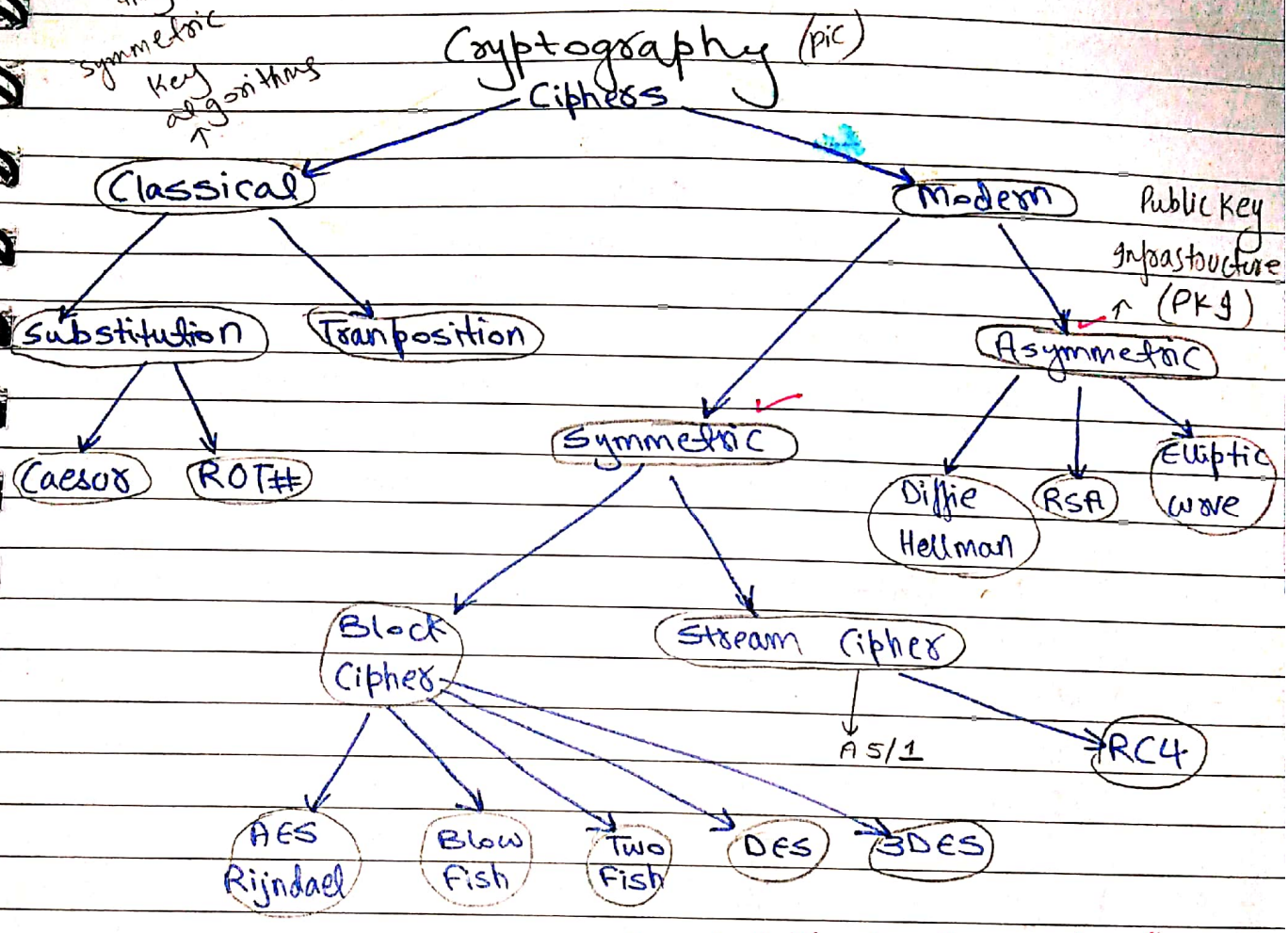
lec # 18

Date:

till now → symmetric ciphers
basic symmetric key ciphers,
Data Encryption stand. (DES)
AES

Classical: they are symmetric key algorithms



the hybrid of these together gives you cmplt bullet prove end to end security.

those who have done the pgp emails, got the idea of combination of symmetric & assymmetric key algo. to achieve end to end 'your' data security!

- dont rely on https etc you need to add your own layer security. Not only rely on providers's security.

Date:

Block Cipher Modes

AES → very strong crypto algo.
DES → weak " "
3DES → bit strong " "

↳ there's still issue →
they apply on a plaintext to make very very unpredictable ciphertext but if somehow you guessed the equality behaviour b/w plaintext then it'll be a disaster.

