Chapter: MAC and Hash Function

* Authentication Functions

-> Any msg authentication or digital signature has two levels of functionality

(1) At lower level, there must be some sort of function that produces an authenticator i.e a value to be used to authenticate a message

(2) This lower-level function is then used as a primitive in a higher level authentication protocol that enables a receiver to verify the authenticity of a message.

-> Authenticator can be grouped into three classes

(1) Message Encryption: Ciphertext serves as its authenticator.

(2) Message Authentication Code (MAC):

-> A function of the message and a secret key that produces a fixed-length value that serves as authenticator.

Scanned by CamScanner

(3) Hash function : A function that maps a message of any length into a fixed-length hash value.

* Message Encryption

(1) Symmetric Encryption.

→ As the key K is secret and shared only by A and B then confidentiality is provided as no other party can recover the plaintext

→ Also B is assured that message was generated by A because only A possesses K and thus only A can generate legitimate ciphertext using K. thus it provides Authentication.

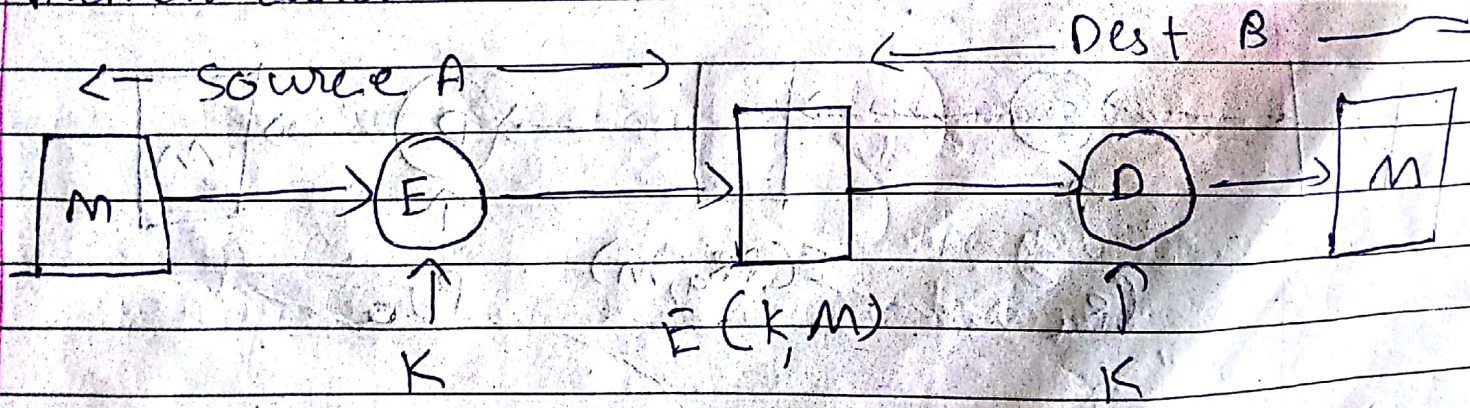
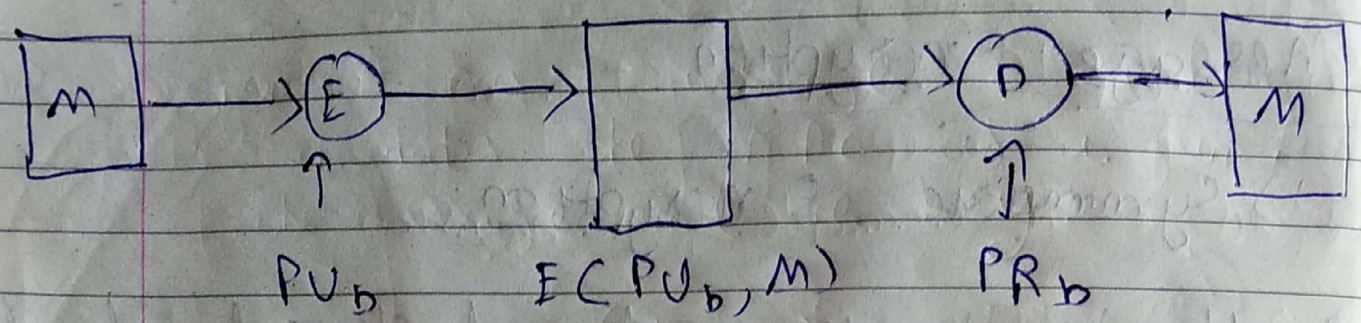


Fig: Symmetric Encryption: Confidentiality and Authentication.

Scanned by CamScanner

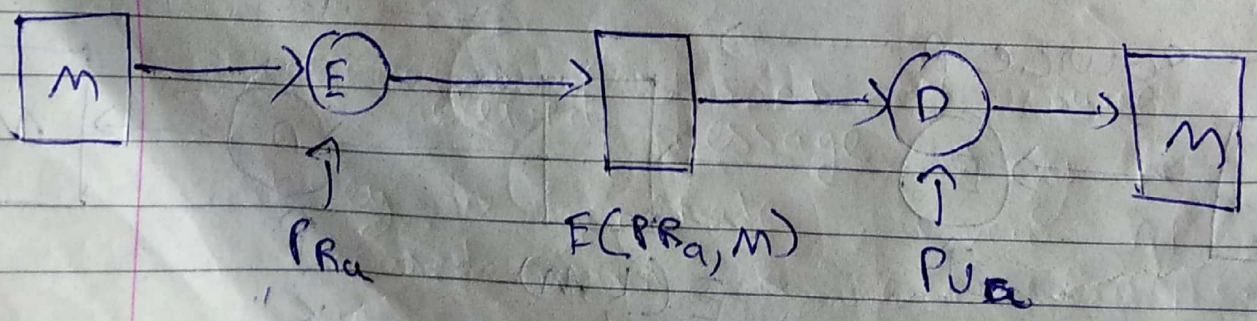
(2) Asymmetric Encryption

Case 1: only confidentiality



→ The above scheme provides confidentiality but no authentication because any opponent could use B's public key to encrypt a message, claiming to be A.

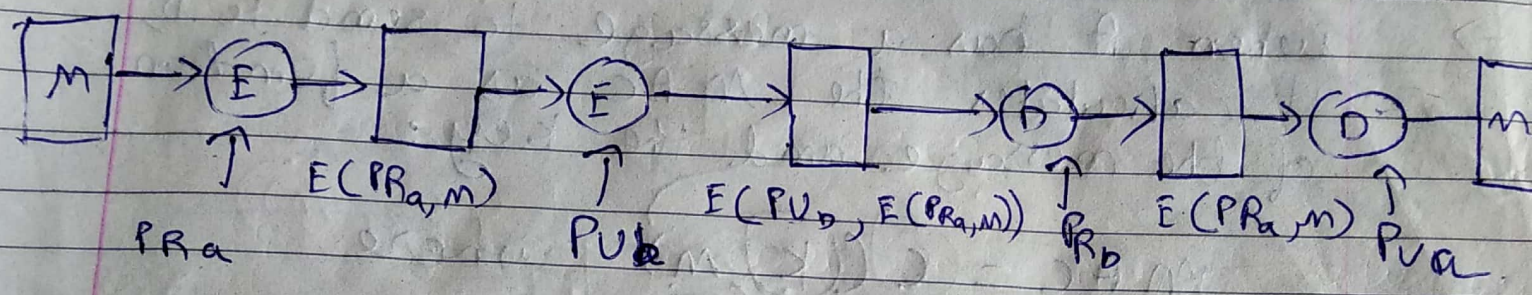
Case 2: Only Authentication/signature



→ The above scheme provides authentication as only A possesses PRa and thus the only party to generate ciphertext using PRa is A. But it does not provide confidentiality

as anyone in possession of A's public key can decrypt the ciphertext.

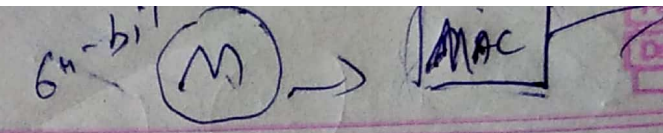
Case 3 : Confidentiality and Authentication/Signat



→ To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality.

→ The disadvantage of this approach is that public key algorithm, which is complex must be exercised four times rather than two in each communication.

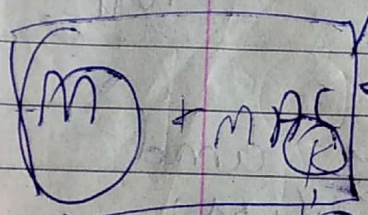
Scanned by CamScanner



* Message Authentication Code

→ This technique assumes that two communicating parties, say A and B, share a common secret key K.

→ when A has a message to send to B, it calculates the MAC as a function of the message and the key:



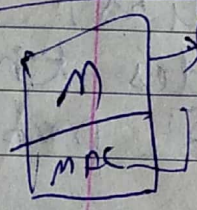
$$MAC = C(K, M)$$ where

M = i/p message

C = MAC function

K = secret key

MAC = Message authentication Code



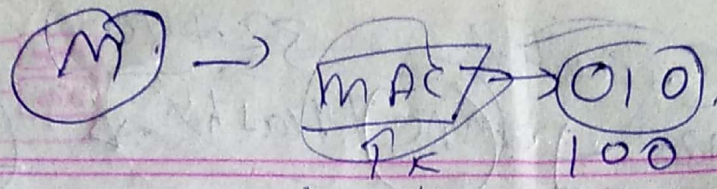
→ The message + MAC are transmitted to receiver.

→ The receiver performs the same calculation on the received message using the same secret key to generate a new MAC which is compared with the received MAC.

→ If both are same then

(1) The receiver is assured that the message is not altered.

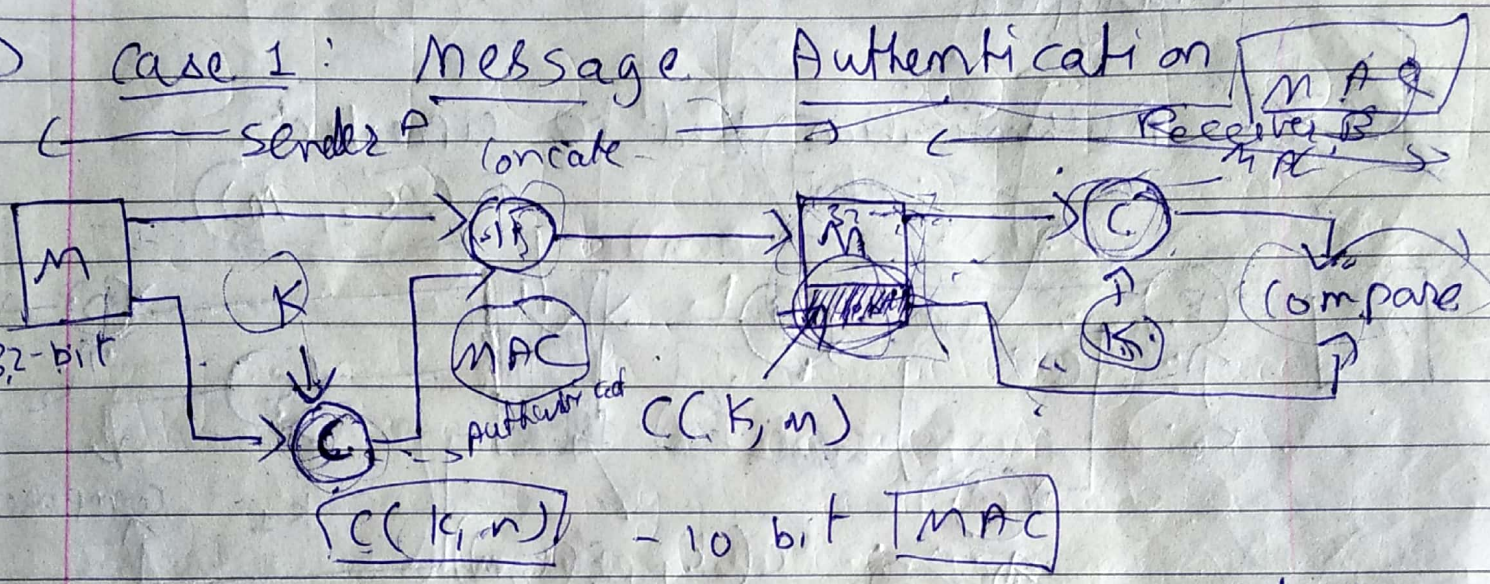
Scanned by CamScanner



(2) The receiver is assured that the message is from the alleged sender.

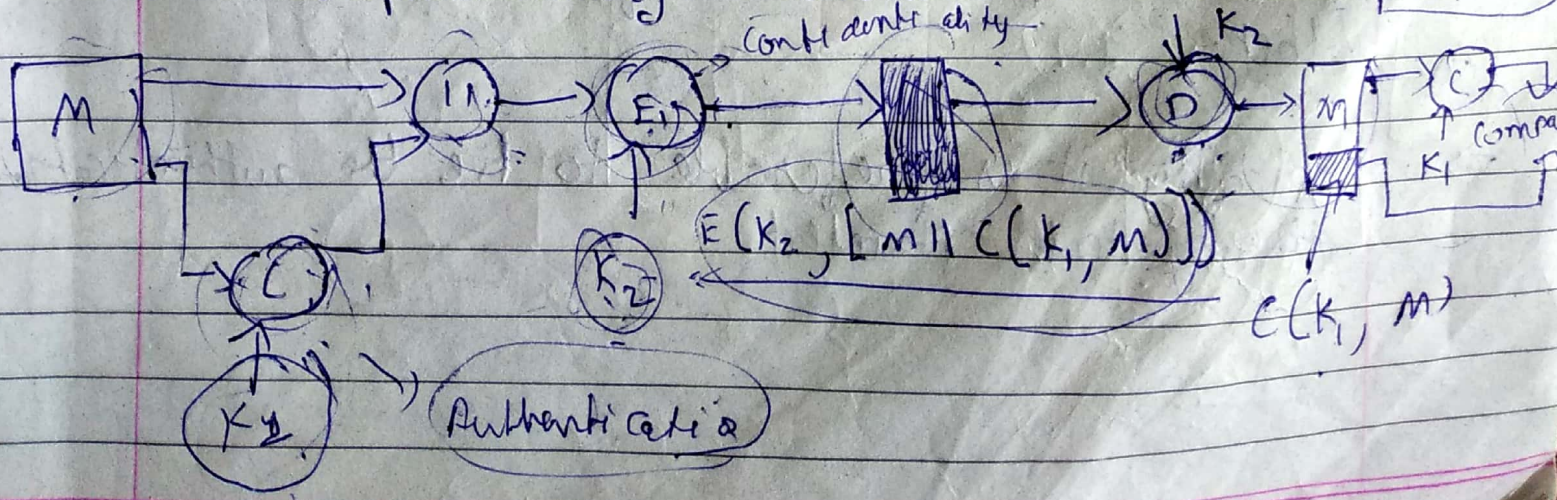
3) If the message includes a sequence no. then the receiver can be assured of proper sequence.

→ Case 1: Message Authentication



→ This scheme provides authentication but not confidentiality because message as a whole is transmitted in the clear.

→ Case 2: Message Authentication and confidentiality: Authentication tied to plaintext



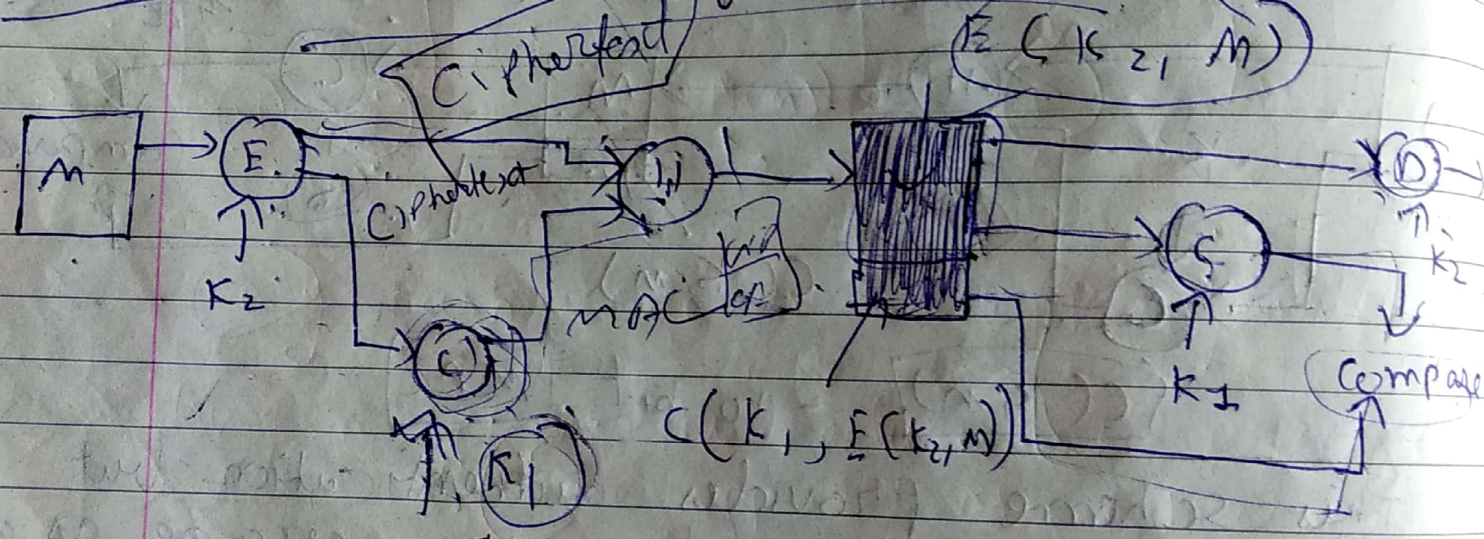
M - 32 Ciphertext 32 + 10

128 → MAC (Hash) → 10

→ In this case confidentiality as well as authentication is achieved.

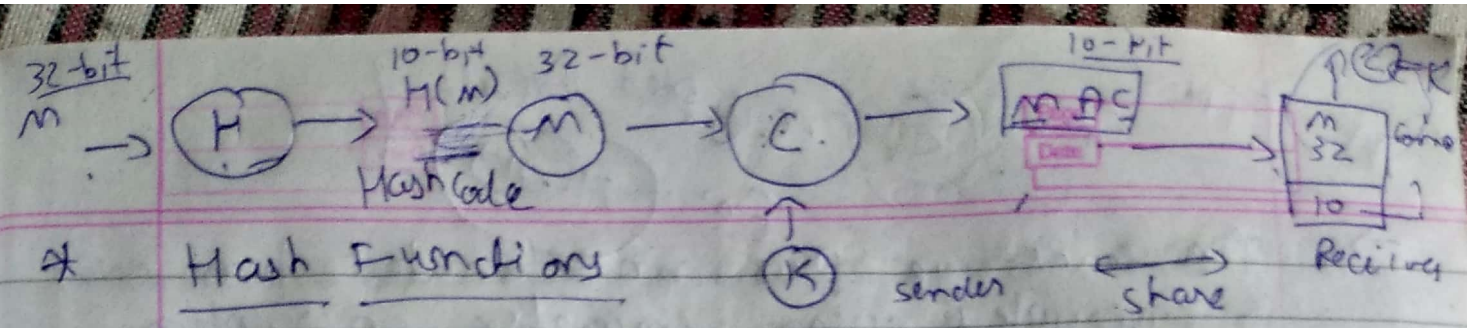
→ Here the MAC is calculated with the message as i/p and is then concatenated to the message. The entire block is then encrypted.

