TEXTBOOKS / REFERENCE :-

i. Information Security Principles & Practice by Mark Stamp 2nd Edition.

ii. Counter Hack by Skedic

* Virus signature are nowadays polymorphic. They change their code as use be.

* The following are some pre-reqs are, CAO, MPBS, OS, CCN ze Internet Computing.

* Do not equate internet security with n/w & security.

∴ CSS = Endpoint Security + Internet Security + Web Security + data static — Security + OS Security + Cloud transit. Security + Mobile Security + N/w Security + Db Security.

→

* Static data could be stored in.

cloud or database.

* with data there is one more security that is ownership security, it is dealt with DRM.

* Search how to remove watermark.

* How does n/w engineering affects Moving Target Defense.

* Use the Port Scanning on Windows & Linux.

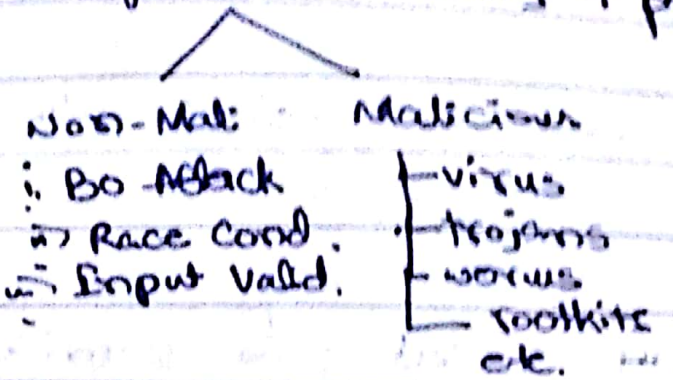
* Search reconnainnance.

* XSS
  → how to block it.
    ↳ at client side
    ↳ at server side.

* Our labs are SEED Labs.
  This is a virtual machine.

* Go to SEED Labs website.

* + email security + Spyware security
+ Software security + privacy.



* Now in a non-malicious software, the intent of the programmer, isn't malicious but due to errors the software becomes vulnerable.

* Now the first occurance of BO Attack was in IIS v4 which was accepting lengthy URLs & getting crashed.

* Now after ~~got~~ crashing, it allowed for shell code execution.

* Security bulletins to look.
  + ubuntu.com/usn.
  + Microsoft Security Bulletin.

* Look for one issue & prepare.

* We are gonna look ~~for b~~ into bits & bytes!

* Facebook shares ~~or~~ cookies with other website.

* ~~Social Security~~ Social n/w security comes under privacy.

* Now there are 2 major paradigm of security,
  - Security by barrier
  - Security by design.

* Throughout the world, there is a trend of implementing security by barrier.

* Also,

  Web Security
  Email Security
  Cloud Security

  — https ✗

* The purpose of https is to ensure secure comm. b/w server

2 web browser.

* Furthermore, there is a common misunderstanding with end-to-end security.

* A secure email is as.
  " Irrespective of the security of web server, email should be end-to-end encrypted".

* Search "Email Security & How to send email security".

*. Book,

"24 Deadly Sins of Software Security"..

* Gartner is the agency which rates the security solutions.

* Software is a poor foundation in security.

* If your software is subject to attack, your security can be broken.
  - Regardless of strength of crypto, access rights, etc.

* Data security is not the only the use of encryption algo. But mostly it is considered as such.

* AES is considered to the best encryption algorithm & considered as a de-facto industry standard

* Cryptography is not everything.

* Gene Spafford quote

* If you just focus on endpoint security & leave everything else, you are leaving your system vulnerable.

* Just focusing on cryptography is foolish.

* Network Security & Internet Security are completely different things.

* NASA Mars Lander Crash.

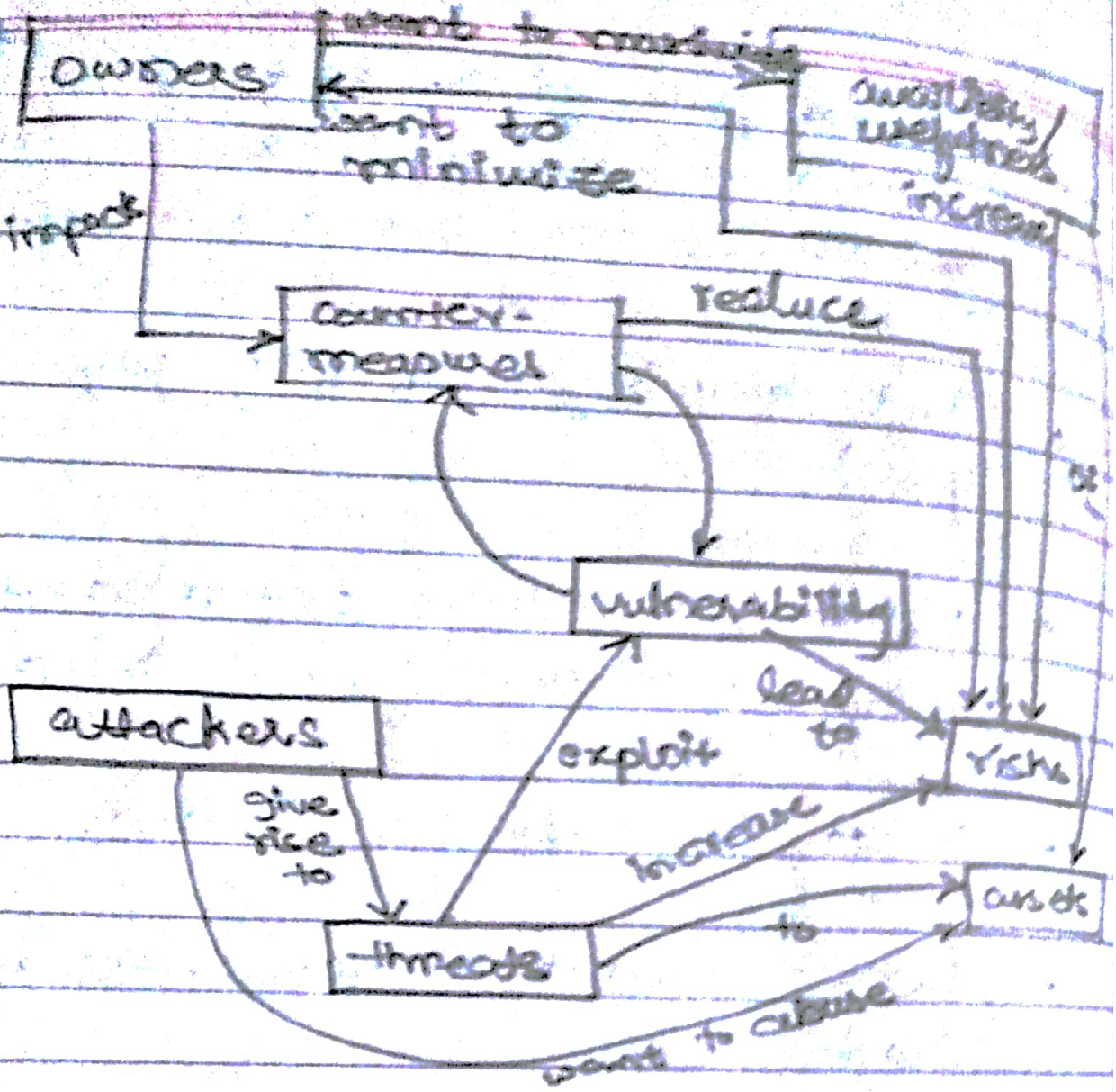
* Denver Airport Baggage Handling Crash.

* MV-22 Osprey fault.

* Security & functionality features are always contradictory.

* SSL & TLS go for more deeper than HTTPS.

* CA is Certification Authority. It offers a public key to enable HTTPS



* If a CA server goes down all the sites having that public key will go down.

* 2 major dangers to availability are.
  - DDoS attackers
  - Malware Attacks.

* Viruses are becoming polymorphic & signatures are becoming obsolete.

* Alice & Bob — Trudy.

* "Complexity is the enemy of security". Paul Kocher.

* Program flaws :-

* An error is a programming mistake.

error —(may lead to)→ fault —(may lead to)→ failure.

* Secure software requires software does only what is intended & nothing else.