- we were just talking about block passing through them multiple rounds etc.
but in real life we have multiple blocks.

Multiple Blocks

- How to encrypt multiple blocks?
- Do we need a new key for each block?
↳ If so, as impractical as a one-time pad!

- Encrypt each block independently? or encryption of each
If a file is divided into one million block? block
↳ means sharing one million keys? dependently.

- Is there any analog of codebook "additive"?
- How to handle partial blocks? → there's no concept of half block so we will do padding & reverse padding.
↳ we won't discuss this issue

Date:

Modes of Operation

we need to apply these algorithms of symmetric key cryptography which apply in blocks, we need to apply them in Modes.

↳ Many modes — we discuss 3 most popular.

* Electronic Codebook (ECB) mode
  ↳ encrypt each block independently.
  ↳ most obvious approach, but a bad idea.

* Cipher Block Chaining (CBC) mode
  ↳ chain the blocks together
  ↳ More secure than ECB, virtually no extra work.
  ↳ that's more strong technique

* Counter Mode (CTR) mode
  ↳ Block cipher acts like a stream cipher
  ↳ Popular for random access.

we are trying to make use of these crypto algo. in modes. like,

ECB mode

- Notation : $C = E(P, K)$
- Given plaintext $P_0, P_1, \ldots, P_m, \ldots$
- Most obvious way to use a block cipher: is ECB mode

Encrypt	Decrypt
$C_0 = E(P_0, K)$	$P_0 = D(C_0, K)$
$C_1 = E(P_1, K)$	$P_1 = D(C_1, K)$
$C_2 = E(P_2, K)$...	$P_2 = D(C_2, K)$..

It can be AES algo.

→ by using same algo., same key

If these blocks come out of order still your decryptor process will work cause there's no dependency b/w the

Date:

decryption process & on encryption process in each block.

Each block is encrypted independently.

" " " decrypted "

this is an advan. that whatever block you get you can decrypt it.

But Pb here?

ECB Cut and Paste

- Suppose plaintext is:
  Alice digs Bob. Trudy digs Tom.
- Assuming 64-bit blocks and 8-bit ASCII:
  we've 64-bit blocks & each character takes 8-bits ASCII.
  So we divide this entire msg into 4 blocks. Each block is of 64 bit (P_0, P_1, P_2, P_3)
  P_0 = "Alice di", P_1 = "gs Bob.", } these are independent blocks.
  P_2 = "Trudy di", P_3 = "gs Tom."
  ↳ we apply whatever strong algo., we get,
- Ciphertext : C_0, C_1, C_2, C_3
  ↳ these crypto msgs can be read by anyone
  Trudy can read this crypto messages &
- Trudy will cuts and pastes : C_0, C_3, C_2, C_1
- then decrypt as,
  Alice digs Tom. Trudy digs Bob.
  ↳ msg reorganization

can be done i.e. the pb of ECB

* independent ~~decry~~ decryption of blocks leads to this Cut & paste pb. → can be used to change the msg.

Date:

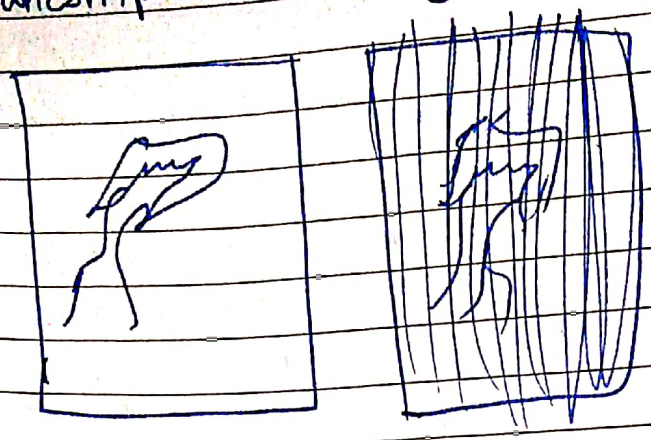
ECB weakness

- Suppose $P_i = P_j$
  P_i is being encrypted separately & P_j is being encrypted seperately.
  There's no dependence b/w P_i & P_j
  Ovio we are using same key. so,
- Then, $C_i = C_j$ and Trudy knows $P_i = P_j$
↳ This gives Trudy some information, even if she does not know P_i or P_j
- Trudy might know ↑any of the block P_i
- Is this a serious issue?
  Repeatedly in ciphertext like here ($C_i = C_j$) cuz C_i will be at diff. place & C_j will be diff. place
  most of the time hackers dont try to break crypto algo. they investigate ciphertexts (they do cryptanalysis). ↳ they see are there any repeating blocks.
  repeating blks gives idea of msg & if the msg is short these pb is more serious.
  How to solve?
  ↳ Solution is you need to add dependency or corelation b/w the encryption of each block.
  Each block or subsequent blocks must not be encrypted separately. There should be some kind of dependence b/w encryption of sub blocks.