Case 3 : Authentication tied to ciphertext



→ In this case, the message is encrypted first. Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block.

→ Typically it is preferable to tie the authentication directly to the plaintext



* Hash Functions

→ A hash function accepts a variable size message M as input and produces a fixed size o/p, referred to as a hash code H(M).

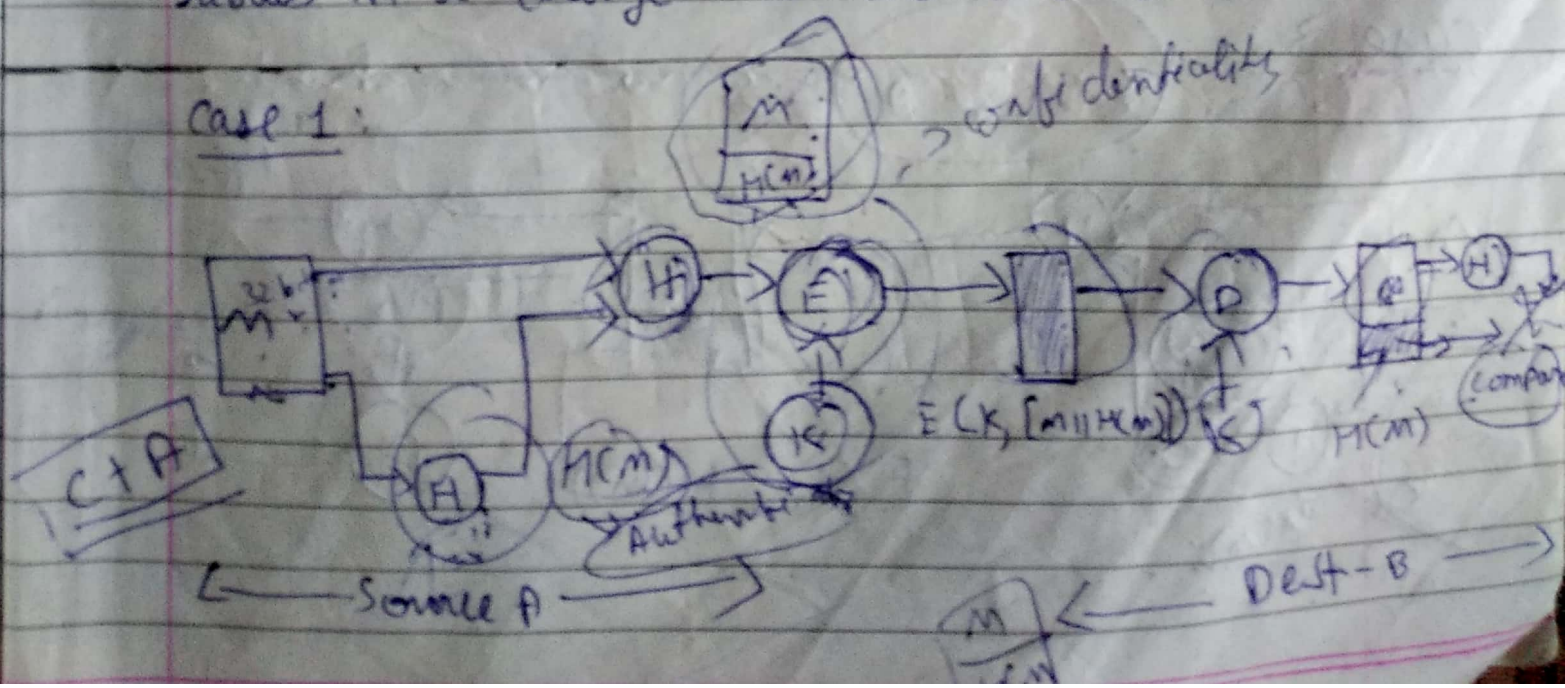
→ Unlike a MAC, a hash code does not use a key but is a function only of the i/p message.

→ The hash code is also referred to as a message digest or hash value.

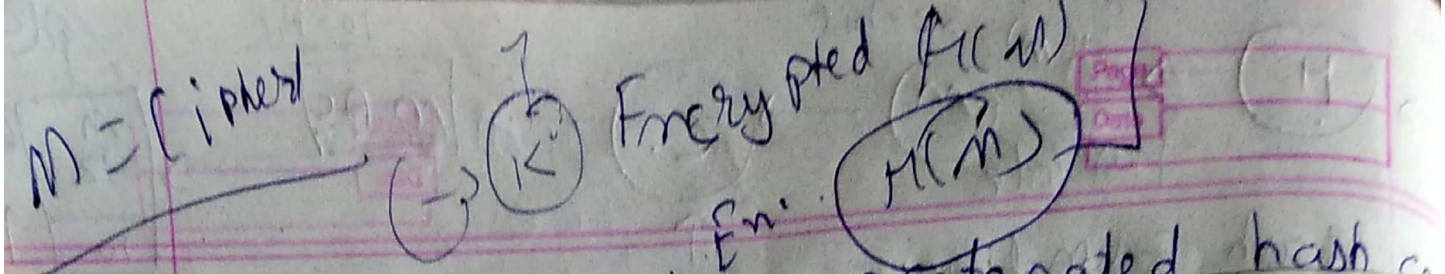
→ The hash code is a function of all bits of the message and provides an error-detection capability.

→ A change to any bit or bits in the message results in a change to the hash code.

Case 1:

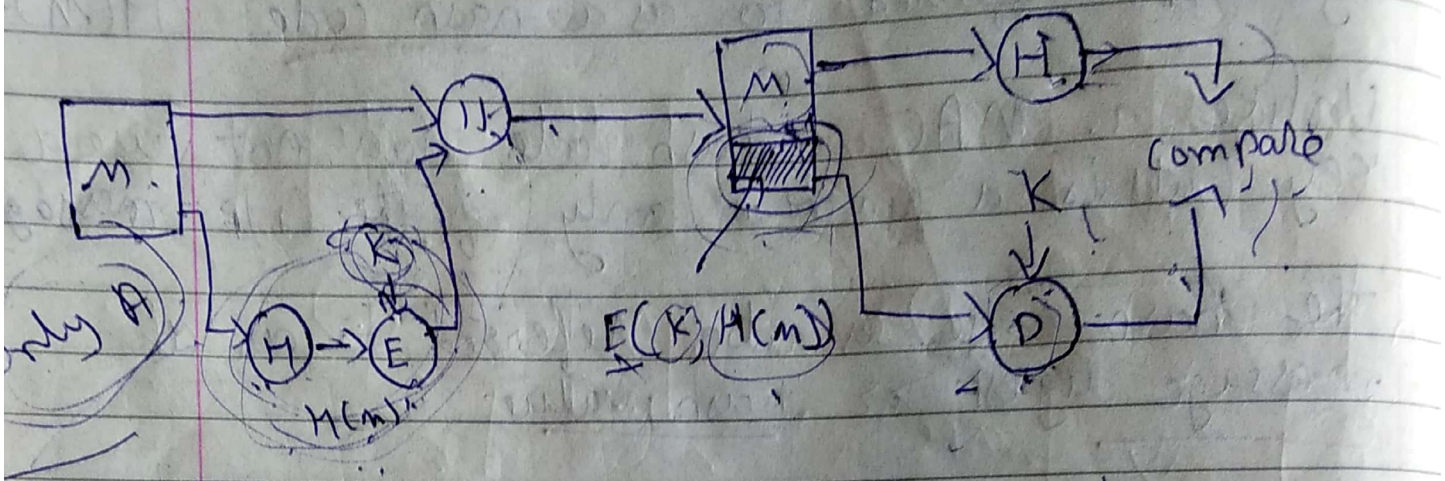


Scanned by CamScanner



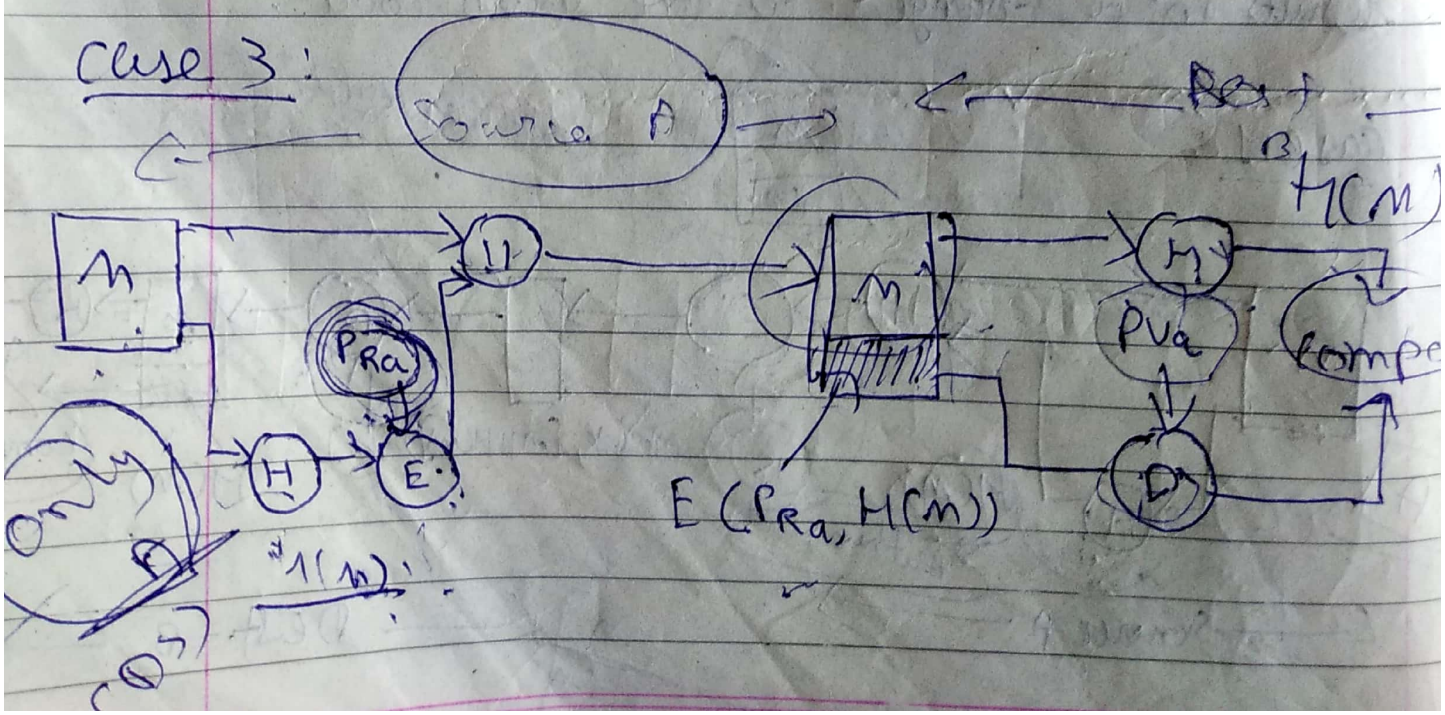
→ The message plus concatenated hash code is encrypted using symmetric encryption.

Case 2 :



→ Only hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

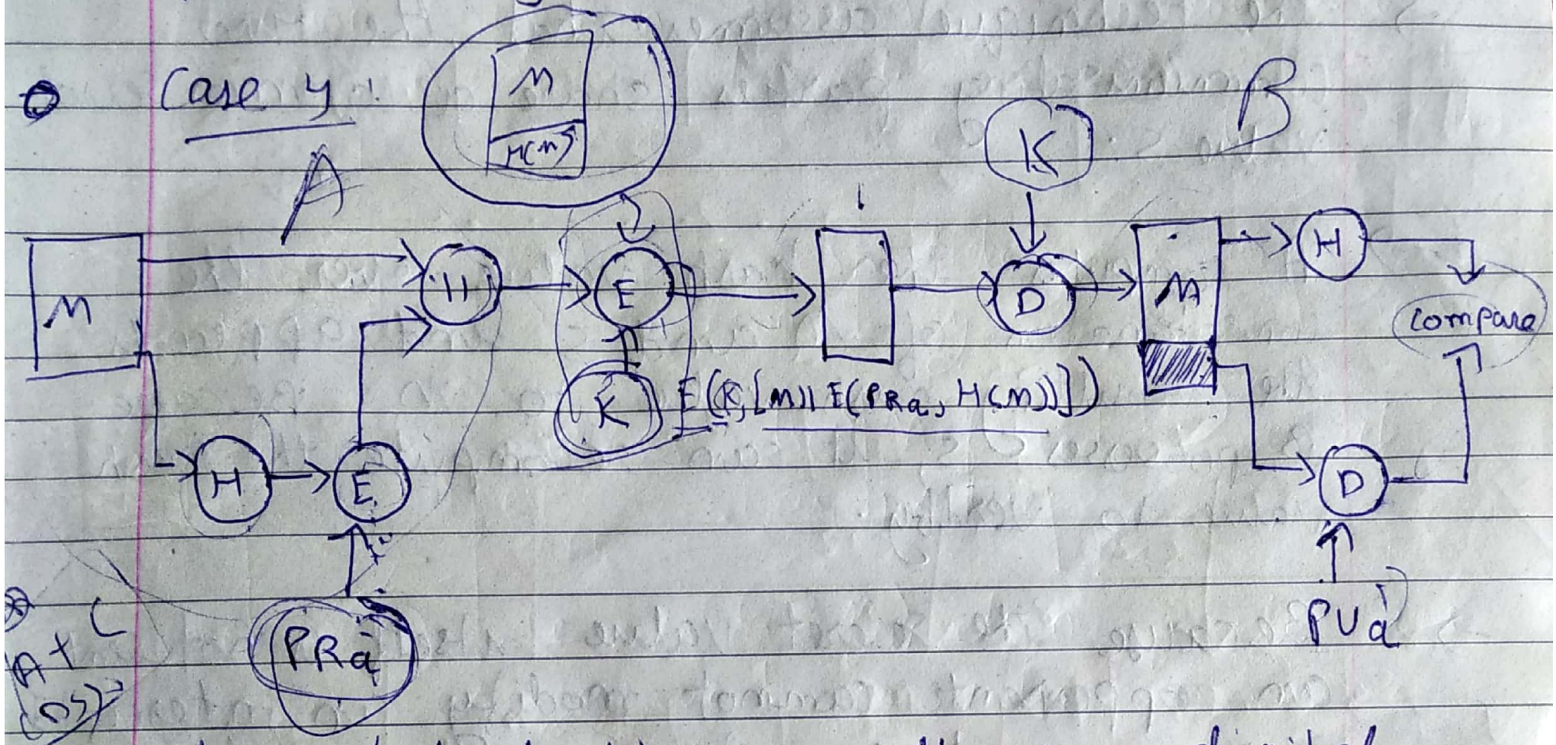
Case 3 :



32 bit
10 bit S
42 bit
42

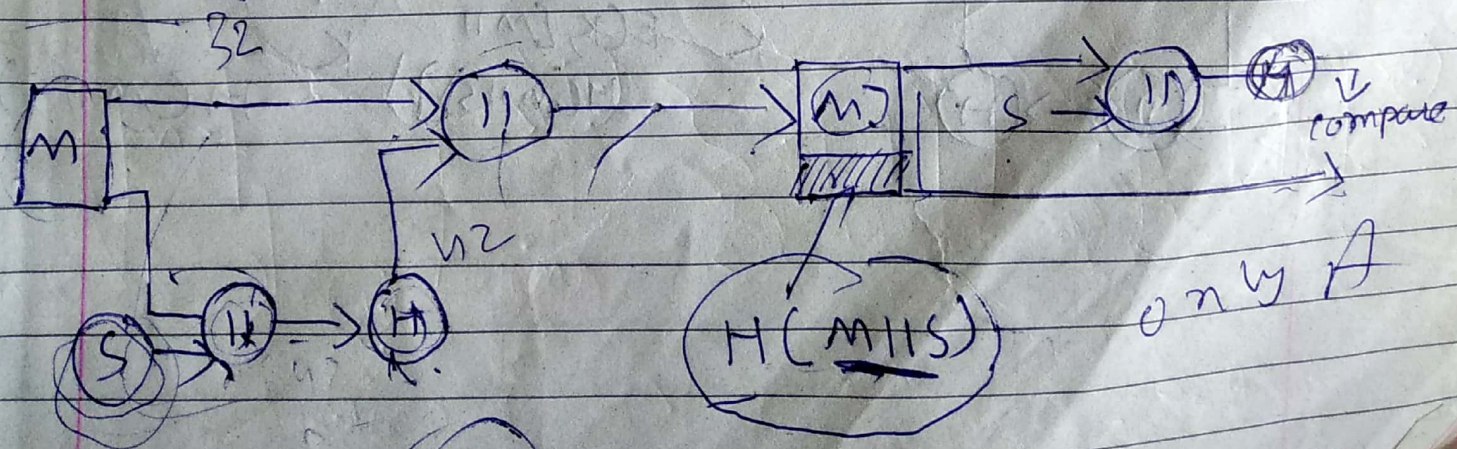
→ Only the hash code is encrypted, using public key encryption and using the sender's private key.

Case 4:



→ If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

Case 5:



79

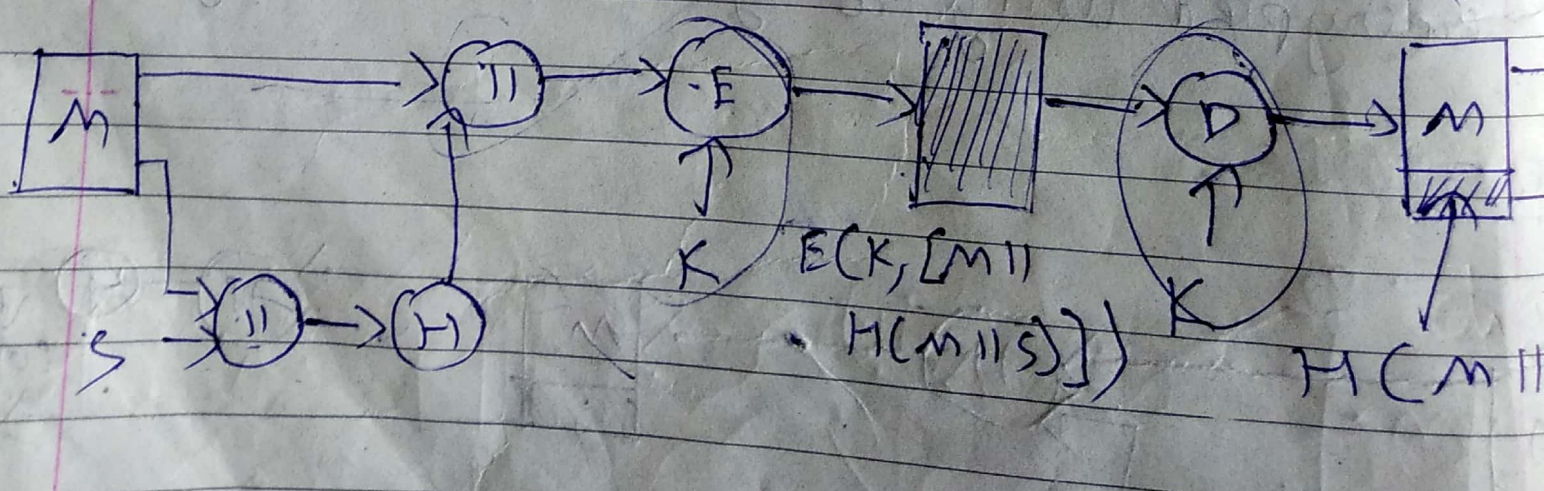
→ It is possible to use a hash function but no encryption for message authentication.