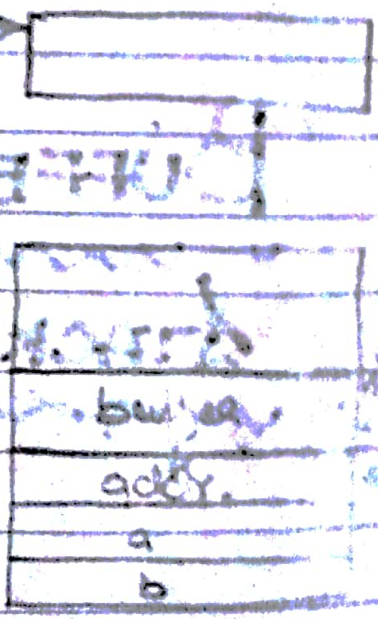
BUFFER OVERFLOW / SMASHING ATTACKS :-

```
int main()
{
    int buffer[10];
    buffer[20] = 37;
}
```

* 3 possibilities are these :-

* It might ~~over~~ overwrite user data.
* It might overwrite system data.
* Program might run just fine.
* Stack always grows from higher to lower.
* Return address is always at the top of the stack.

```
void func (int a, int b) {
    char buffer[10];
}
```




```
void main() {
    func(1, 2);
}
```

RA - Return address

* Now if we give more i/p greater than size of buffer.

Q. What happens if buffer overflows?
- Program "returns" to wrong location.
- Crash is likely.

* One possibility is that we blindly overwrite the buffer & the pointer goes into the space which is not assigned to its process, then OS will catch it.

* Now, if we jump within the process space, there are 2 possibilities
- Jump into data area.
- Jump into code area.

* If we jump into code area, then the opcode will be the opcode of some instruction & the program will execute but could given some unintended/unpredictable o/p.

* But if we jump to a data element & its opcode doesn't matches any instruction, then the program will crash.

* Now the reason for this is that the address of the next instruction would be carried in Instruction Pointer. which expects an instruction. If it carries an address of a data element it is an anamalous execution.

* We could run malicious code through this.

* Now it will run as follows:-
  - Overflow the buffer.
  - Overwrite the main code return address to the starting address of our malicious code.

* Now the problem is that how to guess the exact starting address

of our malicious code.

* Now the problem is that how to guess the exact starting address of our malicious code.

* Theoretically its impossible to guess. So we prepare a type of landing pad, consisting of NOP operations depending on a certain confidence interval we think the starting address would store.

* The reason for having multiple return address (which are same) is that sometimes some people insert PDs & other things, thus, we have multiple return address.

* IT Security Audition

* Perform the IT Audit of mobile, ISP, etc.

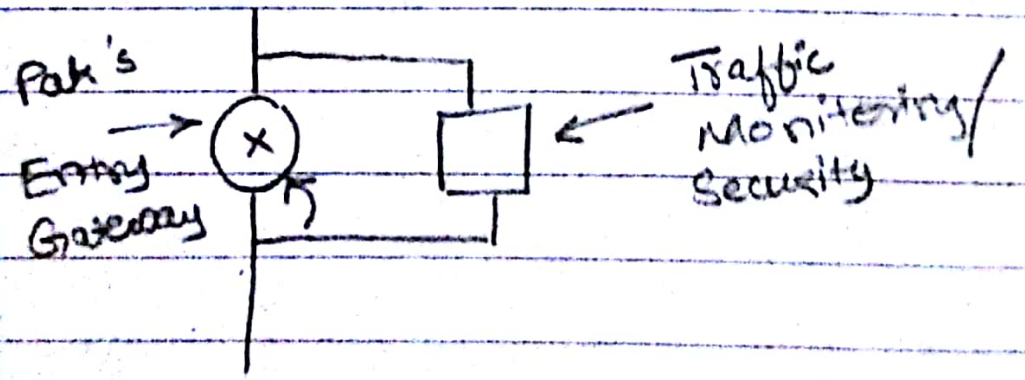
* SIEM. (Security Incident Event Management).

* Look up Arcsight / Fireye / ApacheSpot

* Look up Bro (IDS); An A.I. based IDS.

* The major issue with using A.I. cyber-security solution is the detection of false-positives.

* Passive security gets benefitted by the A.I. based C-S solution.



* If we apply security in main in real-time, then it signifi.

- -cantly affects the throughput. →

* Look up & SIEM, & present next week. Learn how to use it.

→ We train the model seperately & & apply the techniques.

* BO/Stack Smashing Attack cont.

* Malicious code is called shell code.

* There are possibilities of as a result of ~~Smash~~ Stack smashing attack.
  i. The program can crash.
  ii. Execution can be executed.
  iii. Shell code can be executed.

* ~~A~~ A series of NOP becomes a very good signature for identifying BO attack.

* Now what are the solution to this?

* One way is to instead of using NOP instr. use variable instr. that

* don't do actual work (mov ax, ax) just use computer cycles.

* But even this is detectable. So what can even do?

* Now, in order to make attack undetectable, we have to make polymorphic.

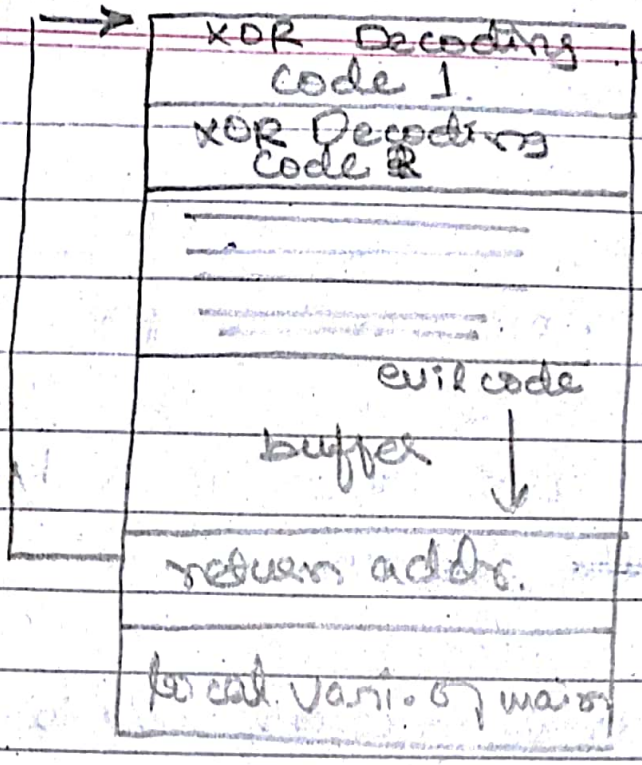
* We are working at opcode.

* Now one way is to use XORing the code.

* 1 0 1 1 0 1 1 1 ← XORing key.
⊕ 0 1 1 0 1 0 1 0 ← ; XORing again gives us back original
---
1 1 0 1 1 1 0 1 ← XORed instr. opcode.

* There is no attack that leaves no trace / trail.

* Now what happen is we use XOR Decoding code.



* The purpose of this code is that it XOR all the address succeeding the last bit of its own code. At runtime that code is changed back to its original form.

* We could attach multiple XOR code segments that XORs even the succeeding XOR function, to make the attack as much polymorphic as possible.

* But once again we are storing at the same place, i.e., how do we guess the first byte of the

of first XOR code.

* The solution to this is to once again use NOP landing pad.

* Thus once again there is a trail. that is ~~data~~ detectable.

* Now we move to word how to prevent this attack at endpoint. even though this is a vulnerability of the developer.

* Revise the stack smashing ~~Attack~~.

* ~~Look at why~~

* The value at the left is called virtual address.

* Look at the byte ordering of different transmission.

* Simulate the example at on Linux & Windows.

* The simple example of BO attack is one of the fundamental concepts of "Reverse Engineering".

* Look up "Reverse Engineering".

* IDAPro (Use the free version) : Static RE.

* This tells the flow diagram of the code.

* OllyDbg : Dynamic RE.

* First we use IDAPro to get an idea of flow diagram & use OllyDbg during runtime to manipulate the flow according to our own desired

* "Computer Security Principle & Practices" 4 Edition by William Stallings.

* Search for PDF.

* In the book example, the case was that we had a comment telling ~~which~~ ~~whos~~ which instruction is the checking instruction.

* Do the BO ~~Attack~~.

1 ~~##~~(while IP == jump_code) {
key = 10110010;
xor_ins = ~~BB~~ XOR (IP, key);
IP = xor_ins;
return.

1. while (n != bytes_of_code)
key = 10110010;
xor_ins = XOR(IP, key);
IP = xor_ins;
n+4;
end.

* See the code for XOR decoding.

How to Defend Against BO Attack :-

* Employ non-executable ~~bi~~ stack.