Date:

Alice Hates ECB Mode

Eg. that
- Alice's uncompressed image, & ECB encrypted (TEA)



- Why does this happen?

main drawback → ↳ Ovio, same plaintext yields same ciphertext! that's why Alice hates ECB mode.

CBC Mode → good eg of professional way of encryption.

* Even if you use a very very strong crypto algo. like AES — 256bit key.
  → if you use this algo. to encrypt your block independently than that's totally useless. So, you need to use these /apply these algo. in a very clever way.
* CBC Mode is a great way to achieve more security
  - Blocks are "chained together".
  - A random initialization, or IV is required to initialize CBC mode.
  - IV is random, but not secret.

idea here → process is similar

Date:

Encryption	Decryption
$C_0 = E(IV \oplus P_0, K)$	$P_0 = IV \oplus D(C_0, K)$,
$C_1 = E(C_0 \oplus P_1, K)$ → uses C_0 (previous ciphertext)	$P_1 = C_0 \oplus D(C_1, K)$,
$C_2 = E(C_1 \oplus P_2, K)$, ...	$P_2 = C_1 \oplus D(C_2, K)$, ...

- Analogous to classic codebook with additive

↳ Any stage ciphertext depends on the ciphertext of previous stage. i.e. kind of sequential dependence b/w ciphertext of each block.

Q/ why there is need of IV? what if i dont use IV? ↳ dependency will be there, then why IV? { live class

w/o IV

↳ Now repetition of plaintext will be not issue here! then ciphertext will be diff. of same plaintext.

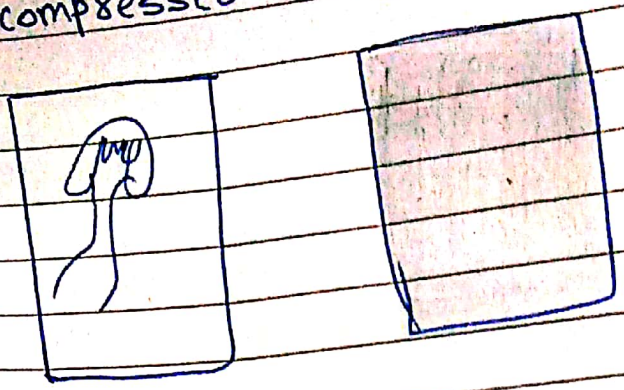
- Identical plaintext blocks yield different ciphertext blocks — this is very good!
- But what about errors in transmission?
  - ↳ If C_1 garbled to, say, G_1 then
    $P_1 \neq C_0 D(G_1, K)$, $P_2 \neq G_1 \oplus D(C_2, K)$
  - ↳ But $P_3 = C_2 \oplus D(C_3, K)$, $P_4 = C_3 \oplus D(C_4, K)$, ...
  - ↳ Automatically recovers from errors!
- Cut and paste is still possible, but more complex (and will cause garbles).

(will discuss in live class)

Date:

Alice likes CBC mode

- Alice's uncompressed image, Alice CBC encrypted (TEA).



- Why does this happen?
  ↳ same plaintext yields different ciphertext.

* previous techniques can not be use in a random manner. they work sequentially. C_1 can't be decrypted until we get C_0.))

Counter Mode (CTR)

- CTR is popular for random access. block comes out of order?
- use block cipher like a stream cipher.

Encryption	Decryption
$C_0 = P_0 \oplus E(IV, K)$,	$P_0 = C_0 \oplus E(IV, K)$,
$C_1 = P_1 \oplus E(IV+1, K)$,	$P_1 = C_1 \oplus E(IV+1, K)$,
$C_2 = P_2 \oplus E(IV+2, K)$, ...	$P_2 = C_2 \oplus E(IV+2, K)$, ...

↳ logic is diff. we is not encrypting plaintext, we are encrypting initial vector with key as IV is same size as block size

* Its a real time algo.

Note : CBC also works for random access
↳ But there is a significant limitation...
↳ has speed of efficiency.