→ The technique assumes that the two communicating parties share a common secret value S.

→ A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify.

→ Because the secret value itself is not sent an opponent cannot modify an intercepted message and cannot generate a false message.

Case 6:

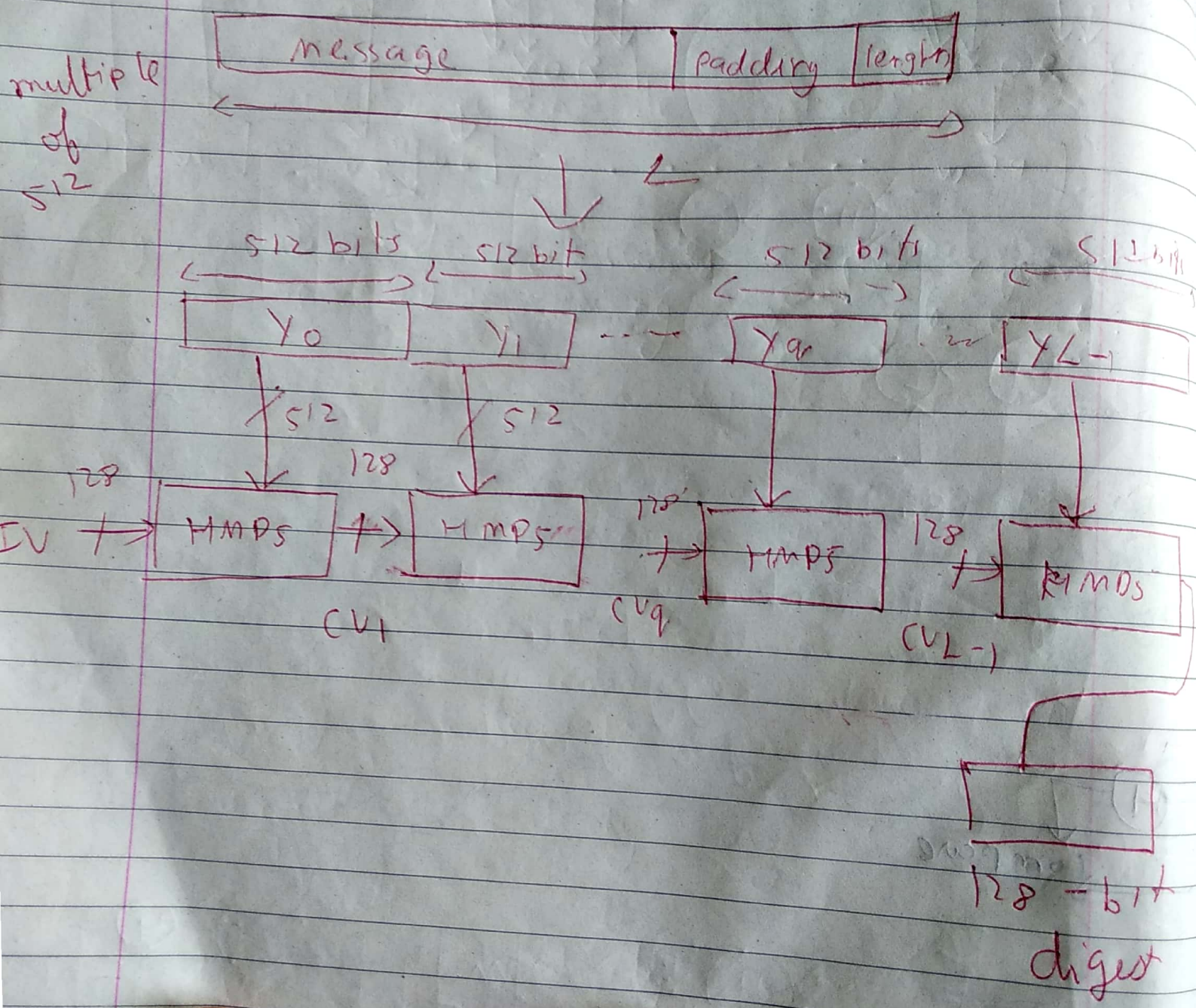


Scanned by CamScanner

confidentiality can be added in case by encrypting the entire message plus the hash code.

Scanned by CamScanner

MD5 Algorithm Architecture



Scanned by CamScanner

SHA - 512

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) + SHR^{7}(x)$$

$$\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x)$$

$ROTR^{n}(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$SHR^{n}(x)$ = left shift of the 64-bit argument x by n bits with padding by zeroes on the right.

Scanned by CamScanner

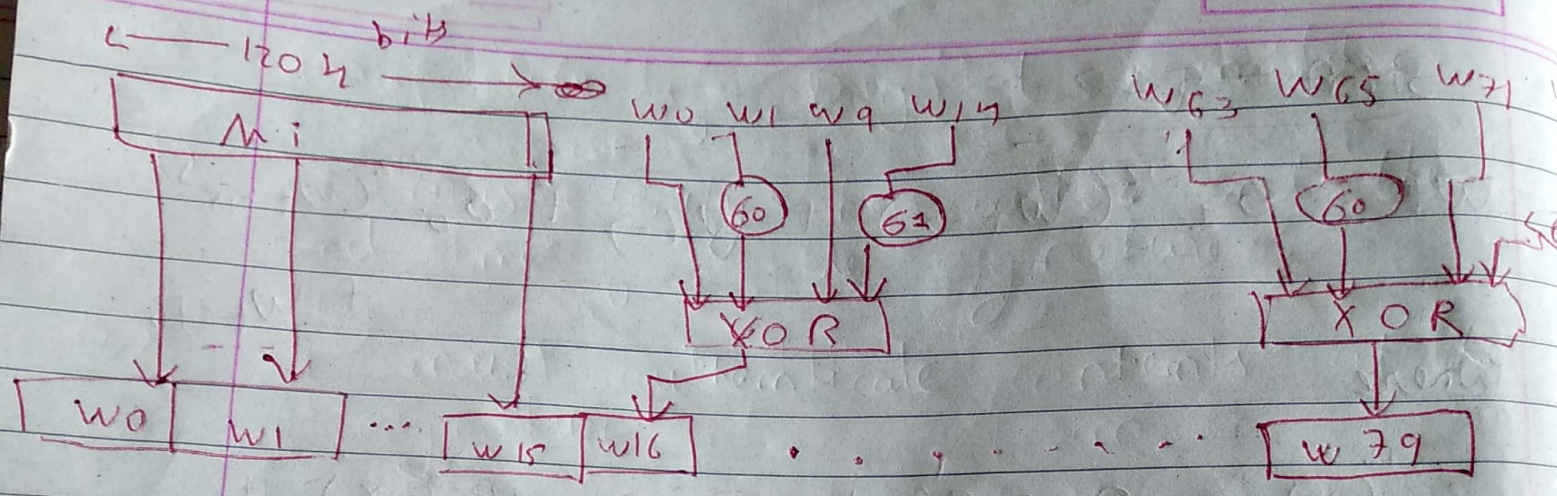


Fig: Creation of 80-word input sequence for SHA-512 processing of single Block.

Scanned by CamScanner

Digital Signature

→ A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.

* **Requirements**

→ Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.

→ Several forms of dispute between two parties are possible.

→ For example, suppose John sends an authenticated message to Mary, using one of the schemes. Following dispute could arise

(1) Mary may forge a different message and claim it came from John.

(2) John can deny sending the message.

→ Both scenarios are legitimate concern.

Scanned by CamScanner

* Properties of Digital signature

(1) It must verify the author and the date and time of the signature.

(2) It must authenticate contents at the time of signature.

(3) It must be verifiable by third parties to resolve disputes.

→ Requirements for a digital signature:

1) The signature must be a bit pattern that depends on the message being signed.

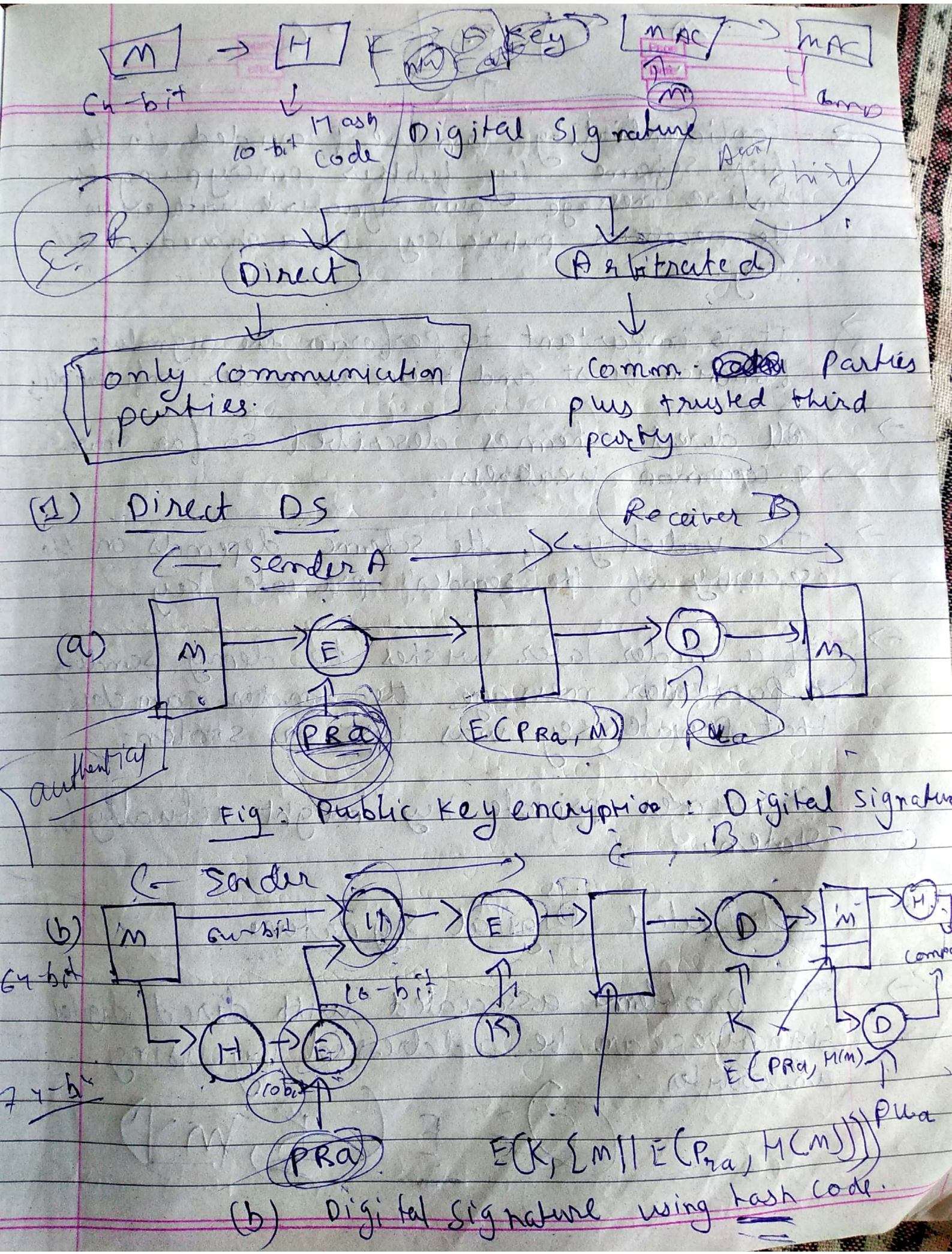
2) The signature must use some information unique to the sender, to prevent both forgery and denial.

3) It must be relatively easy to produce the digital signature.

4) It must be relatively easy to recognize and verify the digital signature.

(5) It must be computationally infeasible to forge a digital signature.

) It must be practical to retain



Scanned by CamScanner

$M \rightarrow \boxed{E} \rightarrow S$

T_{is}

→ Confidentiality can be provided in the first scheme by further encrypting the entire message plus signature with either the receiver's public key or a shared secret key.

→ It is important to perform the signature function first and then an outer confidentiality

→ All direct schemes described so far share a common weakness.

→ The validity of the scheme depends on the security of the sender's private key.

→ If a sender later wishes to deny sending a particular message the sender can claim that private key was lost or stolen.

→ Sometimes private key might actually be stolen.

Arbitrated Digital Signature

→ The problems associated with direct digital signatures can be addressed by using an arbiter.

$$Q = E(K, [M])$$

→ (a) Conventional Encryption, Arbiter sees message.

(1) $X \rightarrow A: M \| E(K_{xa}, [ID_x \| H(M)])$

(2) $A \rightarrow Y: E(K_{ay}, [ID_x \| M \| E(K_{xa}, [ID_x \| H(M)]) \| T])$

→ In the above scheme symmetric encryption is used.

→ It is assumed that the sender X and the arbiter A share secret key K_{xa} and A and Y share K_{ay}.

→ X constructs a message M and H(M).

→ X transmits (M + Signature) to A

→ Signature consists of identifier ID_x of X + hash all encrypted using K_{xa}.

→ A decrypts signature (checks and validate)

→ Then A transmits a message to Y, encrypted with Kay which includes ID_x + original message + signature + timestamp

→ Y can decrypt the message and signature

Scanned by CamScanner

→ The arbiter validates the signature and timestamp in case of dispute.

→ Here both sides must highly trust A.

(b) Conventional Encryption: Arbiter Does Not see message

1) $X \rightarrow A: ID_x \| E(K_{xy}, M) \| E(K_{xa}, [ID_x \| H(E(K_{xy}, M))])$

2) $A \rightarrow Y: E(K_{ay}, [ID_x \| E(K_{xy}, M)]) \| E(K_{xa}, [ID_x$

→ The previous scheme implied that A is able to read messages from X to Y.

→ This schemes provides confidentiality also.

→ It is assumed that X and Y share the secret key K_{xy}.

→ Now, X transmits an identifier, a copy of the message encrypted with K_{xy}, and a signature to A.

The signature consists of Identifier + hash value of encrypte message all encrypted by K_{xa}.

A decrypts the signature and checks hash value to validate the message.

Scanned by CamScanner

→ A is working only with the encrypted version of the message and is prevented from reading it.

→ A then transmits everything that it received from X, plus a timestamp, all encrypted with key to Y.

→ A remaining problem, one shared with first scenario, is that the arbiter could form an alliance with the sender to deny a signed message, or with the receiver to forge the sender's signature.

$H(E(K_{xy}, M) \| T])$

(C) public-key Encryption: Arbiter Does not see msg.

(1) $X \rightarrow A : ID_x \| E(PR_x, [ID_x \| E(PU_y, E(PR_x, M))])$

(2) $A \rightarrow Y : E(PR_a, [ID_x \| E(PU_y, E(PR_x, M)) \| T])$

→ In this case, X double encrypts a message M first with X's private key, PR_x, and then with Y's public key, PU_y.

→ This is a signed, secret version of the message.

Scanned by CamScanner

-> This signed message, together with X's identifier, is encrypted again with PR_x and, together with ID_x, is sent to A.

-> The inner, double encrypted msg is secure.

-> However, A can decrypt the outer encryption to assure that msg have come from X.

-> X's private/public key pair is checked by A and if so, verifies message.

-> Then A transmits a message to Y, encrypted with PR_a.

-> The message includes ID_x, the double encrypted message and a timestamp.

Scanned by CamScanner

* Digital signature Standard

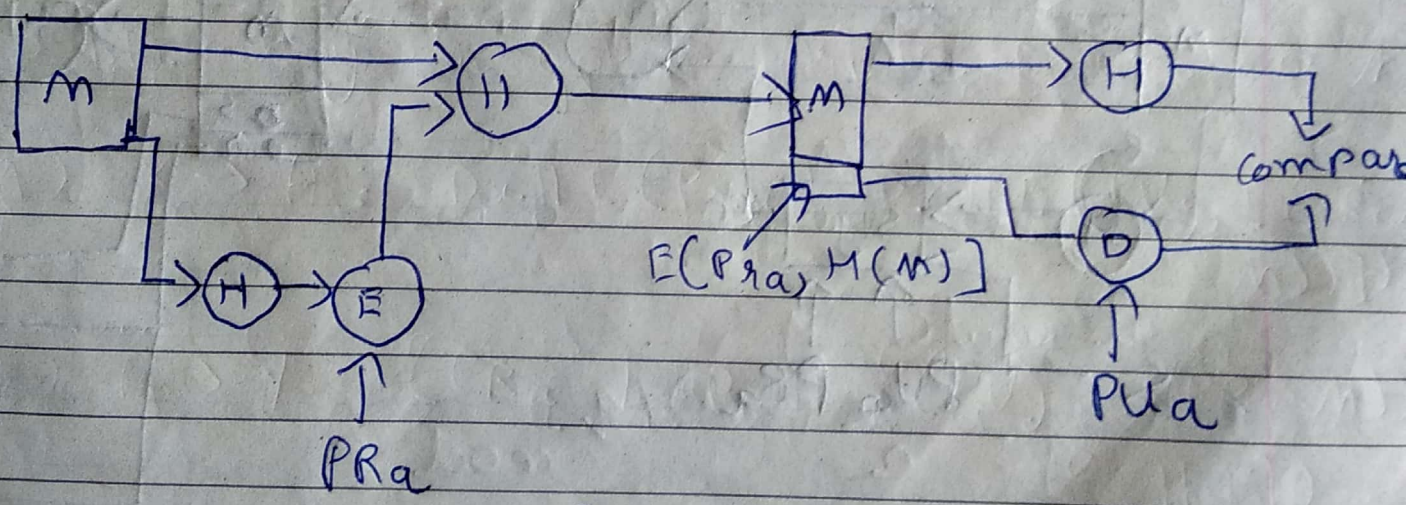
-> The DSS make use of SHA and presents a new digital signature technique.

-> **DSS Approach**

-> DSS uses an algorithm that is designed to provide only the digital signature function

-> Unlike RSA, it cannot be used for encryption or key exchange.