* Use of ~~seperation~~ seperate bit for each memory address which tells that whether the content on that specific address is executable or not. called Executable Bit / Non-Execute Bit.

* But the ~~down~~ demerit of it is it will significantly increase the memory bandwidth. Plus its an architectural level solution.

* Use a Canary :-

* Canary is a technique/solution by Microsoft to employ ~~ss~~ stack security

* It checks a flag before the return address.

* ASLR:-

* Address Space Layout Randomization

* Now this was implemented in Vista & was the reason for Vista's failure.

* Succeeding versions of Windows didn't have this.

* Now this is implemented in Mac.

* Randomise place where code in memory.

* Makes most buffer overflow probabilistic.

* Now memory is not sequential.

* But there is a demerit to it that the layout could be de-rand

-used, & it could be guessed.

INCOMPLETE MEDIATION :-

* Now this ~~could~~ is mostly related to web forms & web services.

* Security processes should be atomic. If they are layered, then vuln. could arises. They must occur all at once.

* AAA (Triple A)

A - Authentication (who you are)
A - Authorization. (What authorised are you to do).
A - Auditing.

* Now these are Triple A severs for example RADIUS,

* Now an ideal solution to this is to have a single server do all this.

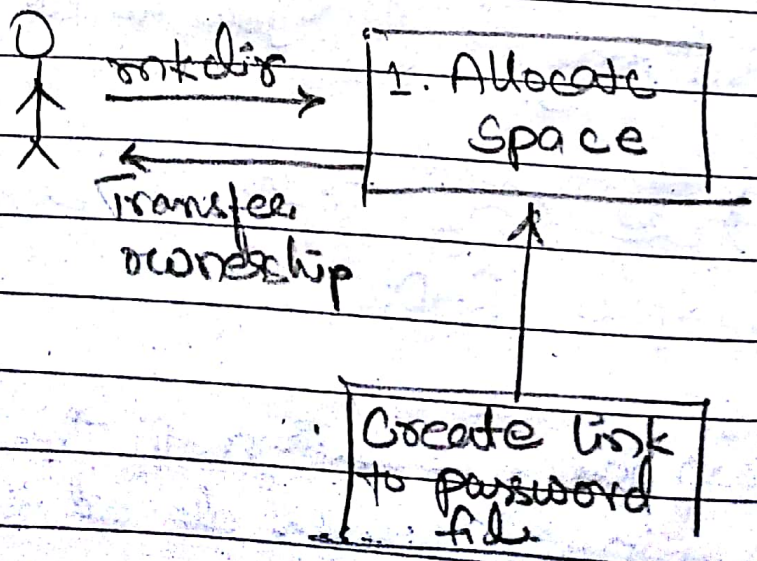
* But these could ~~bee~~ be 2 different servers, on a single network, have done authentication & authorization on different servers. In that case our logical architecture should be as strong, that comm. b/w. 2 servers shouldn't be interfered with

INPUT VALIDATION:-

* Now input validation is a big issue.

* Look deeper how could be input validation.

RACE CONDITION:-



* Now here timing is critical, even down to timing b/w 2 keystrokes.

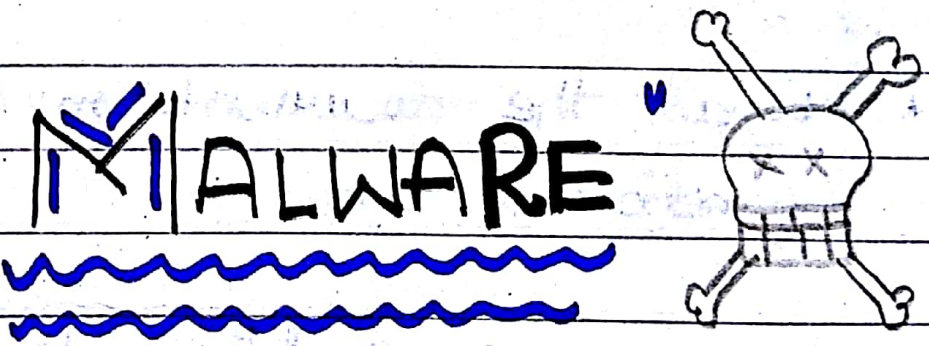
* It was a bug, it was removed.

* Race condition are common.

* Race condition may be prevalent than buffer overflow.

ASSIGNMENT

* Look up USN with the following topics:-
  i. BO ii. Input Validation
  iii. Race condition.



* Malware are software with malicious intent

* Types of Malware:-

- Virus : passive propagation.

- Worm (Write Once Read Many)

* Now even if WORM stands for Write ~~One~~ Once Read Many, it could be written once read one ~~read~~ read few.

* Trojan Horse - unexpected functionali

* Write a program that executes but terminates & stays as in memory after termination.

* Memory - resident programs.

* Read the documentation of keep() function in C++.

* The purpose of this is that whenever a program uses this function the process with which the program is associated with, it doesn't terminates.

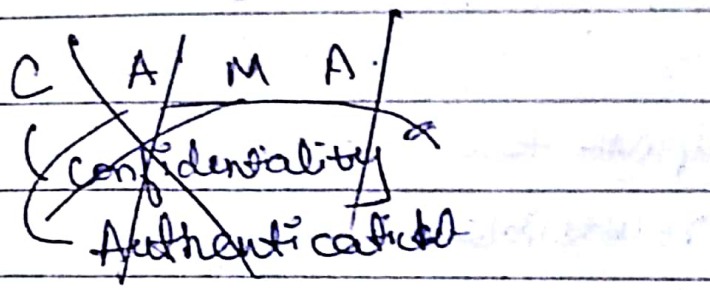
* Look up TSR.

* Point to point functions

* Show the code of this.

* Now if we are stealing the interrupt & we implement it as a memory resident program, the only solution to this, is to reboot the system.

* How to implement a malware in a pdf.



C — Confidentiality
A — Authentication
M — Message Integrity
A — Access & availability.

* Symmetric key - encr. decry. keys identical.
* Public key - encr. key public, decry. key pvt.

* Popular public key algos, AES, Blowfish, 3DES, etc.

* Both the parties come to an agreement on the key-value through a key agreement protocol.

* DES
  - 56 bit symmetric key
  - 64 bit plaintext input

* BLOCK CIPHER:-
  - ~~Must~~ Takes plaintext & a key as i/p.
  - Outputs cipher text.
  - Must be reversible (i.e, plaintext could be recoverable).

* Addition of a key ensures security.

* The more value a key can have, the more secure a block cipher will be.

* A block cipher is just a function that takes in a plaintext message & a key & o/ps cipher text.

* Not every function can be a block cipher.

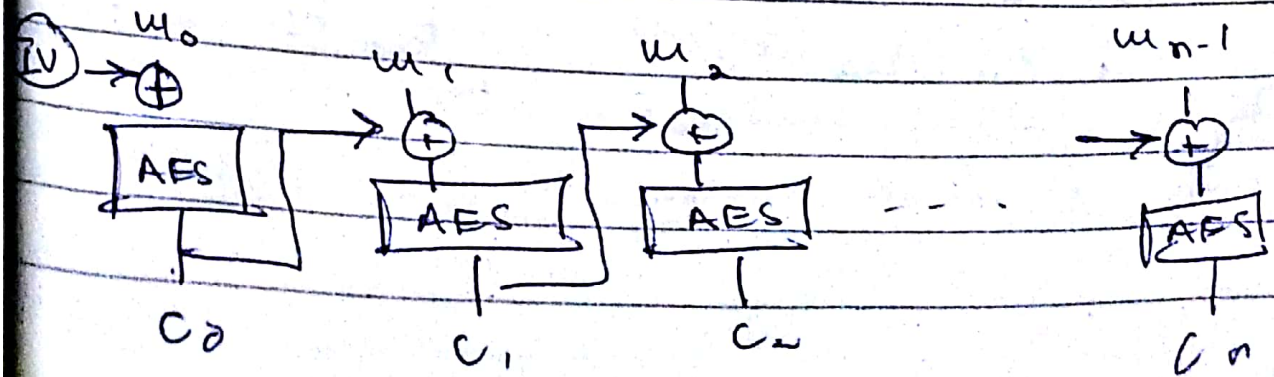
* 2 rules must follow.
  a. The function must be a permutation, i.e, it must have an inverse
  b. The function must be efficiently computable.

* AES & DES use block cipher.

CIPHER BLOCK CHAINING:

* Use cipher text from the previous block to ~~the next block~~ influence the next block.