It has efficiency also, you can prepare these $E(IV, K)$, $E(IV+1, K)$, ... these encryption process in advance as encryption takes more time. XOR is a very quick process.

Lec # 19

Date:

Integrity.

till now we were using cryptography for the purpose of confidentiality

Another very imp. aspect in case of data security is to maintain data integrity.

i.e. any tempering in the data must be detected. There should no tempering at all.

If data is ~~encrypted~~ tempered, it still can be encrypted.

data must be secured & it must be ~~encrypted~~ integrated → there must be no kind of tempering in your data.

just doing encryption is one phase of data.

You have to ensure both confidentiality & integrity of your data.

integrity is critical in case of inter bank fund transfer the confidentiality.

Encryption provides Confidentiality → prevents unauthorized disclosure.

Integrity – detect unauthorized writing (i.e. detect unauthorized mod of data)

- Example: Inter-bank fund transfers.
  - ↳ Confidentiality may be nice, integrity is critical
- Encryption provides "confidentiality" (prevents unauthorized disclosure)
- Encryption alone does not provide integrity
  - ↳ One-time pad, ECB cut-and-paste, etc, etc.

Date:

One of the most famous method to aap apply/ enforce integrity is,

MAC

- Message Authentication code (MAC)
  - ↳ used for data integrity
  - ↳ Integrity not the same as confidentiality.

- MAC is computed as CBC residue

  Cyber block chaining → It introduces dependences b/w each blocks & encrypt the data
  - ↳ That is compute CBC encryption, saving only final ciphertext block, the MAC
  - ↳ The MAC serves as a cryptographic checksum for data.

MAC Computation

- MAC computation (assuming N blocks)

$$C_0 = E(IV \oplus P_0, K),$$
$$C_1 = E(C_0 \oplus P_1, K),$$
$$C_2 = E(C_1 \oplus P_2, K), \ldots$$
$$C_{N-1} = E(C_{N-2} \oplus P_{N-1}, K) = MAC$$

- We send $IV, P_0, P_1, \ldots, P_{N-1}$ and MAC
- Reciever does same computation & verifies that result agrees with MAC.
- Both sender & receiver must know K. ↓ symmetric key.

* this is a way to ensure data integrity.

Date:

How Does a MAC work?

- Suppose Alice has 4 plaintext blocks
- Alice computes

$C_0 = E(IV \oplus P_0, K)$, $C_1 = E(C_0 \oplus P_1, K)$,
$C_2 = E(C_1 \oplus P_2, K)$, $C_3 = E(C_2 \oplus P_3, K) = MAC$

↓ It can be any encryption algo. AES, DES or any symmitric key crypto algo.

↳ last cyphertext is your msg. authentication code.

- Alice sends IV, P_0, P_1, P_2, P_3 and MAC to Bob. (receiver)
  - ↳ there's no transmission of C_0, C_1, C_2, C_3
  - ↳ all the intermediate blocks are discarded accept the C_3 that becomes the MAC.
- Suppose Trudy changes P_1 to X.
- Bob computes

$C_0 = E(IV \oplus P_0, K)$, $C_1 = E(C_0 \oplus X, K)$,
$C_2 = E(C_1 \oplus P_2, K)$, $C_3 = E(C_2 \oplus P_3, K) = MAC \neq MAC$

Trudy changed the plaintext P_1 to X

So C_1 changed → C_2 changed → ... → MAC changed!

- It works since error propagates into MAC.
- Trudy cant make $MAC \neq MAC$ without K.

Date:

Confidentiality and Integrity

- Encrypt with one key, MAC with another key.
  both encryption & integrity
  we were studying just 'integrity', now we can do together

- Why not use the same key?
  ↳ send last encrypted block (MAC) twice?
  ↳ This cannot add any security!
  because the last block is also MAC block.

- Using different keys to encrypt and compute MAC works, even if keys are related.
  ↳ but, add twice as much work as encryption alone
  ↳ can do a little better — about 1.5 "encryptions".