->



(a) RSA Approach

-> In RSA approach, the message to be signed is input to a hash function that produces a SH code of fixed length. This hash code is then encrypted using the sender's private key to form the signature.

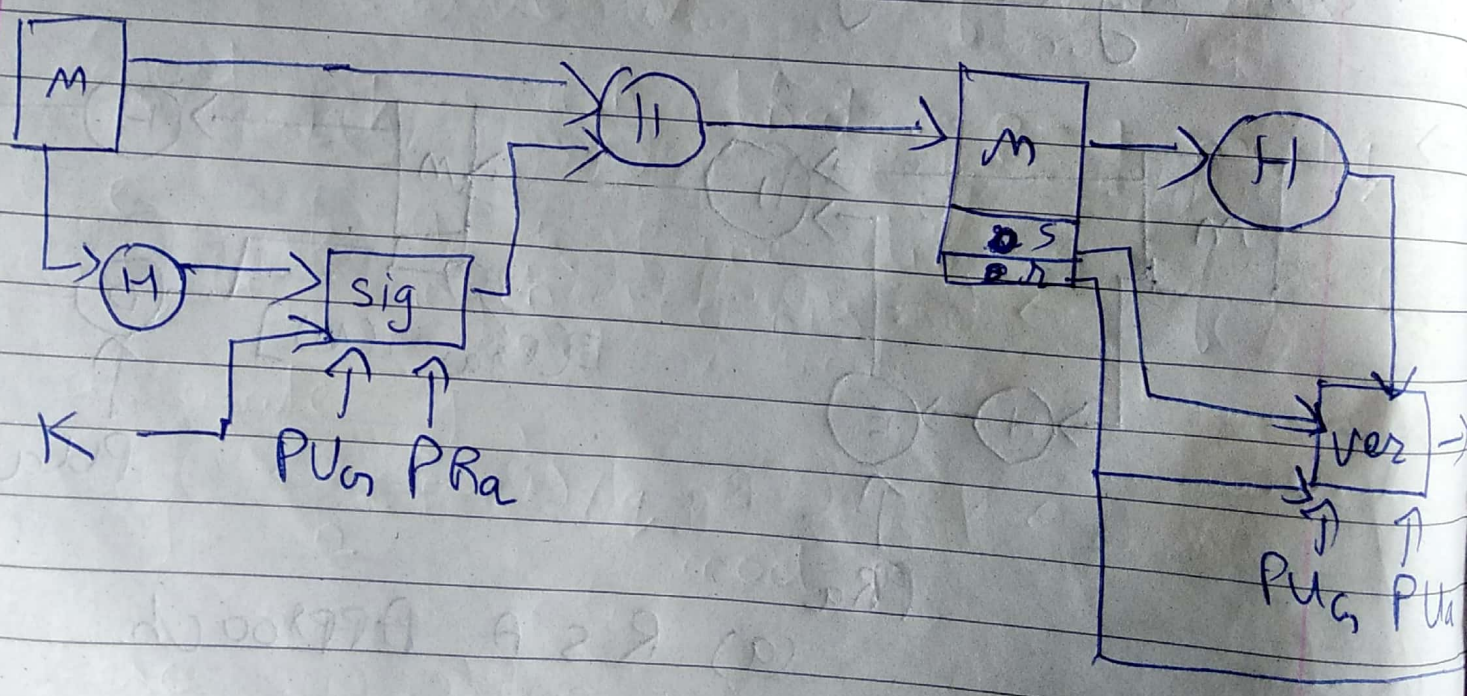
Scanned by CamScanner

→ Both the message and signature are then transmitted.

→ The recipient takes the message and produces a hash code.

→ The recipient also decrypts the signature using the sender's public key.

→ If calculated hash code matches the decrypted signature, the signature is accepted as valid.



(b) DSS approach

Scanned by CamScanner

-> The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random k generated for this particular signature.

-> The signature function also depends on the sender's private key (PRa) and a set of parameters known to a group of communicating principals.

-> We can consider this set to constitute a global key (PUG). The result is a signature consisting of two components, labelled s and r.

-> At the receiving end, the hash code of the incoming message is generated. This plus the signature is i/p to a verification compare function.

-> The verification function also depends on the global public key as well as the sender's public key (PUa), which is paired with the sender's private key.

-> The o/p of the verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of private key could only produce signat

* The Digital Signature Algorithm

→ Global Public - Key Components

Step 1: select prime number p where $2^{L-1} < p < 2^{L}$ for $512 \leq L \leq 1024$ and L is a multiple of 64.

→ i.e bit length between 512 and 1024 bits, in increments of 64 bits.

Step 2: select q such that $p-1 \bmod q = 0$

→ i.e. q is a prime divisor of $p-1$ and bit length 160 bits.

Step 3: Calculate g.

$$g = h^{(p-1)/q} \bmod p,$$

where h is any integer with $1 < h < p-1$ such that

$$h^{(p-1)/q} \bmod p > 1$$

Scanned by CamScanner

→ User's Private Key

Step 4: select x randomly such that $0 < x < q$

→ User's Public Key

Step 5: Calculate $y = g^x \bmod p$

→ User's Per-Message Secret Number

Step 6: Select k = random integer such that $0 < k < q$.

Signing

Step 7: Calculate r

$$r = (g^k \bmod p) \bmod q$$

Step 8: Calculate s.

$$s = [k^{-1}(H(m) + xr)] \bmod q$$

Signature (r, s)

Scanned by CamScanner

Verifying

→ **Step 9:** Calculate w

$$w = (s')^{-1} \bmod q$$

→ **Step 10:** calculate u_1 and u_2

$$u_1 = [H(M')w] \bmod q$$

$$u_2 = (r')w \bmod q$$

→ **Step 11:** Calculate v.

$$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$$

→ **Step 12:** Compare v and r'

if $v = r'$ message is sent by sender only.

Note: M = message to be signed

$H(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s.

Scanned by CamScanner

1) Sum for DSA

Given data: $H(M) = 3$

$p = 7$

$h = 2$

$k = 2$

Solution: q should be such that $p-1 \bmod q = 0$

Step 1

$\therefore \quad 7 - 1 \bmod q = 0$

$6 \bmod q = 0$

Hence we can choose $q = 3$ which computes

$6 \bmod 3 = 0$

Step 2: calculate g.

$$g = h^{(p-1)/q} \bmod p$$

$$\therefore \quad g = 2^{6/3} \bmod 7$$

$$g = 4 \bmod 7$$

$$\boxed{g = 4}$$

Scanned by CamScanner

Step 3: Now select x.

Let $\boxed{x = 2}$ $(\because 0 < x < q)$

$\longrightarrow$ private key

Step 4: Calculate y

$y = g^x \bmod p$

$y = 4^2 \bmod 7$

$y = 16 \bmod 7$

$\boxed{y = 2}$

$\longrightarrow$ Public key

Step 5: select K, but K is already given

$\boxed{K = 2}$

Step 6:

$\boxed{\text{Signing}}$

$r = (g^K \bmod p) \bmod q$

$r = (4^2 \bmod 7) \bmod 3$

Scanned by CamScanner

$r = 2 \bmod 3$

$\boxed{r = 2}$

Step 7: Calculate s:

$s = [k^{-1}(H(m) + x r)] \bmod q$

$= [2^{-1}(3 + 4)] \bmod 3$

$s = 2^{-1}(7) \bmod 3$

$s = 2(7) \bmod 3$

$s = 14 \bmod 3$

$\boxed{s = 2}$

Verifying

Step 8:

$w = (s')^{-1} \bmod q$

$w = 2^{-1} \bmod 3$

$w = 2 \bmod 3$

$\boxed{w = 2}$

Scanned by CamScanner

step 9 : Calculate u_1 and u_2

$$u_1 = [3(2)] \bmod 3$$

$$u_1 = [6 \bmod 3]$$

$$\boxed{u_1 = 0}$$

$$u_2 = (r')w \bmod q$$

$$= 2(2) \bmod 3$$

$$u_2 = 4 \bmod 3$$

$$\boxed{u_2 = 1}$$

step 10 : calculate V

$$V = [(g^{u_1} y^{u_2}) \bmod P] \bmod q$$

$$V = [(4^0 * 2^1) \bmod 7] \bmod 3$$

$$= [2 \bmod 7] \bmod 3$$

$$= (2) \bmod 3$$

$$\boxed{V = 2}$$

Scanned by CamScanner

Step 11 : Compare V and r

We can see $\boxed{V = r = 2}$

Scanned by CamScanner

Key Management

* Key Management

→ One of the major roles of public key encryption has been to address the problem of key distribution.

→ There are actually two distinct aspects to the use of public-key cryptography.

- The distribution of public keys
- The use of public-key encryption to distribute secret key.

* Distribution of Public keys

1) Public Announcement
2) public available directory
3) Public-key authority
4) Public-key Certificates

* Public Announcement of Public keys

→ The Point of public key encryption is that the public key is public.

→ Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to

Scanned by CamScanner

any other participant or broadcast the key to the community at large.

→ For example, because of the growing popularity of PGP (pretty good privacy), which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

Disadvantage

→ Anyone can forge such a public announcement.

* Publicly Available Directory

→ A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.

→ Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

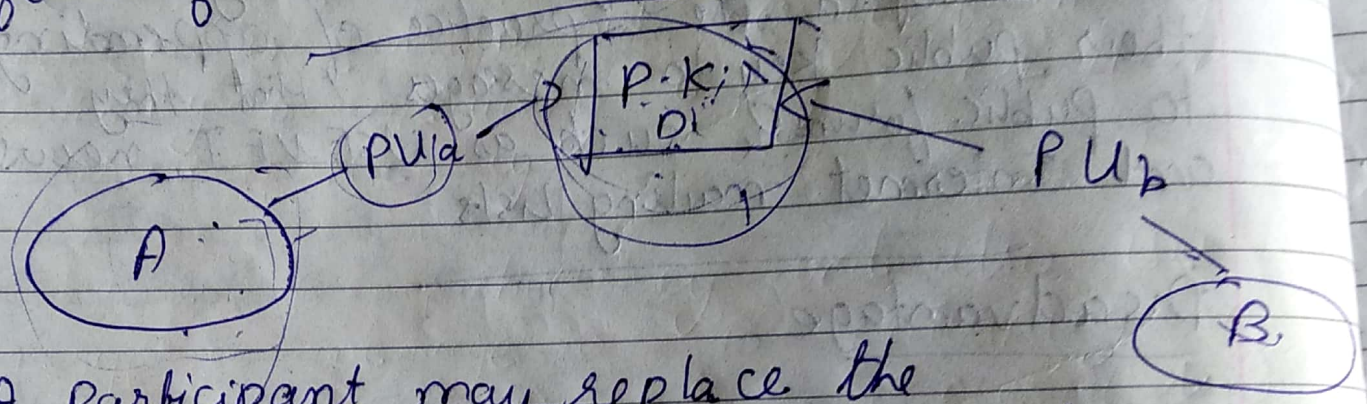
→ Such a scheme would include following elements

Scanned by CamScanner

$\boxed{Prd}$ $\leftarrow \{ \frac{Name}{A} , \frac{P}{Pra} \}$

1) The authority maintains a directory with a {name, public key} entry for each participant.

2) Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.



3) A participant may replace the existing key with a new one at any time.

4) Participants could also access the directory electronically.

Disadvantage

→ If an adversary succeeds in obtaining priv key of the directory authority, the adversary could pass out wrong public keys and subsequently eavesdrop on message.

→ Also attacker may tamper records of authori

* Public - Key Authority

-> The scenario assumes that a central authority maintains a dynamic directory of public keys of all participants.

-> In addition, each participant reliably knows a public key for the authority, with only the authority knowing its private key.

-> The following steps occur:

1) A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2) The authority responds with a message that is encrypted using the authority's private key PRauth.

-> Thus, A is able to decrypt the message using the authority's public key.

-> Therefore, A is assured that the message originated with the authority.

-> The message includes following:

- B's public key, PU_b, which A can use to

Scanned by CamScanner

encrypt messages destined for B.

- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception.

- the original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key.

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

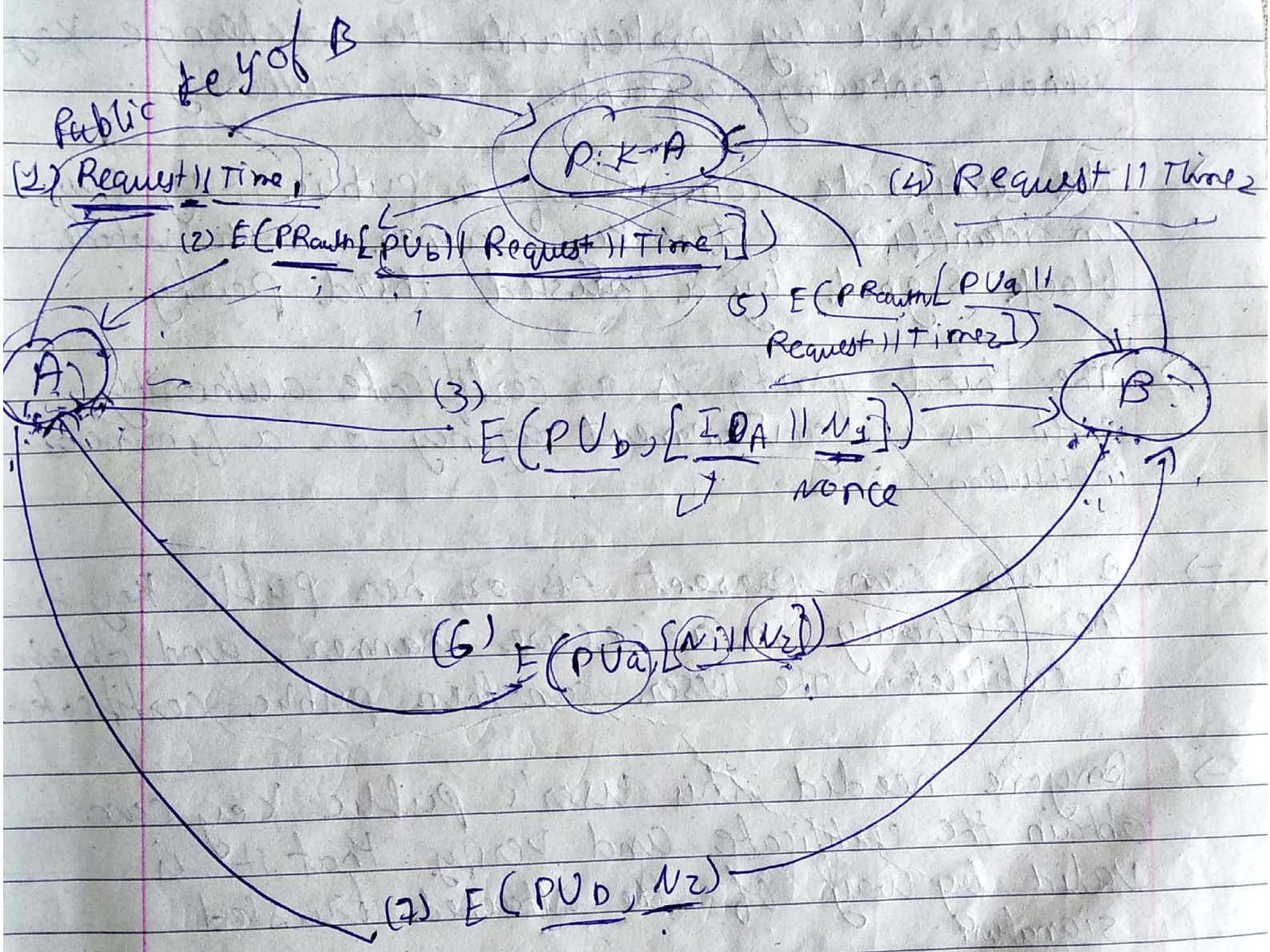
4, 5. B retrieves A's public key from the author in the same manner as A retrieved

6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2).

→ Because only B could have decrypted messa the presence of N_1 in message assures A that the correspondent is B.

Scanned by CamScanner

7. A returns N_2, encrypted using B's Public Key, to assure B that its correspondent is A.



Disadvantage: Here the public-key authority could be somewhat of a bottleneck.

→ As a user must appeal to P.K.A for a public key for every other user to contact.

→ Also it is vulnerable to tampering.

* Public key Certificates

-> Another approach is to use certificates that can be used by participants to exchange keys without contacting a public-key authority.

-> The certificate consists of a public key plus an identifier of the key owner, with whole block signed by a trusted third party.

-> The third party is a certificate authority, such as a government agency or a financial institution.

-> A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then public certificate

-> Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.

Following requirements can be placed:

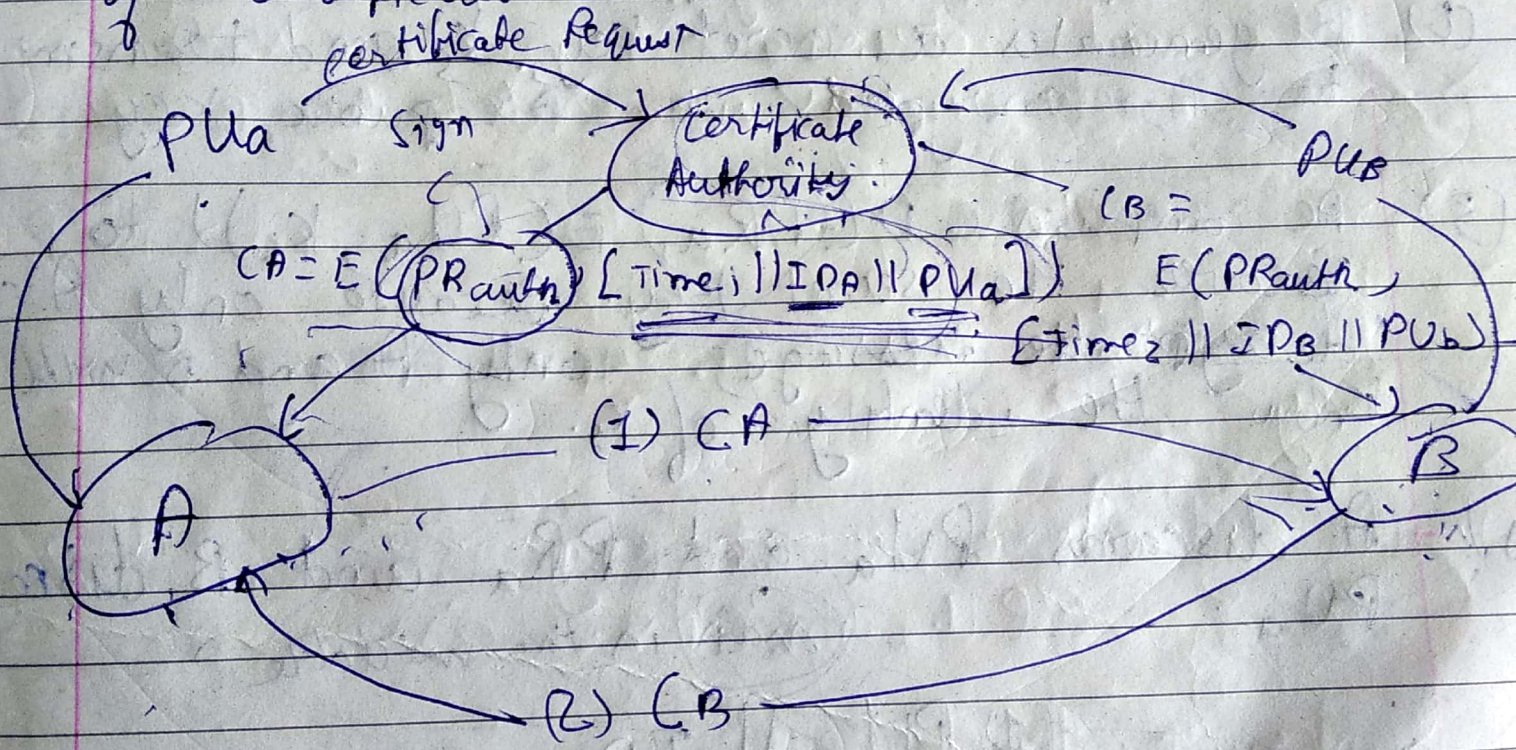
Any participant can read a certificate to determine the name and public key of the certificate's owner.