$m = m_0\ m_1\ m_2 \cdots m_{n-1}$



AES-

* Block size is 128-bits.

* Key length is → 128 bits
  → 192 bits
  → 256 bit
  } Now rounds = 10-14.

* Each round has four functions.
  i. Byte Sub
  ii. Shift Row..
  iii. Mix column.
  iv. Add Round. Key

* Treat 128-bits as 4 × 4 byte array.

Step 1 (Byte Sub)

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{\text{Byte Sub.}} \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Step (2):

	0	1	2	3	...	c	...	f
0						⋮		
1								
2								
3	- - - - - - - - - - - - - - - - eb.							
⋮								
f								

byte sub(8c) = eb.

Step (3): (Shift row)

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{13} & a_{10} & a_{11} & a_{12} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{31} & a_{32} & a_{33} & a_{30} \end{bmatrix}$$

Look from notes.

Step (3): (Mix Col.)

$$\begin{bmatrix} a_{0i} \\ a_{1i} \\ a_{2i} \\ a_{3i} \end{bmatrix} \xrightarrow{\text{Mix. Col}} \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \\ b_{3i} \end{bmatrix}$$

Round

Step (4): (Add Round key).

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} K_{01} & K_{02} & K_{02} & K_{03} \\ K_{10} & K_{11} & K_{12} & K_{13} \\ K_{20} & K_{21} & K_{22} & K_{23} \\ K_{30} & K_{31} & K_{32} & K_{33} \end{bmatrix}$$

* In industry data sec. is considered to be AES / HTTPS. This conception is wrong.
* we must think whether we are talking about static data sec. or transit data sec.
* There is a difference b/w block ciphers & stream ciphers.
* A5/1 —— Page # 53 (obsolete)

  RC4 —— Page # 55
* Read A5/1
* If the data is less than 128-bits

If not we have to ~~pad~~ using a padding.

* Standards are
  128-bit key —— 10 rounds.
  192- " " —— 12 rounds
  256- " " —— 14 rounds
  Though these are tweakable

* Byte Sub is an operator. It has different operation in diff. txt. And it is static (doesn't changes for different rounds).

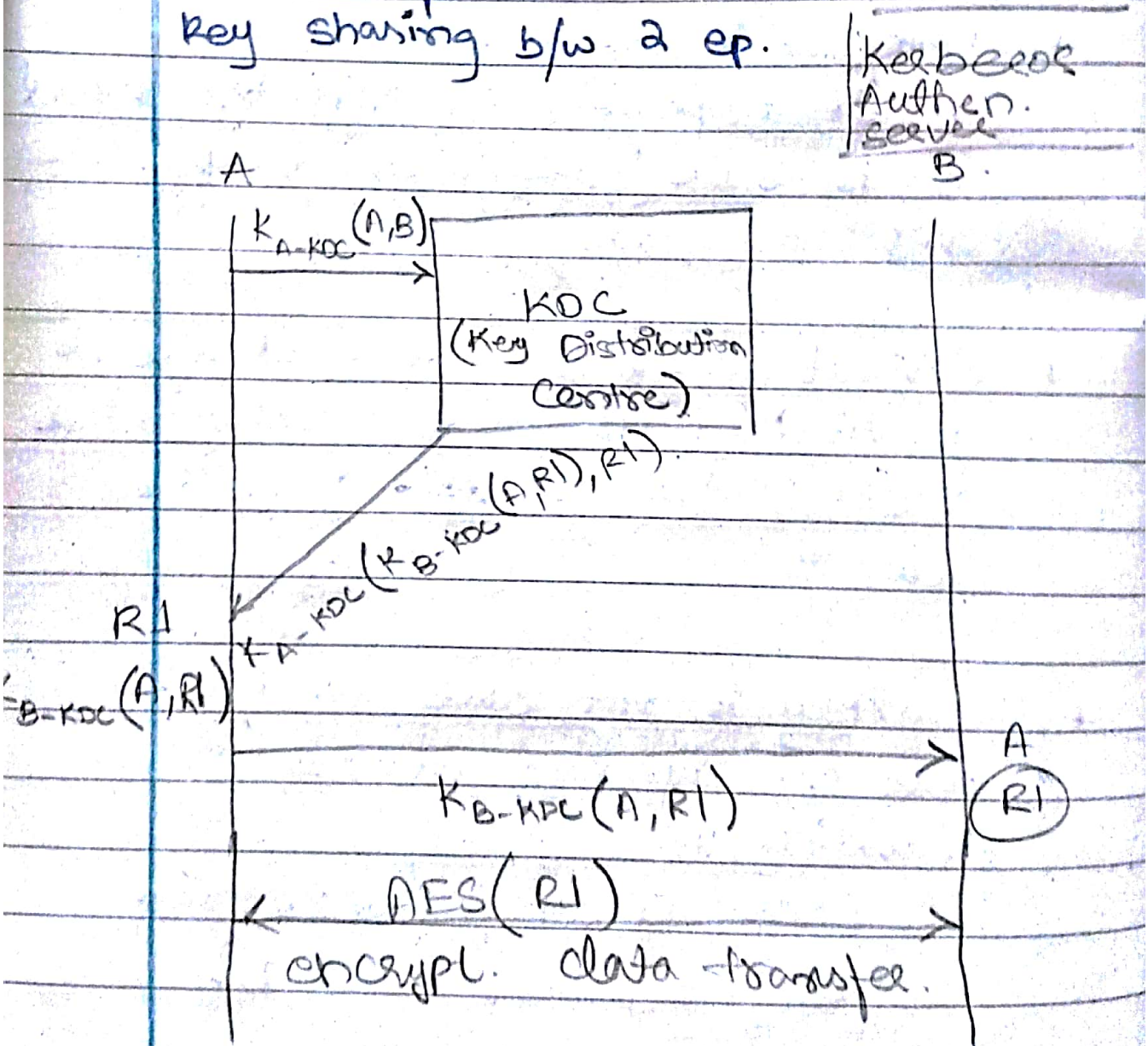
* Easiest ways to implement bytesub & MixCol is to devise a lookup table & substitutes the ~~repet~~ respective elements.

* RoundKey is not same as I/p key. There are different methods of generating it.
  LOOK IT UP ! (Next class assgn.)

* Now after getting through rounds, the ciper-text for that blk. & move on to the next blk. in data.

* Now, ~~it~~ to counter the problem of block repetition is we use cipher block chaining.

* Now the problem ~~is~~ arises in key sharing b/w 2 ep.



* Now this whole is implemented on ~~Le~~ LAN. This methodology shouldn't be implemented on public Internet.

* See that in order to encrypt msgs b/w A & KDC & B-KDC needs also to be communicated, & for that we also need to encrypt them. →

* It is called Kerberos Authentication Server.

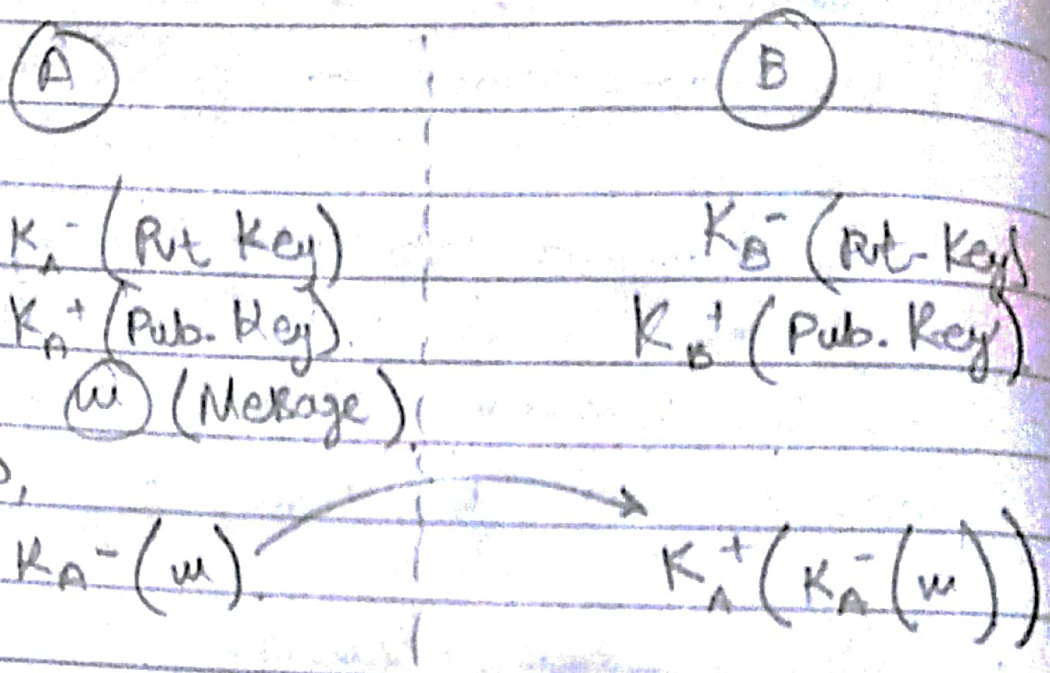
* → So, what we do for sec. is bind the K_{A-KDC} & K_{B-KDC} to a username & pw so that they aren't compromised until someone has an access to a username / pass.