- Confidentiality and Integrity with same work as one encryption is research topic.
  (will discuss in live class)

Uses for Symmetric Crypto

- Confidentiality
  ↳ Transmitting data over insecure channel
  ↳ secure storage on insecure media
- Integrity (MAC)
- Authentication protocols (later...)
- Anything you can do with a hash function (upcoming chp:)

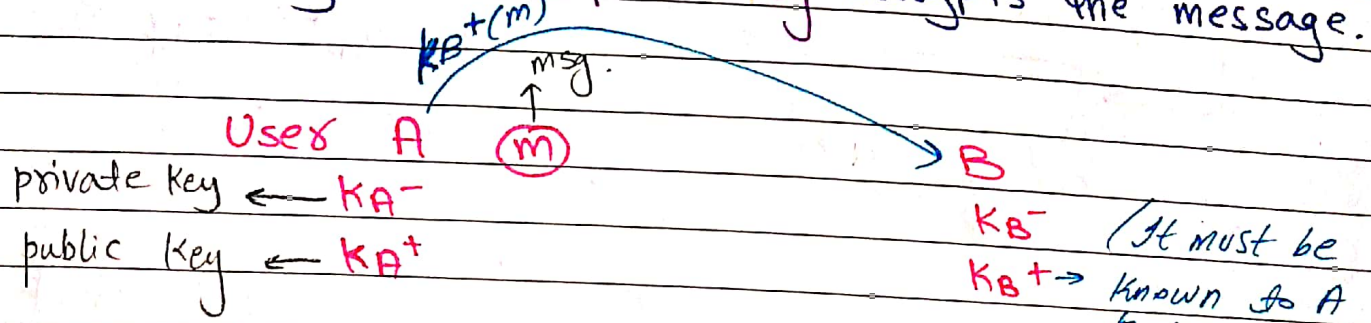
ch 4

Date:

Public Key Cryptography

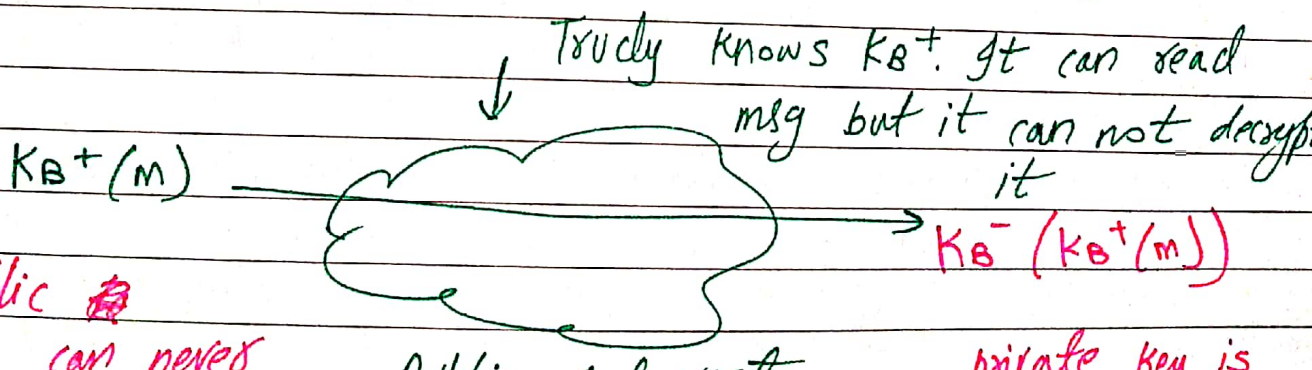
* every entity has two keys private & public
* private key of all the user of any entity is never shared it resides only on its machine.
* the public key of any user is known to all.

- Two keys, one to encrypt, another to decrypt
  - ↳ Alice uses Bob's public key to encrypt
  - ↳ Only Bob's private key decrypts the message.



A wants to send a msg to B. So, A will need the public key of B

Most famous public key algo. is RSA algo.



* Public key can never decrypt msg.

private key is applied on the public key encrypted msg. i.e. m → actual msg.

Date:

If private key is known to Trudy then it'll be a DISASTER!

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

↳ one of the fundamental property of the Public key algorithm.

m → H(m)

If you've a msg 'm', you compute its hash (hash func. of m)

$K_A^-(H(m))$ DS

If you encrypt this hash of m by your private key K_A^-.

↳ this is called as "digital signature" (DS)

receiver has the K_A^+

$K_A^+(K_A^-(H(m))) = H(m)$

this is called as non-repudation

↳ means sender can not deny that it has send the msg.

i.e. the concept of Digital signature

* Apart from confidentiality, integrity, apart from secure end to end msge. There's also the need of 'Non-repudation'.

* How the receiver (B) will ~~verify~~ verify that msg has not been send by A.

↳ receiver has the public key of A. → it'll recompute the msg & confirms that A has sent it.

Date:

- Based on "trap door, one way function"

↳ "One way" means easy to compute in one direction, but hard to " " other direction.

↳ Example: Given p and q, product N = pq easy to compute but hard to find p & q from N.

↳ "Trap door" is used when creating key pairs.