Scanned by CamScanner

2) Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.

(3) Only the certificate authority can create and update certificates.

(4) Any participant can verify the currency of certificate.



→ The recipient uses the authority's public key, PUauth, to decrypt the certificate.

$$D(PU_{auth}, CA) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a]))$$

$$= (T \| ID_A \| PU_a)$$

Scanned by CamScanner

* Distribution of Secret Keys Using Public-key Cryptography

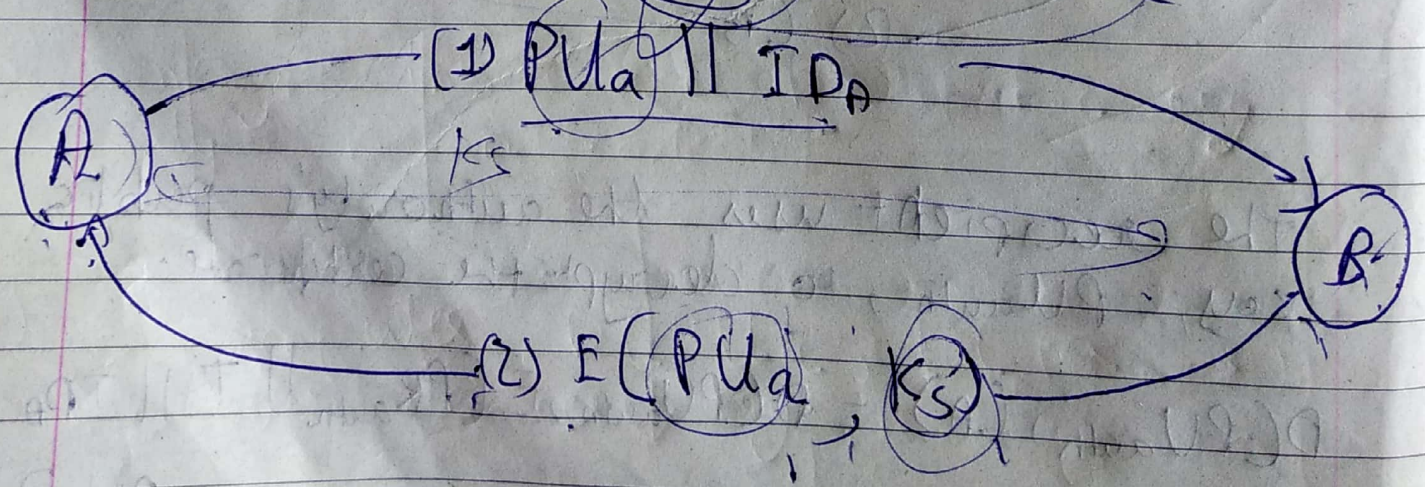
→ (1) A generates a Public/Private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A.

(2) B generates a secret key, K_s and transmits it to A, encrypted with A's public key.

(3) A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.

(4) A discards PU_a and PR_a and B discards PU_a.

man in the middle



→ A and B can now securely communicate using conventional encryption and session key K_s.

Scanned by CamScanner

→ Disadvantage :

→ This sheme does not work for man-in-the middle attack.

(1) A generates a public/Private key pair {PUa, PRa} and transmits a message intended for B consisting of PUa and an identifier of A, IDA.

(2) E intercepts the message, creates its own public/private key pair {PUe, PRe} and transmits PUe || IDA to B.

(3) B generates a secret key, Ks, and transmits E(PUe, Ks).

(4) E intercepts the message, and learns Ks by computing D(PRe, E(PUe, Ks)).

(5) E transmits E(PUa, Ks) to A.

→ The result is that both A and B know Ks and are unaware that Ks has also been revealed to E.

Scanned by CamScanner

$E(PU_b, ID_A \| N_1)$

* Secret Key Distribution with Confidentiality and Authentication.

→ (1) A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely

(2) B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2).

→ the presence of N_1 assures sender is B.

3) A returns N_2, encrypted using B's pub key.

(4) A selects a secret key (K_s) and sends $M = E(PU_b, E(PR_a, K_s))$ to B.

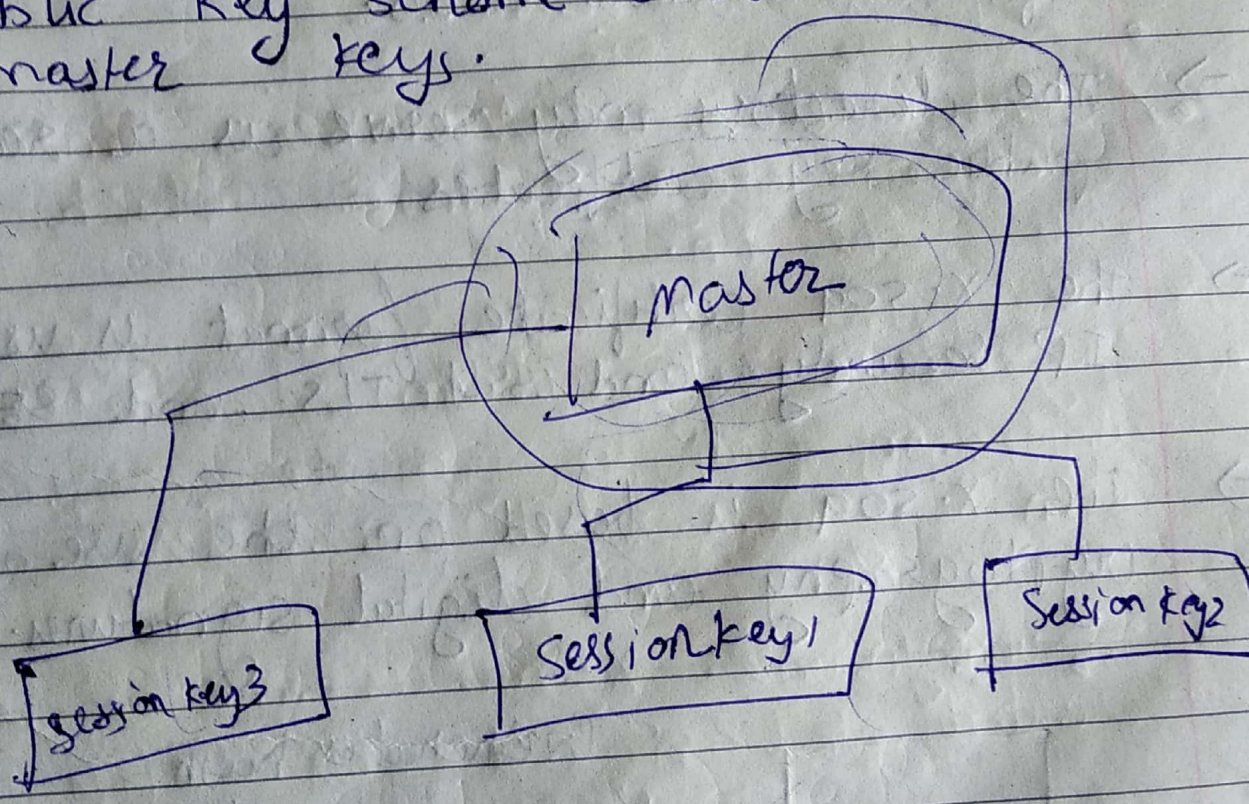
→ Encrypt this with B's public key so that only B can read.

5) B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

Scanned by CamScanner

A Hybrid Scheme

- This scheme retains the use of a KDC that shares a secret master key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys.



Scanned by CamScanner

* X.509 Authentication Service

→ ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service.

→ The directory is, in effect, a server or distrib set of servers that maintains a database of information about users.

→ The directory may serve as a repository of public-key certificates.

→ The X.509 certificate format is used S/MIME, IP Security, and SSL/TLS and SET.

→ The X.509 is based on the use of public-key cryptography and digital signatures.

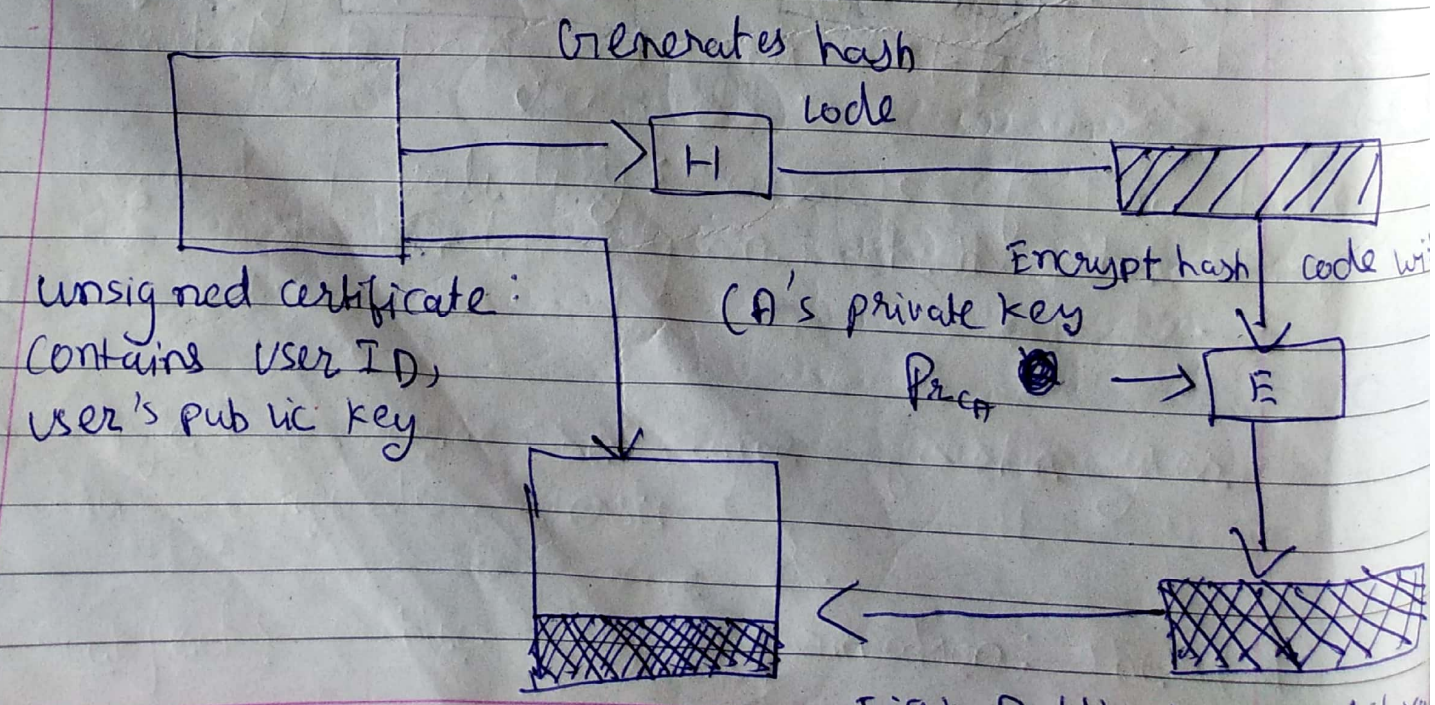


Fig: Public-key c...

→ The heart of X.509 scheme is the public-key certificates associated with each user.

→

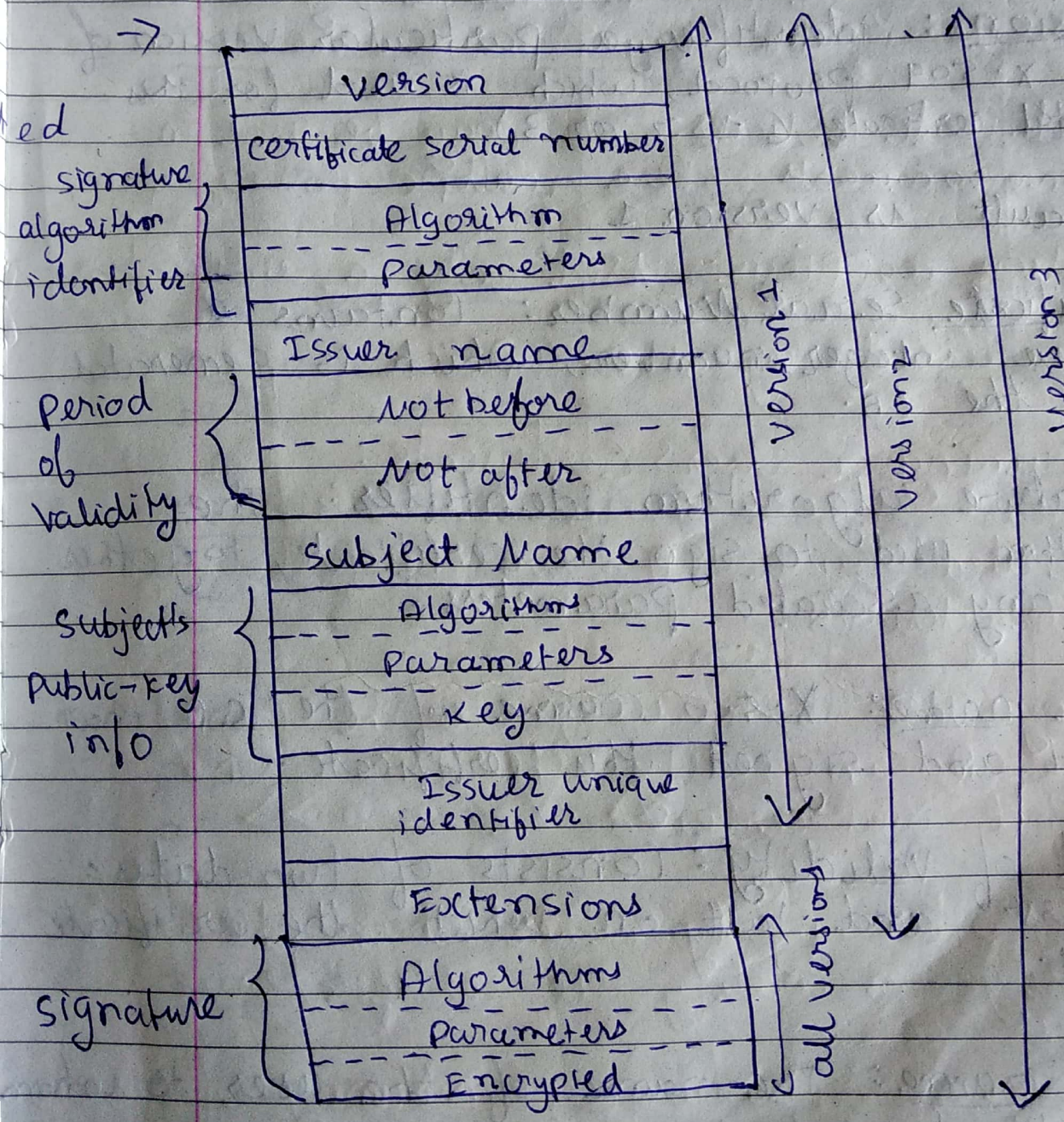


Figure: X-509 Certificate

Use

Above figure shows the general format of a certificate which includes the following elements

- Version: identifies a particular version of the X.509 protocol, which is used for this digital certicate (1, 2 or 3)

→ Default is version 1

- Certificate Serial Number: Contains a unique integer number, which is generated by the CA.

- Signature algorithm identifier: The algorithm used to sign the certificate, together with any associated parameters.

- Issuer name: X.500 name of the CA that created and signed this certificate.

Period of validity: Consists of two dates: the first and last on which the certificate is valid.

Subject name: The name of the user to which this certificate refers.

Subject's Public-Key information: The public key

Scanned by CamScanner

of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

- Issuer unique identifier: Helps identify a CA uniquely if two or more CAs have used the same Issuer Name over time.

- Subject Unique Identifier: Helps identify a subject uniquely if two or more subjects have used the same Subject Name over time.

- Extensions: A set of one or more extension fields. Extensions were added in version 3 are are as follows:

- ~~Auth~~ Authority Key identifier: Identifies the public key to be used to verify the signature on this certificate or CRL. Enables distinct keys of the same CA to be differentiated.

- Subject key identifier: Identifies the public key being certified. A subject may have multiple key pairs for different purposes.

- Key usage: Indicates a restriction imposed as to the purposes for which, and the policies under

Scanned by CamScanner

the certified public key may be used.

- Private - Key usage Period: Indicates the period of use of the private key corresponding to the public key.

- Certificate Policies: It may be used in environments where multiple policies apply

- Policy mappings: Used only in certificates for CA's issued by other CAs.

* Obtaining a User's Certificate: User certificates generated by a CA have the following characteristics:

  - Any user with access to the public key of the CA can verify the user public key that was certified.

  - No other party other than the certificate authority can modify the certificate.

→ If there is a large community of users it may not be practical for all users to subscribe to the same CA.

Scanned by CamScanner

-> Because it is the CA that signs certificates each participating user must have a copy of the CA's own public key to verify signatures.

-> This public key must be provided to each user in an absolutely secure way so that the user has confidance in the associated certificates.

-> Thus with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

-> Now suppose that A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2.

-> If A does not securely know the public key of X_2, then B's certificate, issued by X_2, is useless to A.

or

-> A can read B's certificate, but A cannot verify the signature.

-> However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

Scanned by CamScanner

(1) A obtains from the directory, the certificate of X_2 signed X_1. Because A securely knows X_1's public key, A can obtain X_2's public key from its certificate and verify it by means of X_1's signature on the certificate.

(2) A then goes back to the directory and obtains the certificate of B signed by X_2. Because A now has a trusted copy of X_2's public key. A can verify the signature and securely obtain B's public key.

→ A has used a chain of certificates to obtain B's public key.