* Now the questions arises if we have to implement this on public ~~key~~ internet

* Think about it! (Although this is not recommended.)

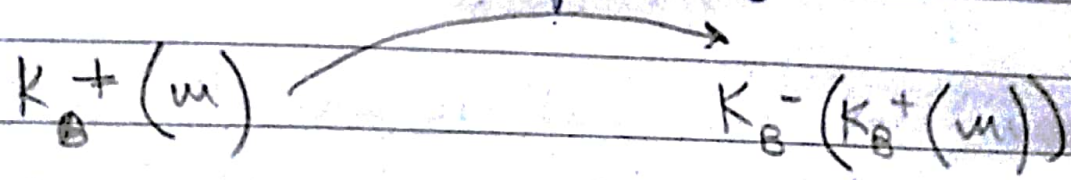
PUBLIC - KEY CRYP.

Consider the following scenario.



* Now this is useless for transfer since K_A^+ is public. & if the mesg is sniffed by someone, he/she could decrypt using K_A^+ & get the mesg.

* Now consider the following.



* Now this could only be decrypted by the recipient.

* The property is,

$$K_B^+(K_B^-(m)) = K_B^-(K_B^+(m)) = m$$

* But there exists a big problem.

* Since the pub. key needs to be known to all, ~~it need~~ it could be vulnerable to modification by unautho-rized authority & could lead to useless messages.

* Thus, to solve this, there comes a party that authorizes a exchange of public keys, known as Certification Authority (CA).

* These ~~certificates authorize that~~ the public key is from ~~author~~ authentic source.

* Some example of CA are DigiCert, Verisign, ThawTE, etc. But these ~~for~~ are all paid.

* Last year, an open source solution by the name of Let's Encrypt.

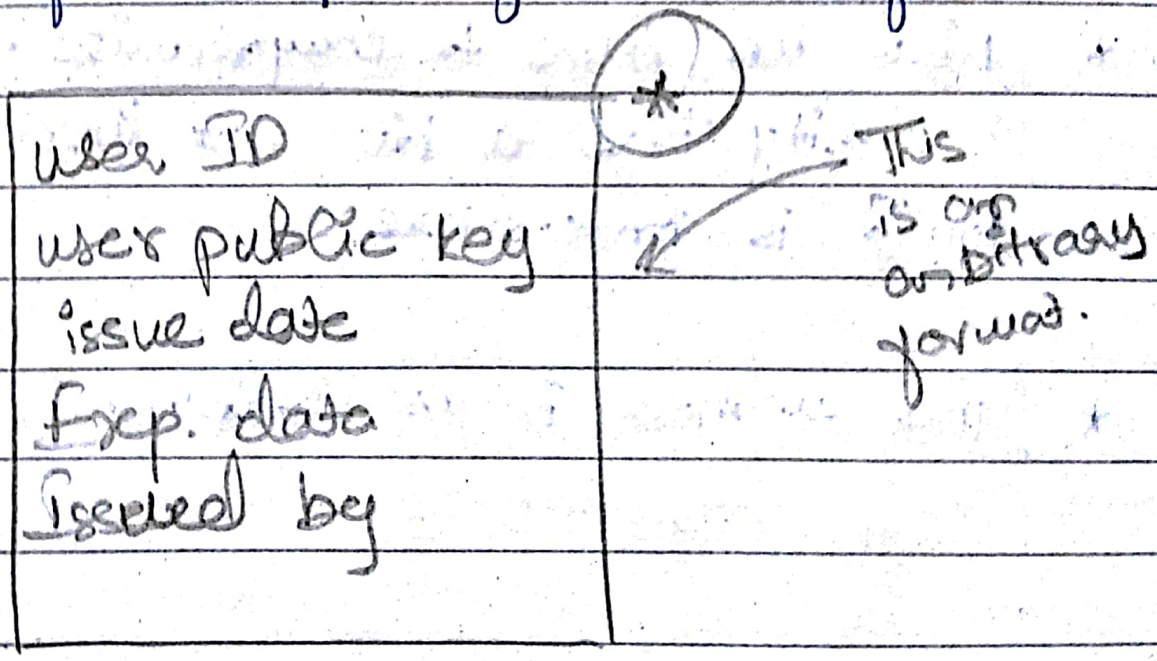
Assignments.

- Read about round key generation.
- Read about CA.
- Read about Lets Encrypt.
- Read about business of CA.
- Implement Secure Key Exchange for Public ~~key on Public~~ Internet. for AES.
- Read about RSA / Deffie Hellman.

CERTIFICATION AUTHORITY :-

* The purpose of the certificate is to bind the public key & the user bind.

* The format of certificate is as follows.



* Certificate don't have ~~the~~ a standard format.

* Now the certificate is itself encrypted using public key crypto.

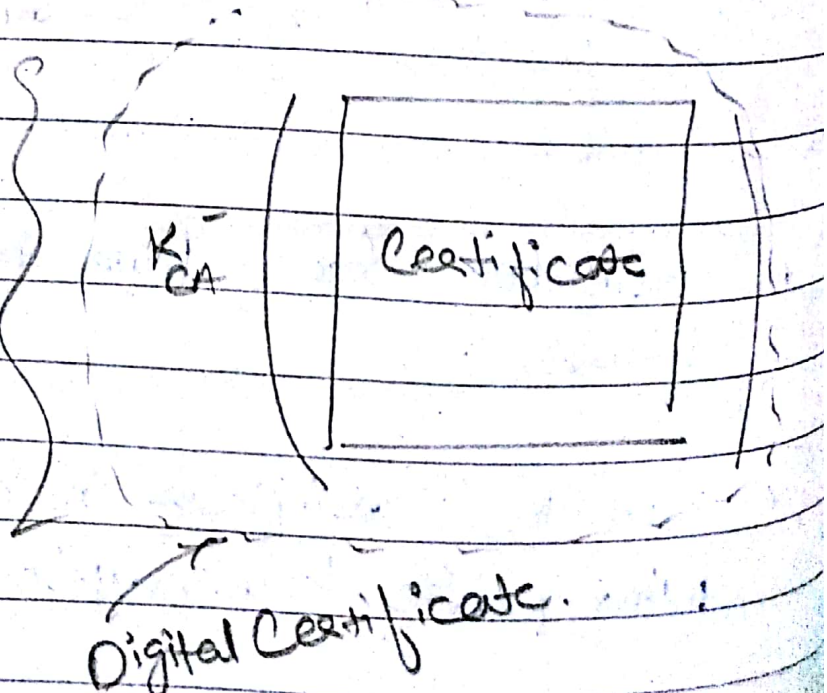
* And we need the public key of CA to decrypt it.

* Now the operation is to decrypt the certificates & to transfer the public key of it ~~with~~ to the browser.

* The method of that is to give the public key of CA to the browser org. & while installation / updating of the browser, it gets installed.

* Now we have to compromise the security here a bit, but then again 100% is impossible.

