encryption key**, and so does the cipher text to plain text

Symmetric data encryption

- a type of encryption that uses the **same key** to encrypt and decrypt data
- **Process**
 1. Plain text is encrypted into cipher text using an encryption key
 2. The cipher text and the encryption key are sent **separately** to the receiving device
 3. The same key is then used to decrypt the cipher text back into its plain text form

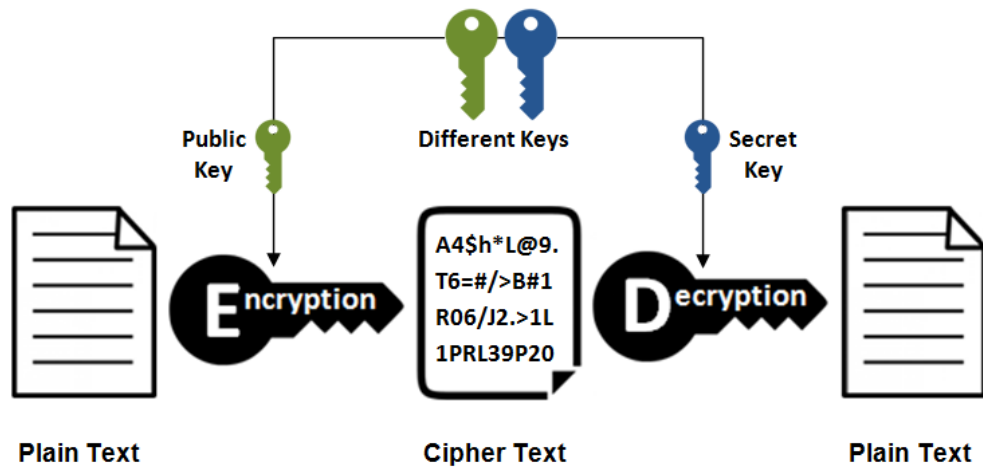


- However, this method would be useless if the hacker also gets the encryption key when it is being transmitted through the network.

Asymmetric data encryption

- a type of encryption that uses **two different keys** (public and private) to encrypt and decrypt data
- **Process**
 1. Plain text is encrypted into cipher text using a **public key**
 2. The cipher text is transmitted to the receiving device
 3. The cipher text cannot be decrypted using the **public key**, it can only be decrypted using a **private key**

Asymmetric Encryption



- The public key is made public to everyone. So if anyone wants to send you a message, it can gain the public and encrypt the data
- The private key is only authorized to you, and only you can decrypt the data that is encrypted using your public key, through your private key.