In the notation of X.509, this chain is expressed as

$$X_1 \langle\langle X_2 \rangle\rangle \; X_2 \langle\langle B \rangle\rangle$$

In the same fashion, B can obtain A public key with the reverse chain:

$$X_2 \langle\langle X_1 \rangle\rangle \; X_1 \langle\langle A \rangle\rangle$$

Scanned by CamScanner

-> Thus the directory entry for each CA includes two types of certificates:

(1) Forward certificate : Certificates of X generated by other CAs.

(2) Reverse certificates : Certificates generated by X that are the certificates of other c

* Revocation of Certificates : Each certific includes a period of validity, much like a credit card.

-> Typically, a new certificate is issued just before the expiration of the old one.

-> In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons

(1) The user's private key is assumed to be compromised.

(2) The user is no longer certified by

(3) The CA's certificate is assumed to be compromised

attacker alters that message and doesn't

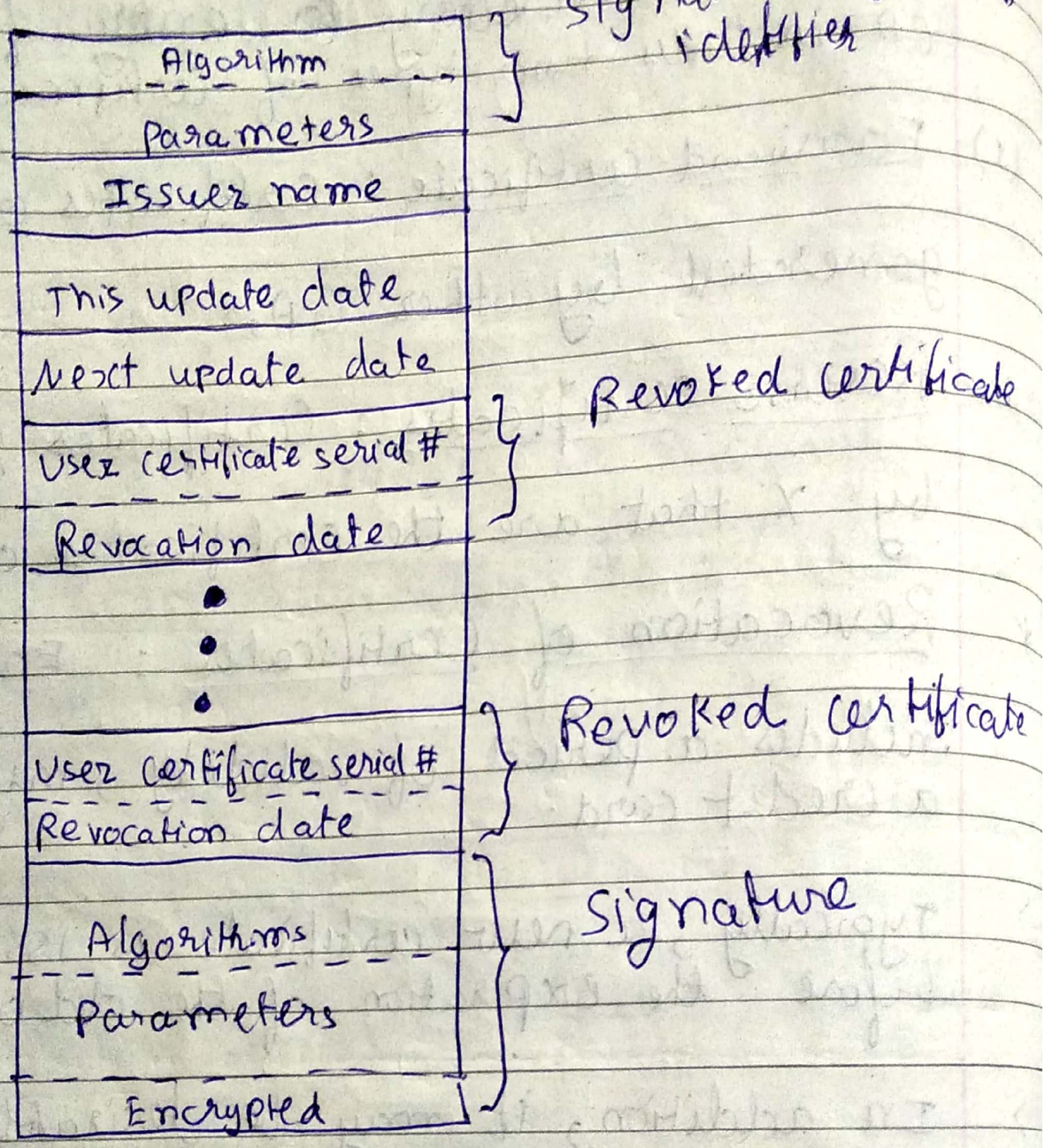


Fig: Certificate revocation list

→ Each CRL posted to the directory is signed by the issuer and includes the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate.

Scanned by CamScanner

Public - Key Infrastructure

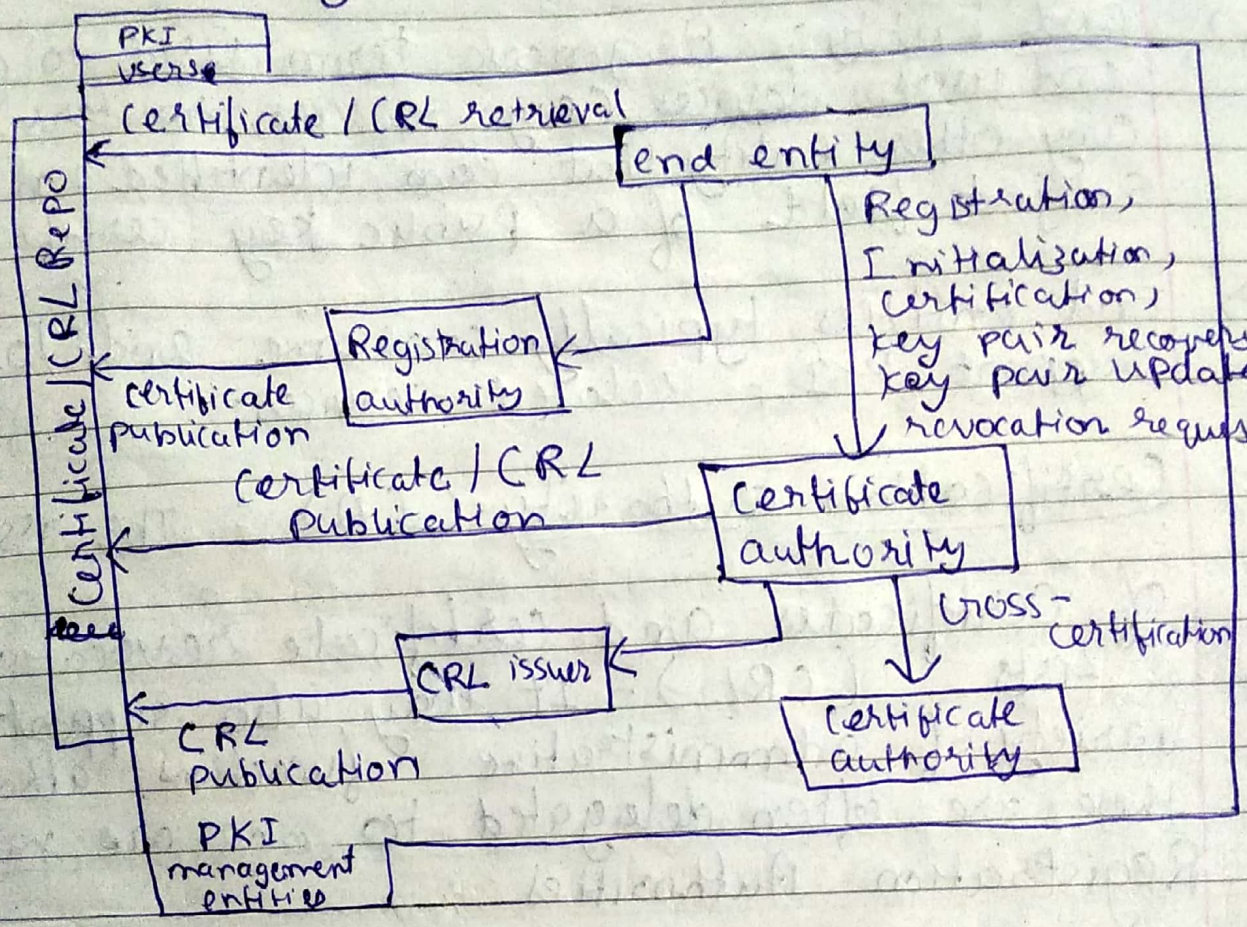


Fig: PKIX Architectural Model

- The principal objective for developing PK is to enable secure, convenient, and efficient acquisition of public keys.

- PKI is the set of hardware, s/w, peopl policies and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

Scanned by CamScanner

* The key elements of PKIX model are:

(1) End Entity: A generic term used to denote end users, devices (e.g. servers, routers), or any other entity that can identified in the subject field of a public key certificate.

→ End entities typically consume and/or support PKI - related services.

2) Certification authority (CA): The issuer of certificates and certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

) Registration Authority (RA): An optional component that can assume a number of administrative functions from the CA.

The RA is often associated with the End Entity Registration process, but can assist in a number of other areas as well.

CRL Issuer: An optional component that a CA can delegate to publish CRLs.

Scanned by CamScanner

5) Repository : A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by End Entities.

PKIX Management Functions

Registration : This is the process whereby a user first makes itself known to a CA prior to that CA issuing a certificate or certificates. It usually involves some offline or online procedure for mutual authentication.

Initialization : The client needs to be securely initialized with the public key and other assured information of trusted CA's.

Certification : This is a process in which a CA issues a certificate for a user's public key, and returns that certificate to the user's client system and/or posts that certificate in a repository.

Scanned by CamScanner

(3) Key pair Recovery : Key pairs are used to generate digital signature, verification, encryption and decryption.

-> When normal access is not possible, there should be some mechanism to the keys.

-> Hence Backup is taken.

(4) Key pair update : Key pairs need to be updated regularly with new certificate issued.

-> Updation is required due to

1) Certificate expires
2) Certificate is revoked
(3) compromise of private Key.

(5) Revocation Request :

An authorized person advices of abnormal activities to CA to revoke certificate. Revoke because:

(1) compromized private key
(2) changes in affiliation
(3) Change in name.

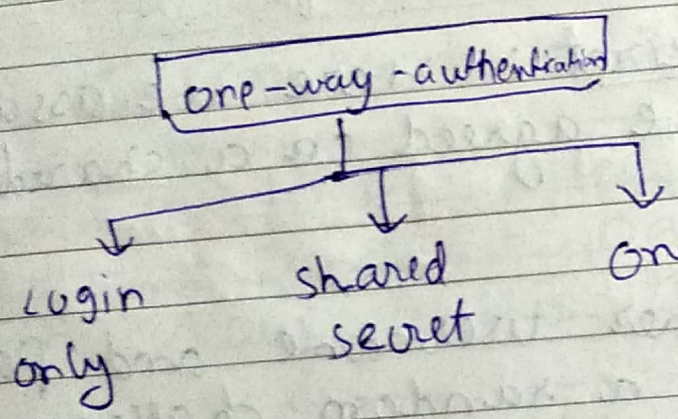
Scanned by CamScanner

Cross Certification

→ Two CA's exchange information used in establish a cross certificate.

Authentication

One - Way Authentication

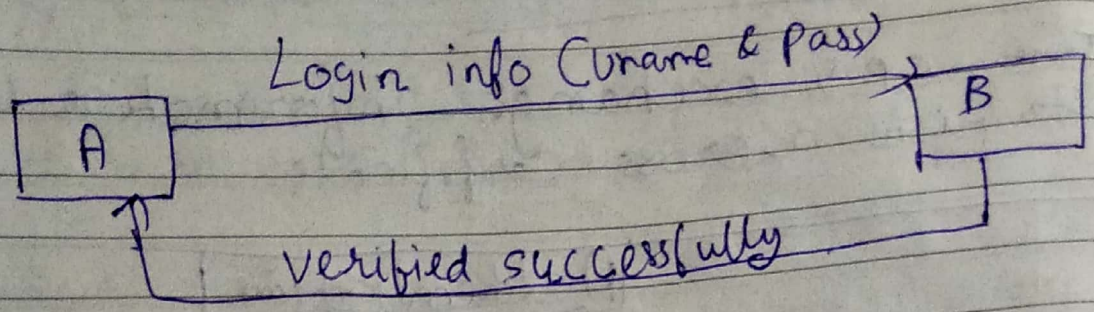


Login only :

User A sends her user-name and password in the plaintext form to other user B.

→ B verifies the user name and password. If the user name and password are correct, communication starts happening between A and B. No further encryption or integrity checks are performed.

Scanned by CamScanner



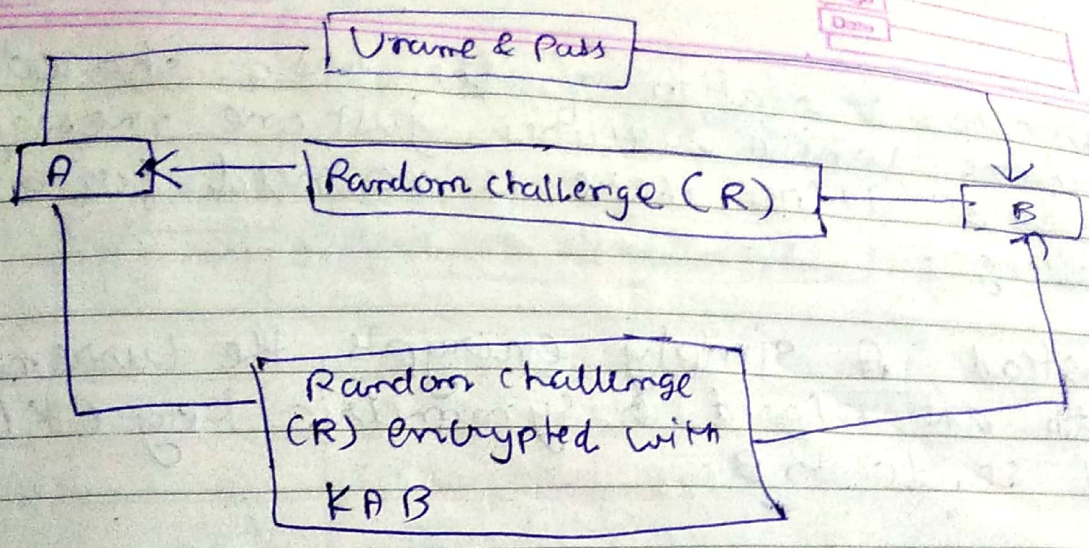
→ This approach is ineffective.

* <u>Shared Secret</u>: Here we assume that A and B have agreed on a shared symmetric key KAB.

1) A sends her username and password to B.
2) B creates a random challenge R and sends it to A.

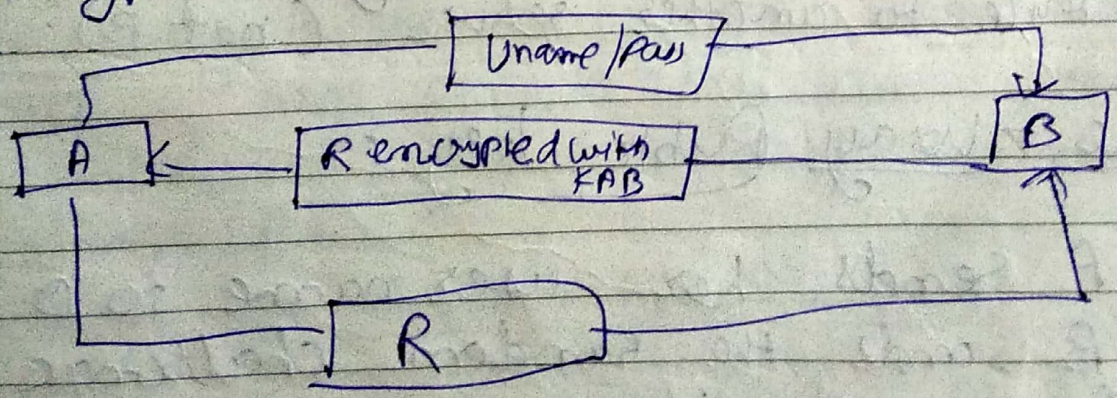
3) A encrypts the random challenge (R) with the shared symmetric key between A and B (KAB) and sends the encrypted R to B. B also encrypts the original random challenge (R) with the same shared symmetric key (KAB). If this encrypted challenge matches with the one sent by A, B considers A to be authentic.

Scanned by CamScanner



→ Problem: C can send an old random challenge (R) to A. After A encrypts R with KAB and sends it back to C, C simply ignores it. Now A thinks that C is B.

→ A variation here can be instead of sending R as it is to A, B first encrypts it with shared symmetric key (KAB) and sends the encrypted random challenge to A.

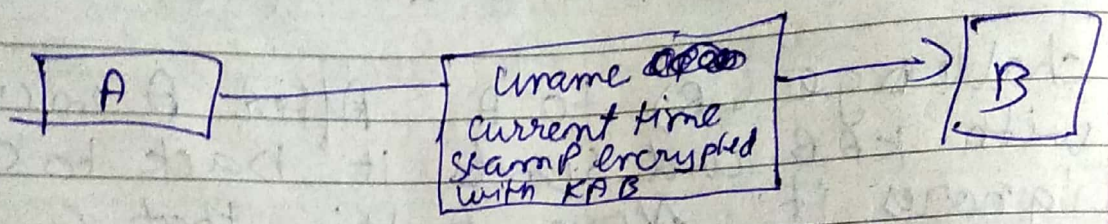


Scanned by CamScanner

-> Another variation of the basic shared secret scheme, which requires just one message from A to B. Here, there is no need for a random challenge.

-> Instead, A simply encrypts the current time with the shared symmetric key (K_{AB}) and sends to B.

-> B decrypts and authenticates A.