* The method is as follows,

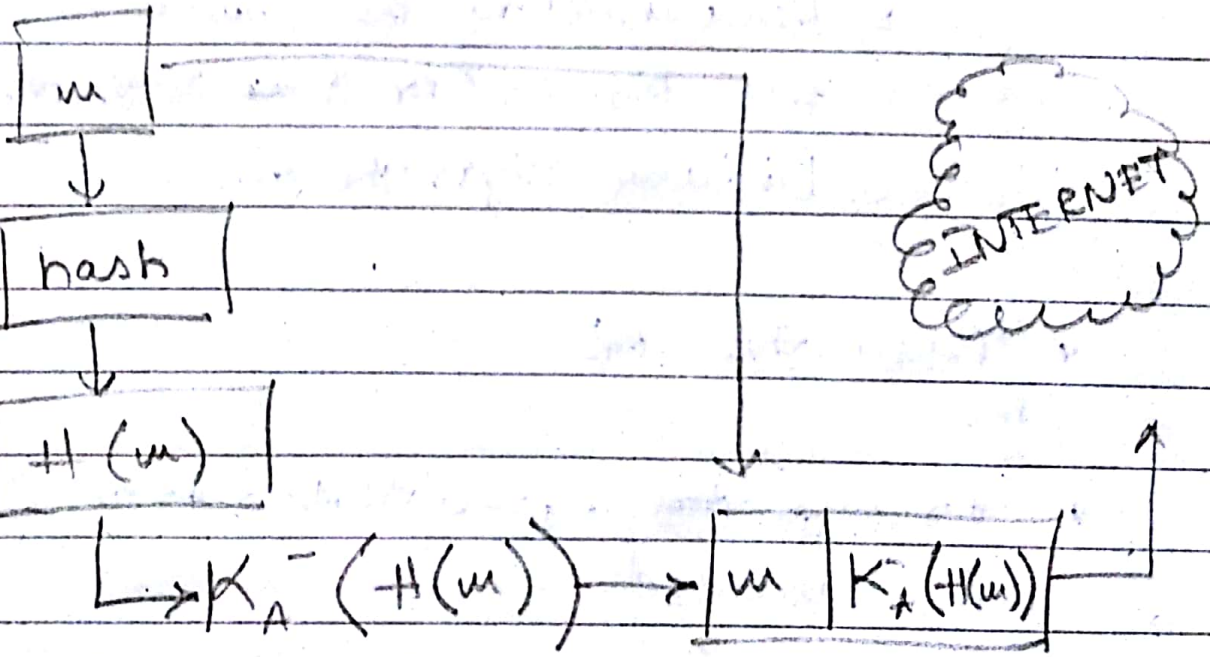


* The publice keys stored in the browser are actually public keys of CAs.

* Now this is a whole business.

DIGITAL SIGNATURE:-

* Now digital signatures ~~are the~~ carry the same value as a physical signatures.

* 

* Now why did we encrypt it using the ~~public~~ private key of sender?

* Because otherwise there is no other way of confirming the identity of

the sender.

* Now the public key of the sendie is given by ~~cer~~ certificate.

* Assignment.

Achieve end-to-end security using.

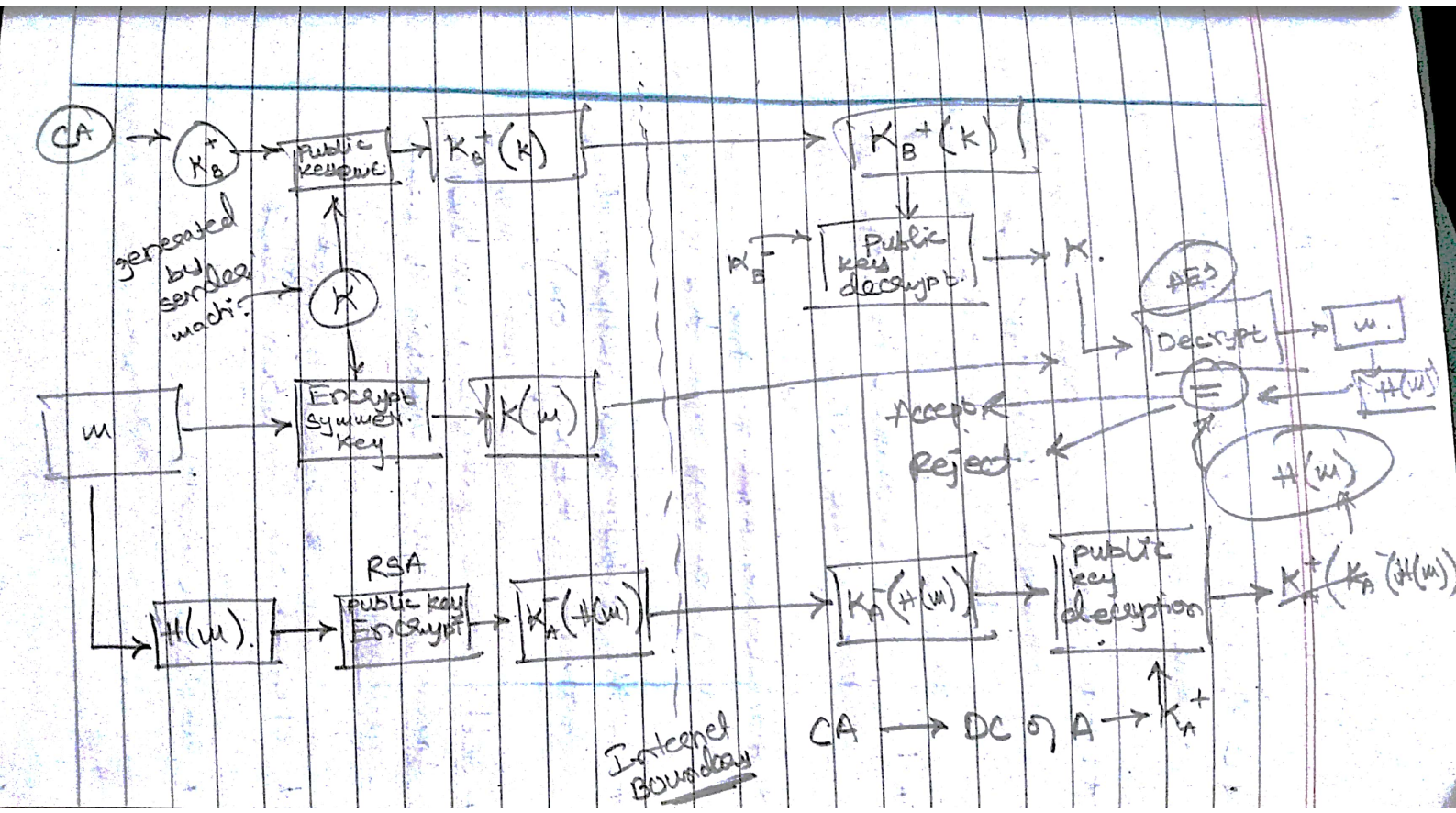
a. Encrypting the message of AES.

b. Communicating the symmetric key b/w sender & ~~re~~ receiver.

c. Digitally sign the file.

* Read the SSL

* Prepare ~~the~~ a presentation ~~of~~ on SSL & Heartbleed.



SECURITY CERTIFICATES

* Certificates are communicated over SSL protocal.

* Even signed certificate have 2 types
  - Fully Authenticate. (certified by CA)
  - Domain Authenticate./ validated.

* Digicert & Symantec have the largest market share.

* For LetsEncrypt.org, there is a requirement of domain name that is registered

* ACME - Automated Certificate Mgmt. Authority. (LetsEncrypt.org).

* Self-signed certificate is just like taking a test & checking it yourself.

* openssl req . — abc.csr ← certificate signing request

* abc.crt ← certificate

DRM :-

* DRM is Digital Rights Management.

* If someone can spoof the email. they could change the signature.

* One example is using a encrypted txt file, sent through an email, which contains a license file, which will just be possible to run on a singular device to ensure good DRM.

* But it is not always possible.

* Sometimes, some people equate. DRM & crypto. But they are completely different.

* There is an over-reliance on cryptography.

* # DRM LIMITATIONS :-

* There is something called "Analog Hole".

* Content protected under DRM. can be easily stolen through analog means (taking a picture of some screenshot-locked app).

* When content is rendered, it can be captured in analog form.

* DRM is not a specifically a technical problem.

SOFTWARE BASED DRM:-

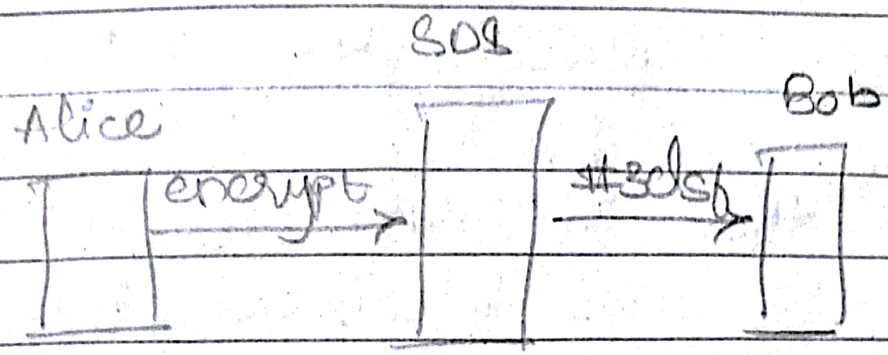
* Strong software based DRM is impossible.

* SRE can break a majority of software & violate DRs.

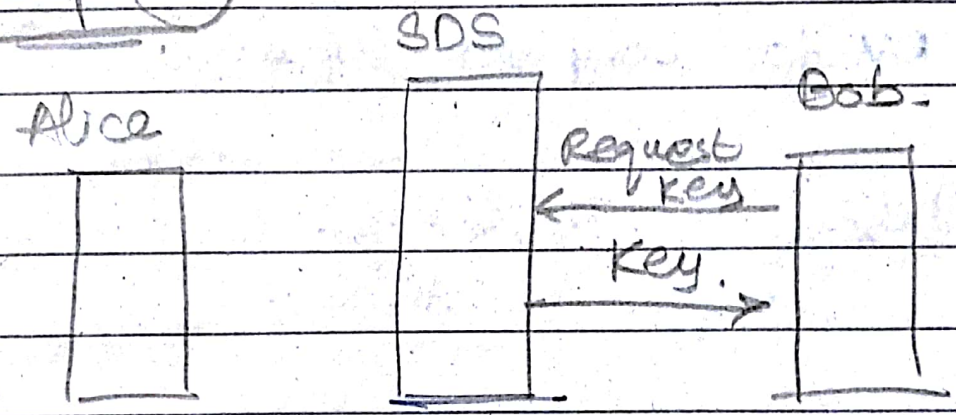
* Example 1

DRM for PDF, Doc.

2 parts of the sys.
- Server (a secure ~~sf~~ server)
- Client.



Step (2)



Step (2).

* Bob authenticates to SDS.
* Bob requests key from SDS.
* Bob can access the doc, but only through special software. (which could only be accessed
* Now we haven't ~~discu~~ discussed

where are we applying this (on LAN/WAN).

* But we already know that.

* But then in solving one prb: we gave rise to several other problems.

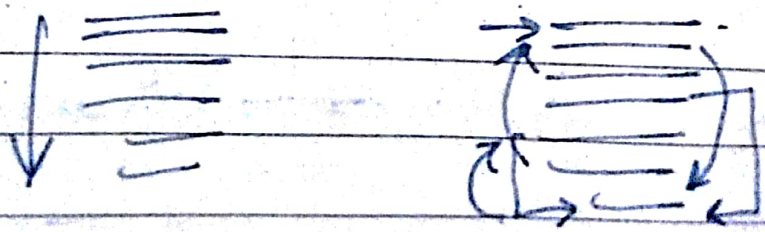
* Now we move towards ensuring DRM for computer software.

* There are 2 things.
  - Tamper Resistence.

Anti-Debugger ⟷ Encrypted code.

* Encrypted code will prevent static analysis of PDF plugin software.

* For example consider a code flow,



* Now we included deliberate jumps to make the reverse engg. debugging

* Although its still possible, but its difficult.

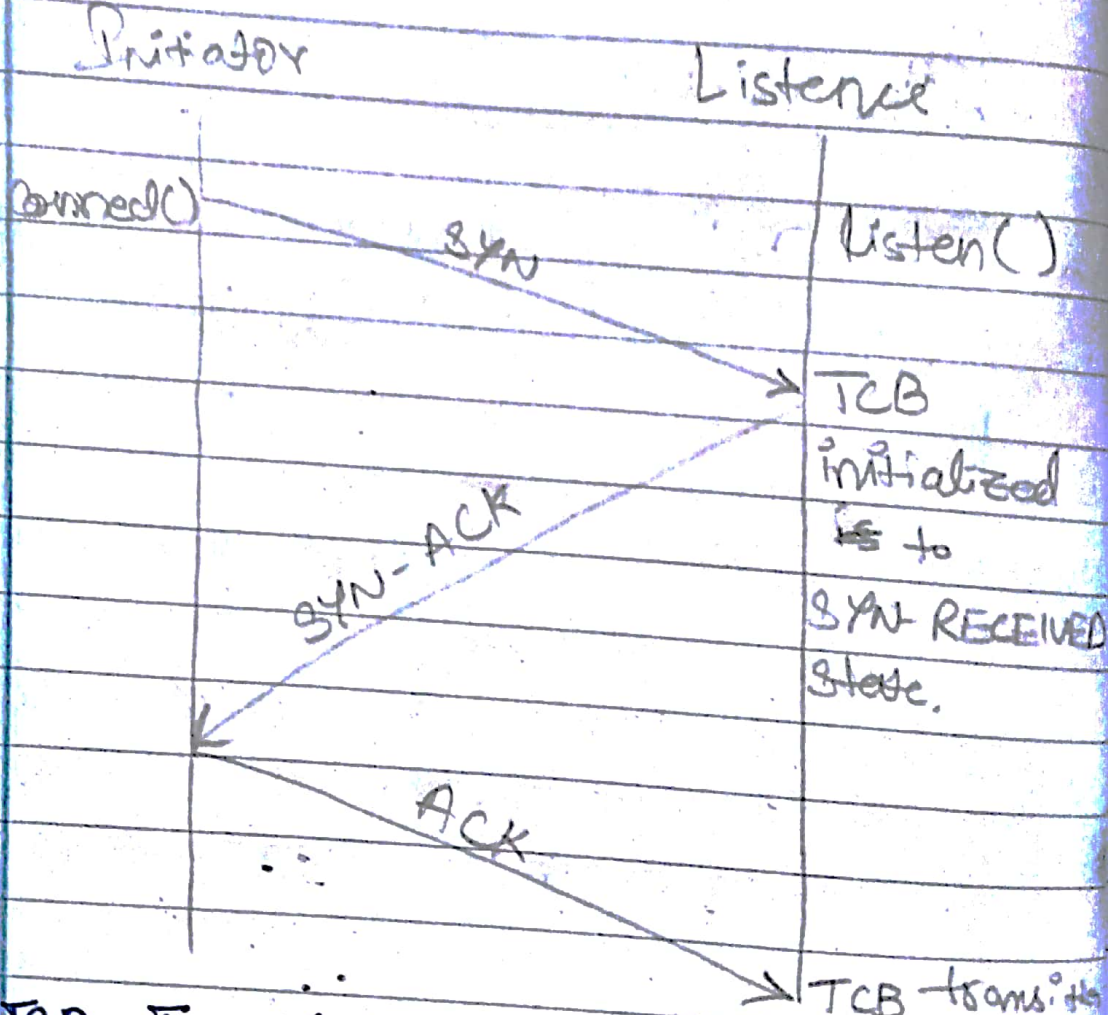
* It also wreaks havoc for performance & SDLC.

* Assignment:

- DRM for Streaming media.
- DRM for P2P Applications

INTERNAL SECURITY

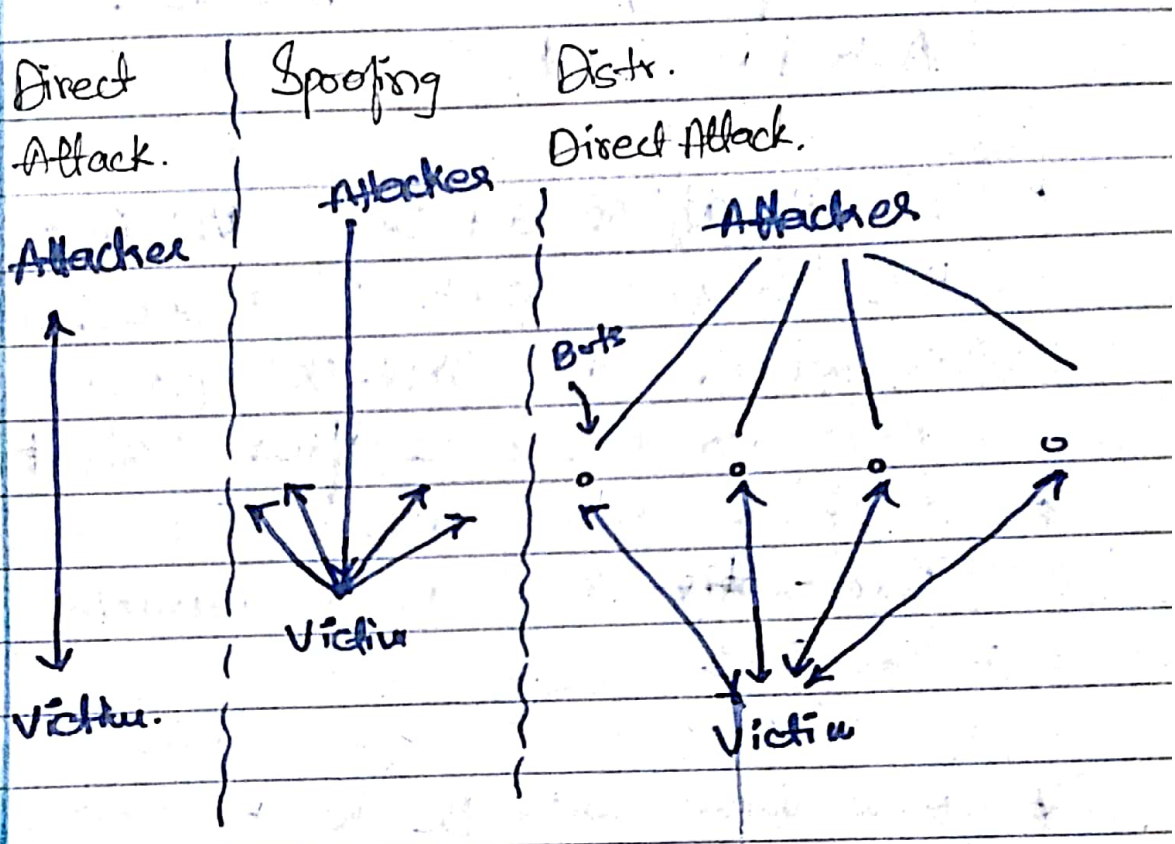
DDOS/DDOS ATTACKS & DEFENCE :-



* TCB = Transmission Control Block (280 bytes - 1300 bytes.)
* TCB Reclaim time = 189 secs.

* Backlog parameters = max no. of TCB in SYN-RECEIVED state allowed simultaneously.
(128 - 2048)

DOS/DDOS ATTACK VARIANTS:-



* Now the very basic attack of DoS is a ping based attack.

* SYN request is a TCP request/pkt. of which the SYN-bit is 1 & relevant headers are.

* TCB is a data structure that the listener stores in its RAM.

* When the listener sends the SYN-ACK pkt., its SYN bit is 1. & ACK bit is 1.

* Now if after sending SYN-ACK. the listener waits for 3 sec before re sending the SYN-ACK pkt. Then for 5 sec & after that it doubles until reaching TCB Reclaim Time ~~until~~ & then it removes the TCB.

* Now what will be in the TCB? Src. IP, port no., & other things related the network layer.

* Application attach their own header that differentiate the pkts at above layers

* ~~Now first~~ Now lets move on to SYN DDos attack.

* Spoofed. pkts are ~~#~~ easy to trace.

* Now the question is how much traffic we need to send in order to make the DDoS successful?

* Now we just can't send a large no. of pkts. 2 expect our attack to be successful. That is inefficient & resource hogging.

* So we generalize the attack rate required to successfully ~~launch~~ launch a SYN DoS attack

$$\text{Attack Rate} = \frac{(\text{Backlog Param})(\text{Pkt. size})}{(\text{TCB Reclaim Time})}$$

* If the Backlog Param = 128

$$\therefore \text{Attack Rate} = \frac{128 \times (20 + 20)}{\text{TCB Re. Time} = 189.}$$

(IP header size, TCP header size)

$$\Rightarrow \text{Attack Rate} = \boxed{27 \text{ bytes/sec}}.$$

* Now this rate needs to revised every 189 sec. Since all the created TCB will be discarded.

* Now what is the solution to this?

* There could be multiple solutions/ protection mechanism.

* What if, after receiving SYN, we don't allocate the TCB?

* But then we see that the since one server is serving multiple machines & we need to distinguish b/w different users.

* But what if we receive the TCB encoding ~~at~~ in the sequence no. i.e. at the step 2 (when the client send ACK after SYN-ACK).

* This can be implemented.

* Now ~~ever~~ if the attacker follows

the same method, the attack won't be successful.

* Although there is still a way & that is to send a large no. of pkts & choke the n/w of the victim.

* But ~~do~~ to do this, ~~or~~ he/she would need to do several things that are quite difficult to do.

* But there is another solution to this.

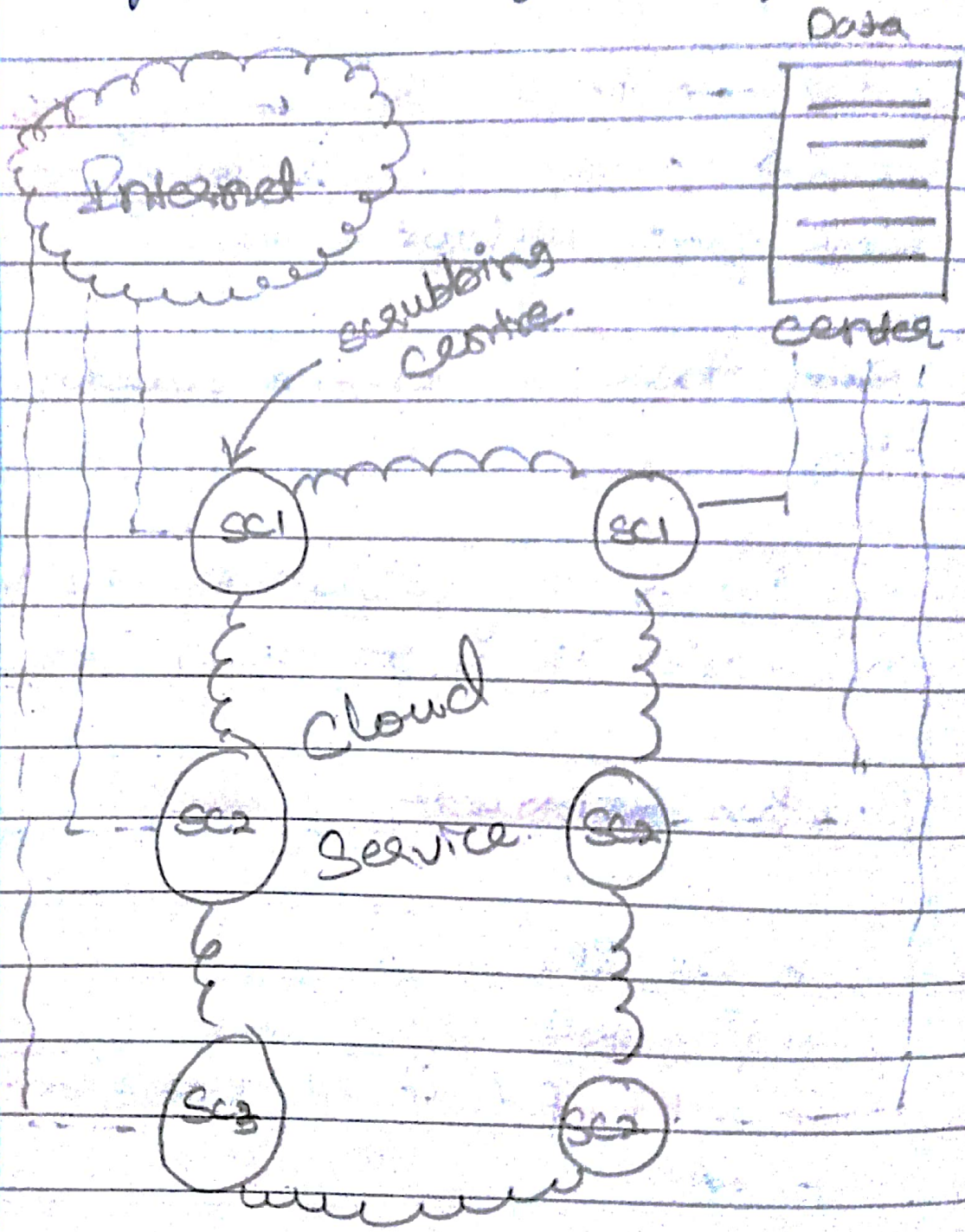
* The larger the volume of the traffic the easier it is to catch/trace.

Reading Assignment:

→ IDS/IPS
→ Firewall
→ APT (Advanced Persistent Threat)
→ AAA
→ Hash functions.

* Scrubbing Centre (Peering)

* Regional ~~Redirect~~ ~~Attack~~. DNS Redirect. BGP Multiple Roots (2 ways of dividing the traffic ~~is different~~ through different gateways).



* Within the SOC of Scrubbing Centre

~~strict~~ they could ~~be~~ break the tunnel (for SSL) ~~if~~ by offering fake certificates.

*