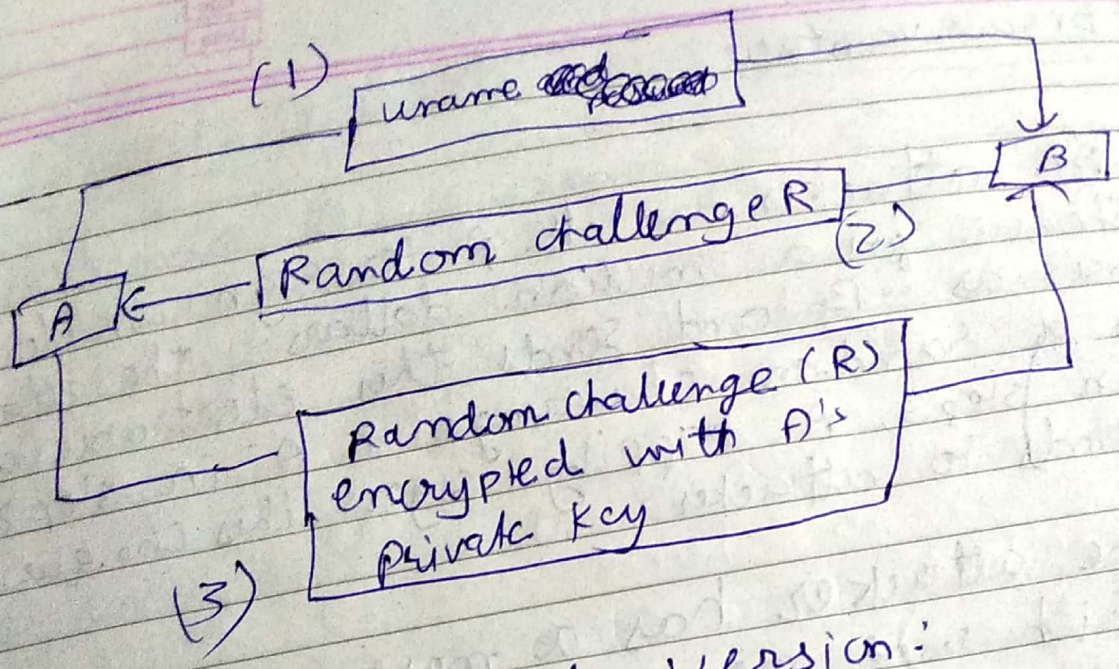


-> Disadvantage: When A sends the original message to B, the attacker C simply copies it and then uses it to try and impersonate as Alice to another server (not B).

One-Way Public key:

(1) A sends her user name to B
(2) B sends the random challenge to A.
(3) A encrypts the random challenge, and matches it with the original random challenge (R).

Scanned by CamScanner

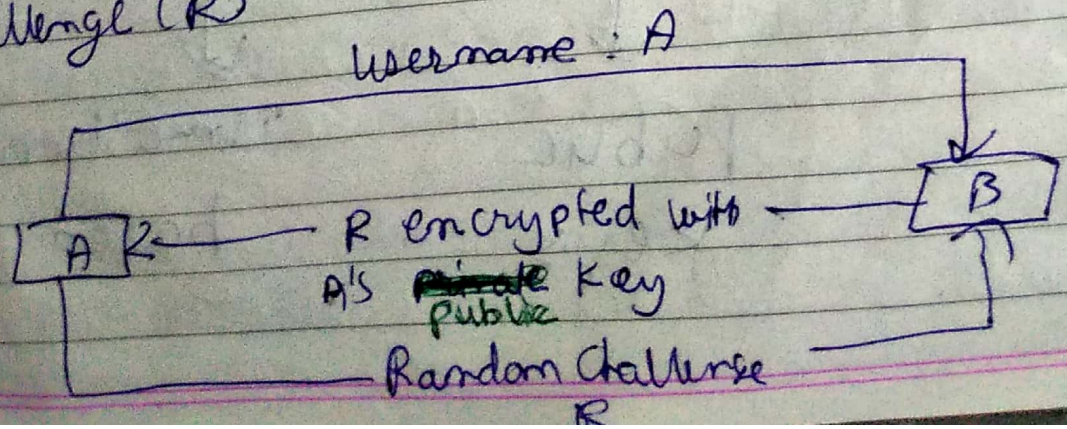


→ slightly modified version:

(1) A sends her username to B

(2) B creates the random challenge (R) and encrypts it with the public key of A. B sends this encrypted random challenge to A.

(3) A decrypts the encrypted random challenge with her private key and sends it to B. B matches it with the original random challenge (R).



Scanned by CamScanner

Disadvantage:

(1) An attacker wants A to sign an electronic cheque for a million dollars. The attacker poses as B and sends this electronic cheque as a random challenge to A in step 2. In step 3, A happily signs this cheque and sends to attacker.

(2) The attacker has a message from the past, which was meant for A. Hence, the message was encrypted with A's public key.

→ Now, the attacker takes this encrypted message and sends as the random challenge in step 2. Poor A thinks that she is signing some random challenger with her private key. Instead she is decrypting a message that was encrypted with her public key.

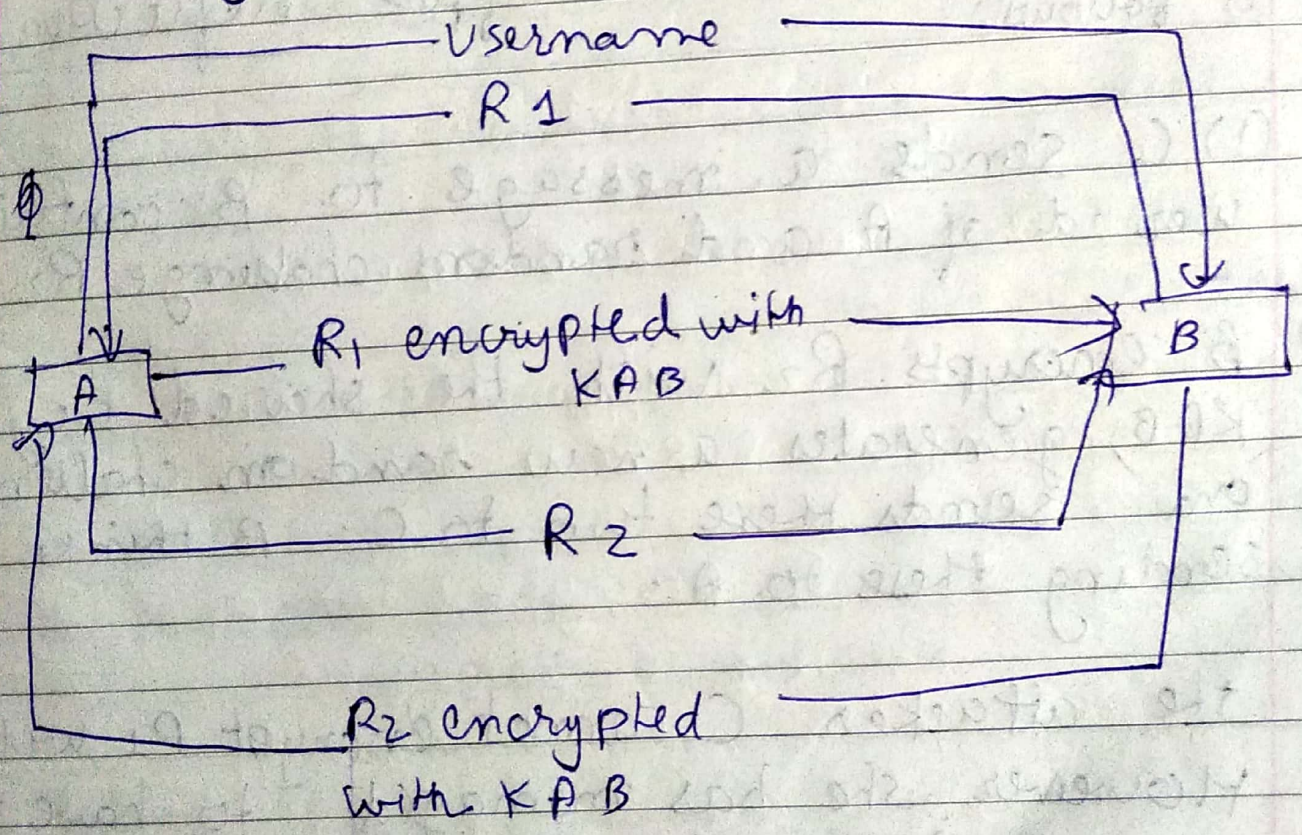
Mutual Authentication

- Shared secret
- Public keys
- Timestamps-based

Scanned by CamScanner

* Shared Secret

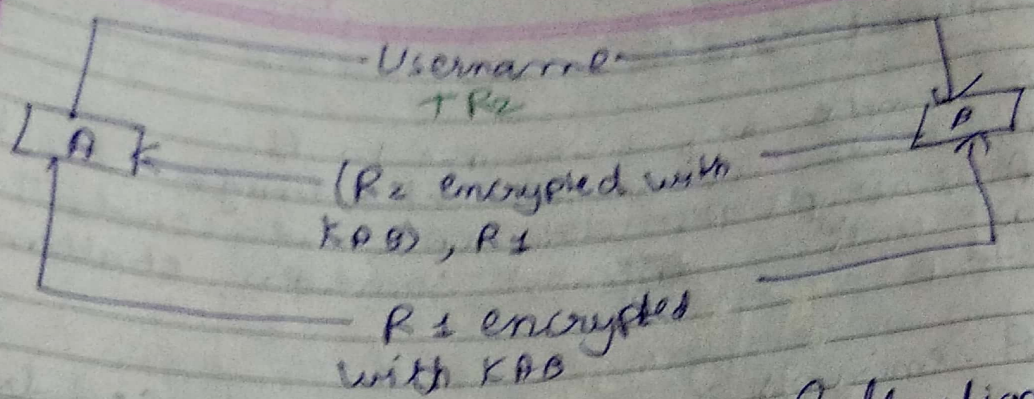
1) A sends her username to B.
2) B sends a random challenge R_1 to A
3) A encrypts R_1 with KAB and sends it to
4) A sends a different random challenge R_2 to B.
5. B encrypts R_2 with KAB and sends it to A



→ Alternative approach to reduce steps:

1) A sends Uname + R_2 to B
2) B encrypts R_2 with KAB and sends R_1 to A
3) A verifies R_2, encrypts R_1 and send to B.

Scanned by CamScanner



→ The above scheme suffers <u>Reflection Attack</u> as follows:

→ (1) C sends a message to B containing user id of A and random challenge R2.

(2) B encrypts R2 with the shared key KAB, generates a new random challenge (R1) and sends these two to C. B thinks he is sending these to A.

→ The attacker C cannot encrypt R1 with KAB. However, she has managed to have B encrypt R2.

Now, the attacker C opens a second session with B, distinct from the first session, which is still active.

C sends a message to B containing user id of

and Random challenge R1.

(2) B encrypts R1 with shared symmetric key KAB, generates a new random challeng (R3) and sends these two to C. B thinks he is sending these to A.

→ The attacker C cannot proceed with this session, since she cannot encrypt the new random chadlenge R3.

→ However, she need not anyway proceed with this session.

→ Instead, she can go back to her first session opened with B earlier.

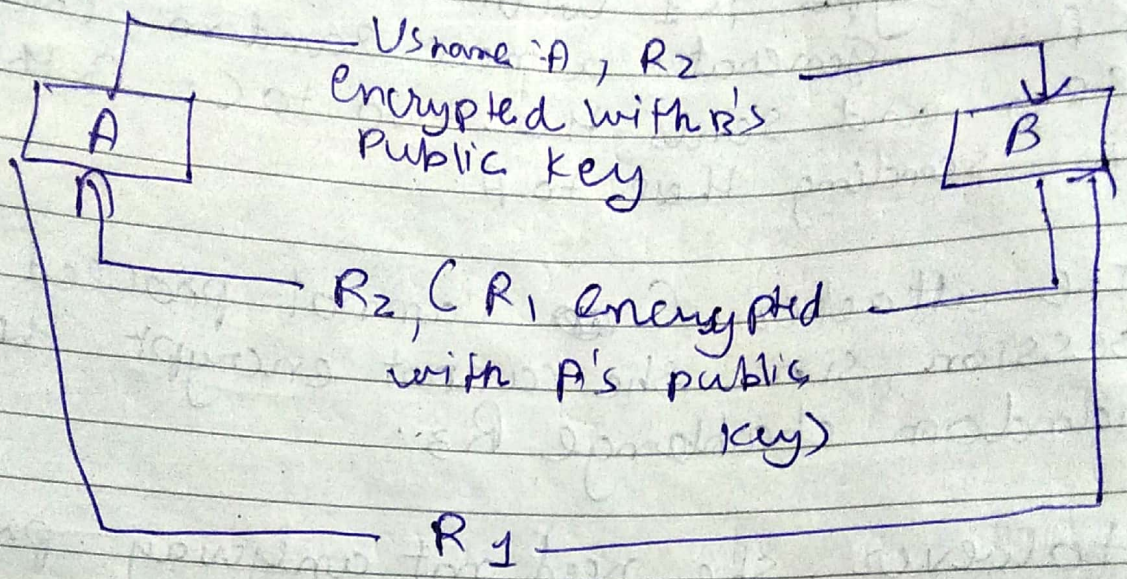
→ Now C has R1 encrypted with KAB thanks to this second session. She sends it to B and completes authentication.

Public keys

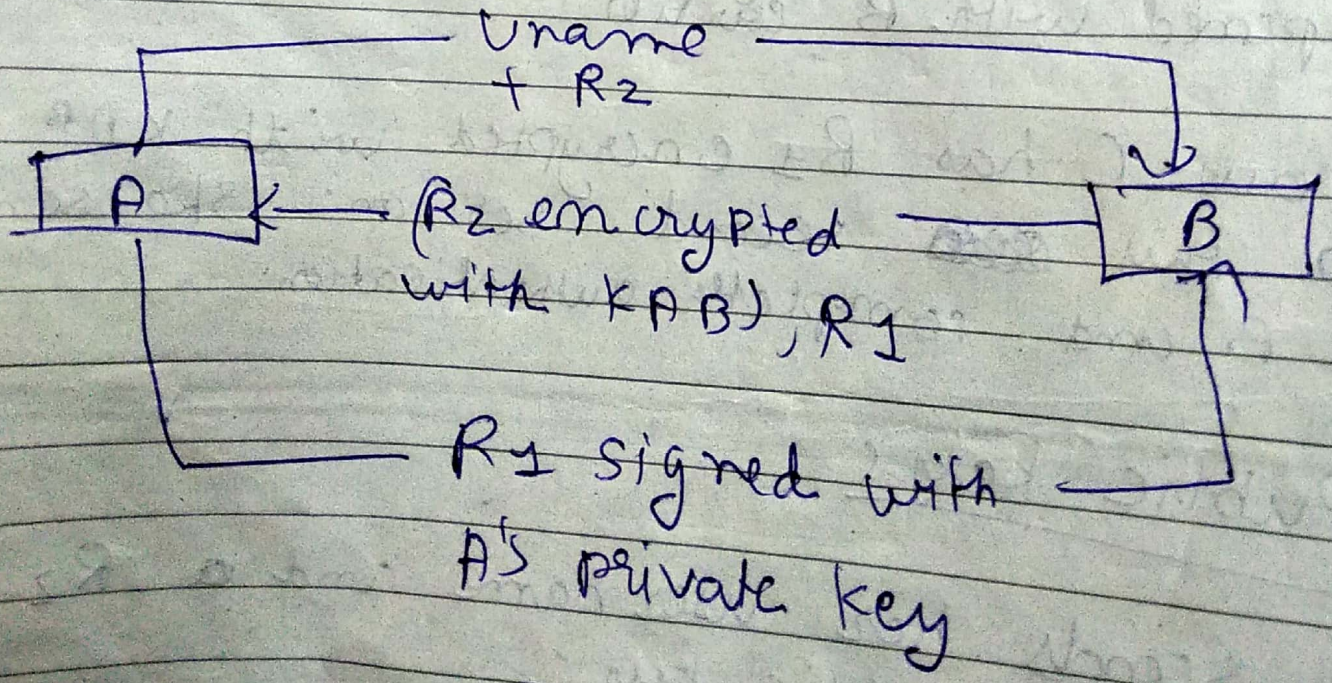
(1) A sends her user name and R2 encrypt with B's public key.

(2) B decrypts the random challenge R2 with his private key. B creates R1 and encrypts with A's public key. B sends these two to A

(3) A decrypts R_1 with her private key and sends it to B. B verifies R_1.



Variation in above scheme



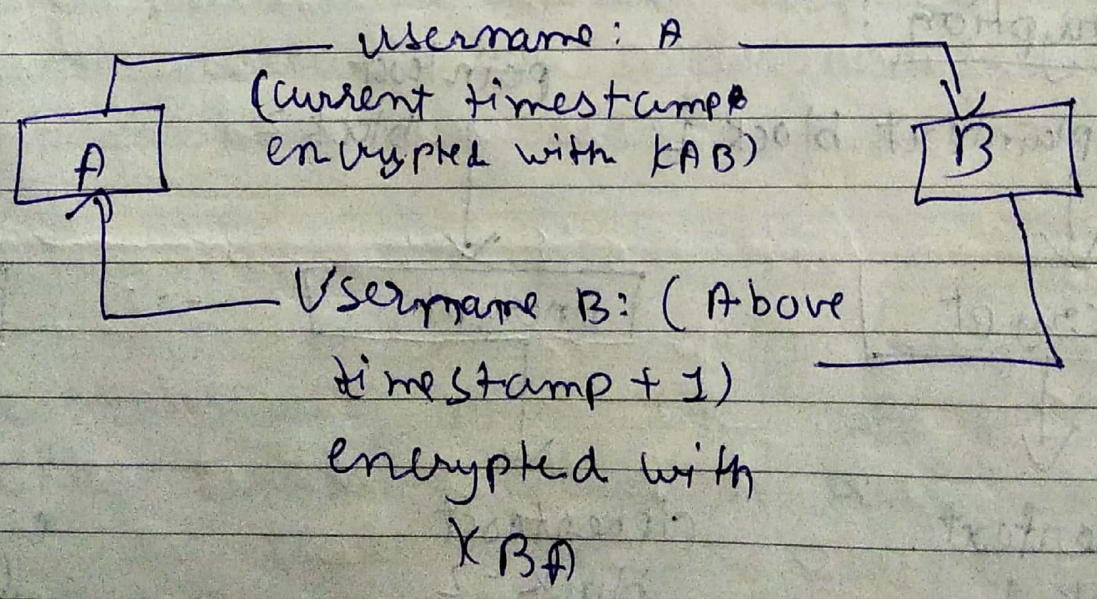
Scanned by CamScanner

Timestamps : We can reduce the mu... authentication process to just two steps by using timestamps, instead of random no's as challenges.

1) A sends her user name and the current timestamp encrypted with a shared symmetric key (KAB) to B.

2) B retrieves the timestamp by decrypting the above block using KAB and adds one to the timestamp.

-) B encrypts the result with (KBA) not (KAB!) and sends it to A along with username.



Scanned by CamScanner

Block cipher modes of operation

Stream

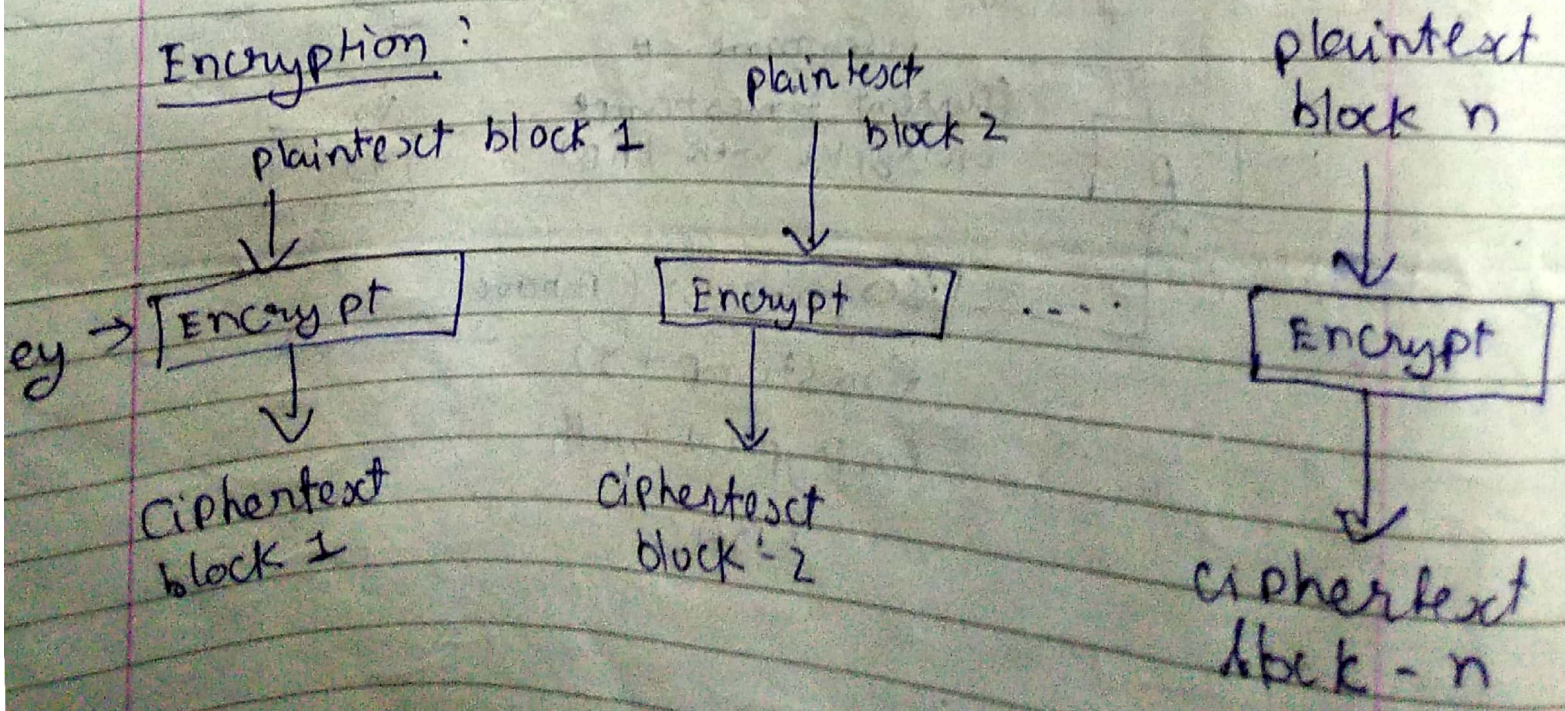
Algorithm mode

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)

(1) <u>Electronic Code Book</u> is the simplest mode of operation. Here, the incoming plain text message is divided into blocks of 64 bits each.

→ Each such block is then encrypted independently of the other blocks.

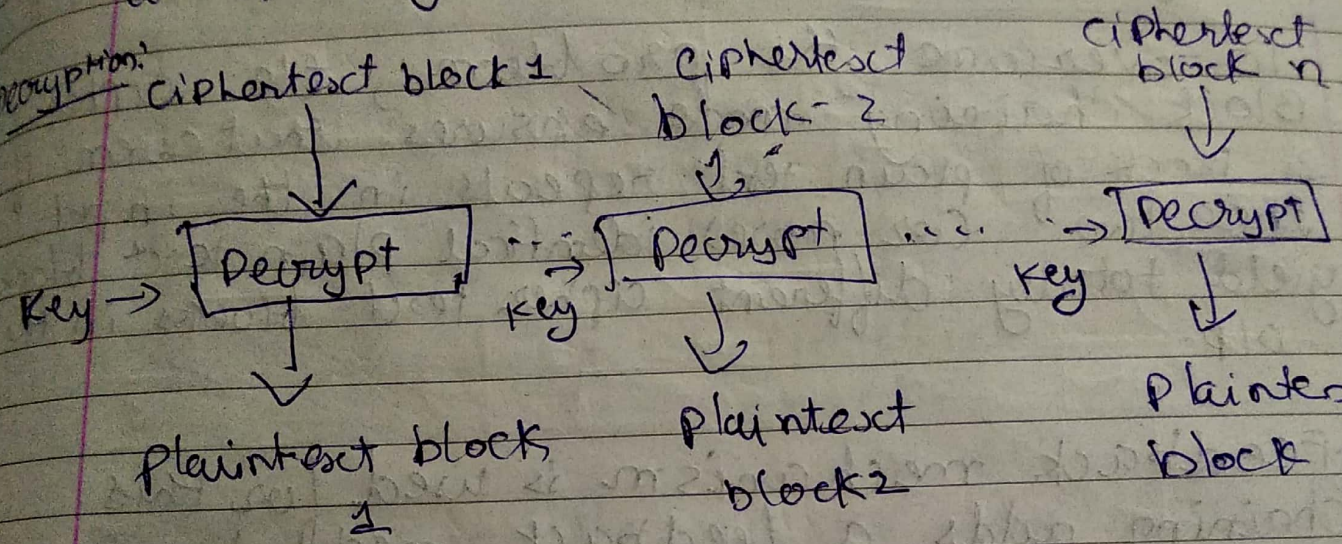
<u>Encryption</u>:



Scanned by CamScanner

(64 bit)

→ At the receiver's end, the incoming data is divided into 64-bit blocks and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block.



* Disadvantage

→ If a plain text blocks repeats in the original message, the corresponding cipher text block will also repeat in the encrypted message.

→ Thus, ECB is suitable only for encrypting small messages, where the scope for repeating the same plaintext block is quite less.

Scanned by CamScanner

2) Cipher Block chaining mode

→ We saw in case of ECB, within a given message (for a given key), a plaintext block always produces the same cipher text block.

→ To overcome this problem, the cipher Block chaining mode ensures that even if a block of plain text repeats in the input, these two (or more) identical plain text blocks yield totally different cipher text blocks in the o/p.

→ Feedback mechanism is used for this. Chaining adds a feedback mechanism to a block cipher.

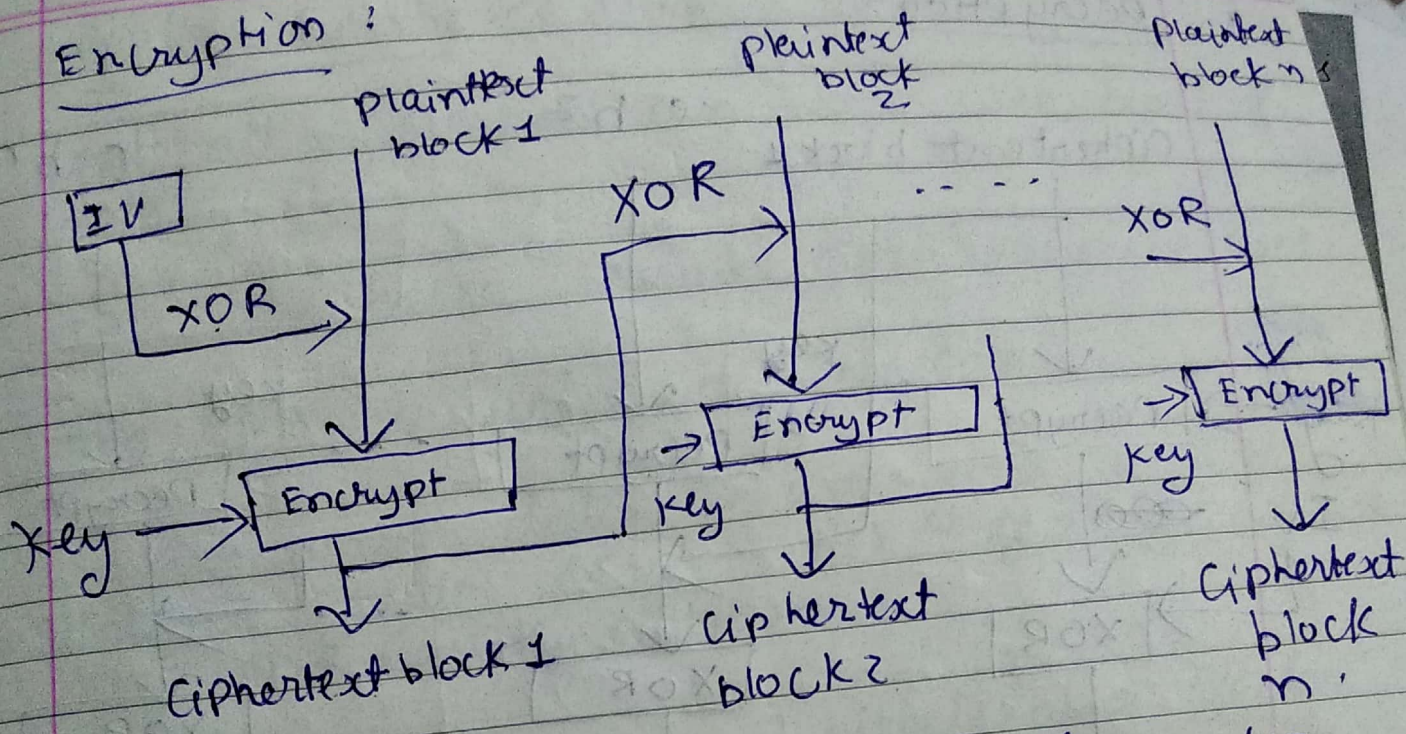
→ In CBC, the results of the encryption of the previous block are fed back into the encryption of the current block.

→ That is, each block is used to modify the encryption of the next block.

→ Thus, each block of ciphertext is dependent on the corresponding current i/p plaintext as well as all the previous plain text block

Scanned by CamScanner

Encryption :



-> As shown in the figure, the first step receives two inputs: the first block of plain-text and a random block of text called as Initialization vector (IV)

-> IV is randomly generated.

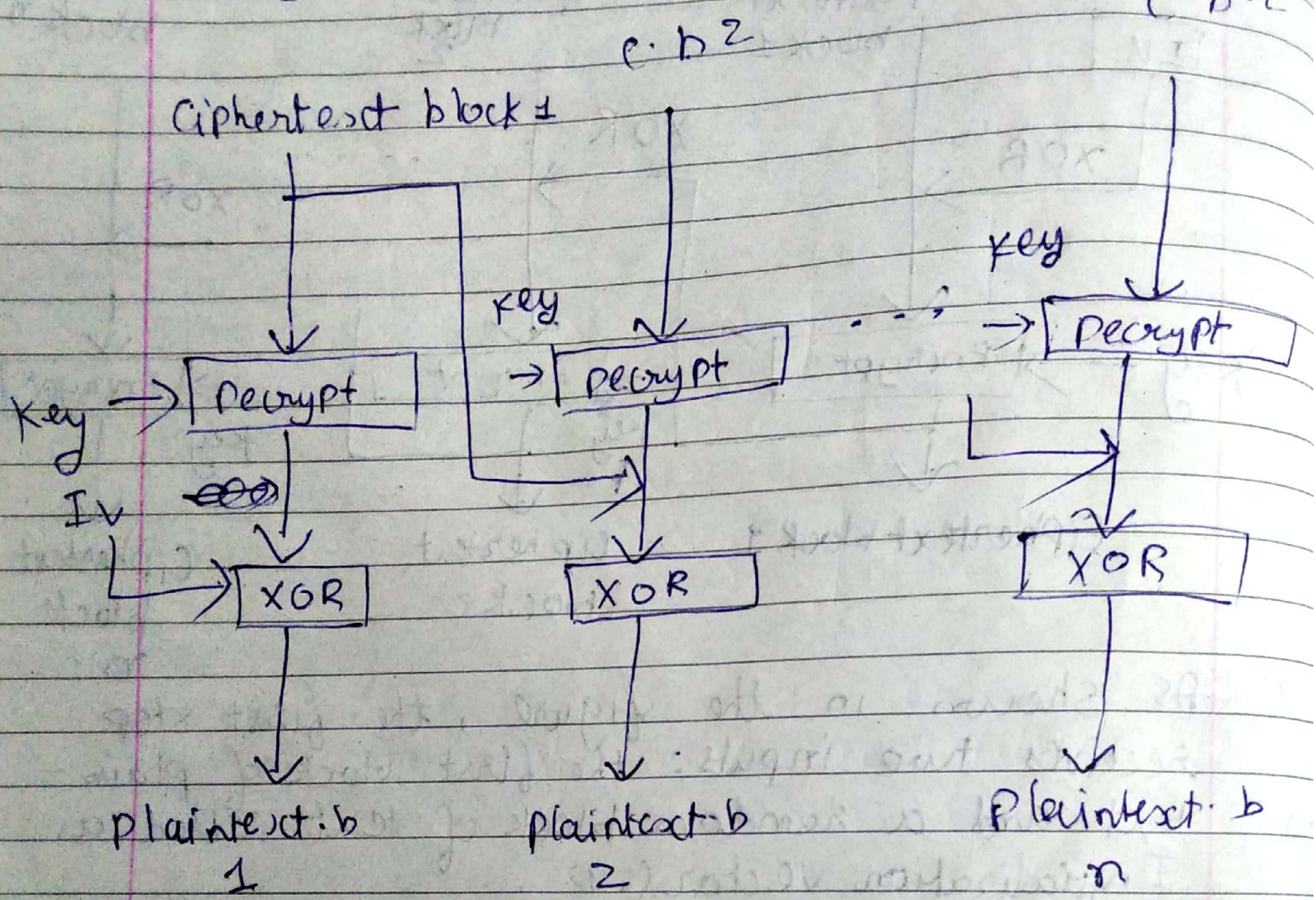
-> The second plaintext block is XORed with the o/p of step 1, i.e the first ciphertext block

-> This process continues for all remaining blocks

-> Same key is used for encryption in each block.

Scanned by CamScanner

* Decryption:



→ For decryption:

(1) The ciphertext block 1 is passed through the decryption algorithm using the same key which was used during the encryption process for all the plain text blocks. the o/p of this step is then XORed with the IV. This yield plaintext 1.

(2) In step 2, ciphertext block 2 is decrypted and its o/p is XORed with ciphertext 1. This process con

(3) Cipher Feedback Mode (CFB):

-> Not all applications can work with block of data.

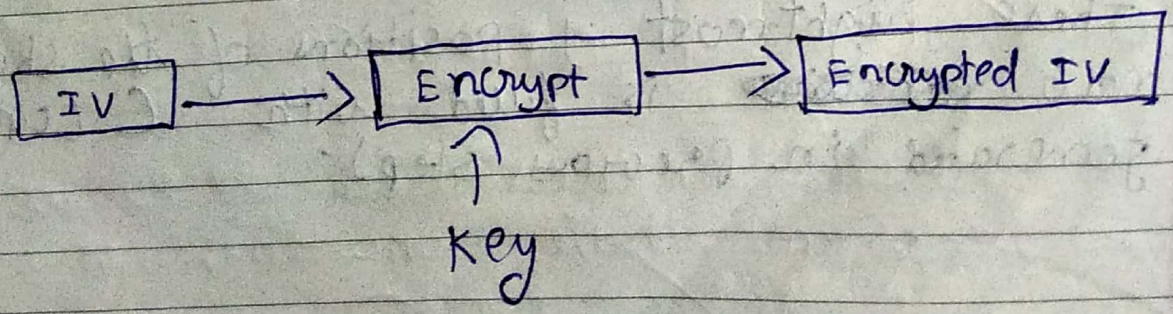
-> In some cases, stream cipher must be used.

-> The CFB is useful in such cases.

-> In this mode, data is encrypted in units that are smaller (e.g 8 bits) than a defined block size (usually 64 bits).

-> Step 1: A 64-bit Initialization vector (IV) is used in the case of CFB. The IV is kept in a shift register.

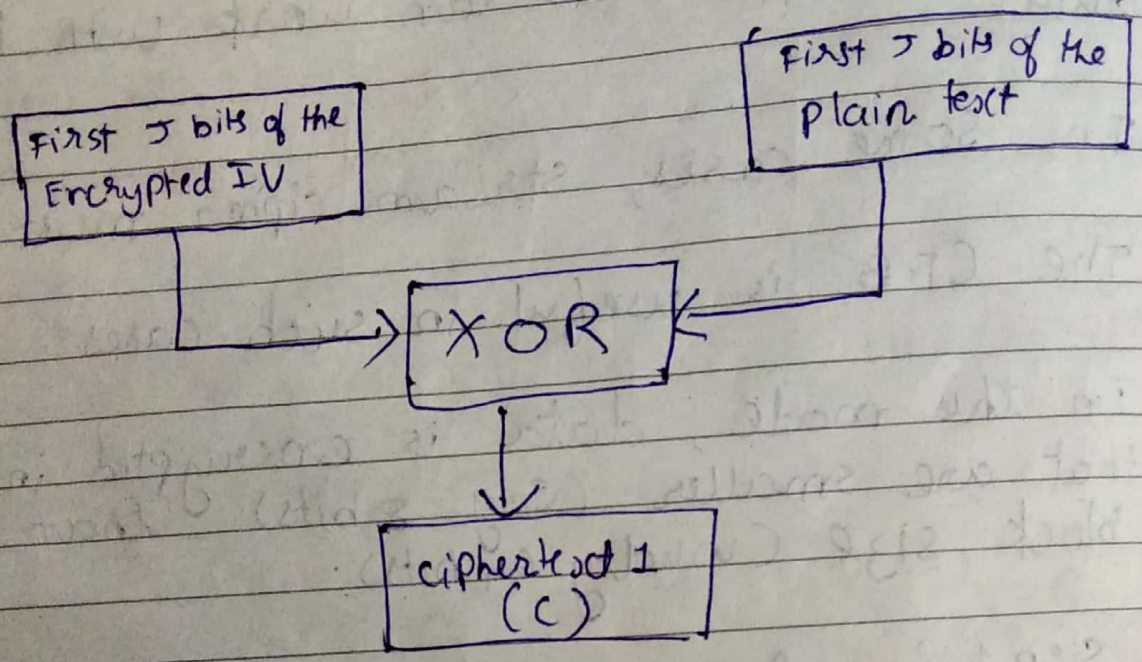
-> It is encrypted in the first step to produce a corresponding 64-bit IV ciphertext.



Step 2: Now, the leftmost j bits of the encrypted IV are XORed with the first j bits of plaintext.

Scanned by CamScanner

→ This produces the first portion of cipher text as shown in fig:



Step3: Now, the bits of IV are shifted left by J positions.

→ Thus the rightmost J positions of the shift register now contain upredictable data.

→ These rightmost j positions of the shift register now is filled with C (ciphertext generated in previous step).

Step4: Now Steps 1 to 3 continue until all the plaintext units are encrypted.

Scanned by CamScanner

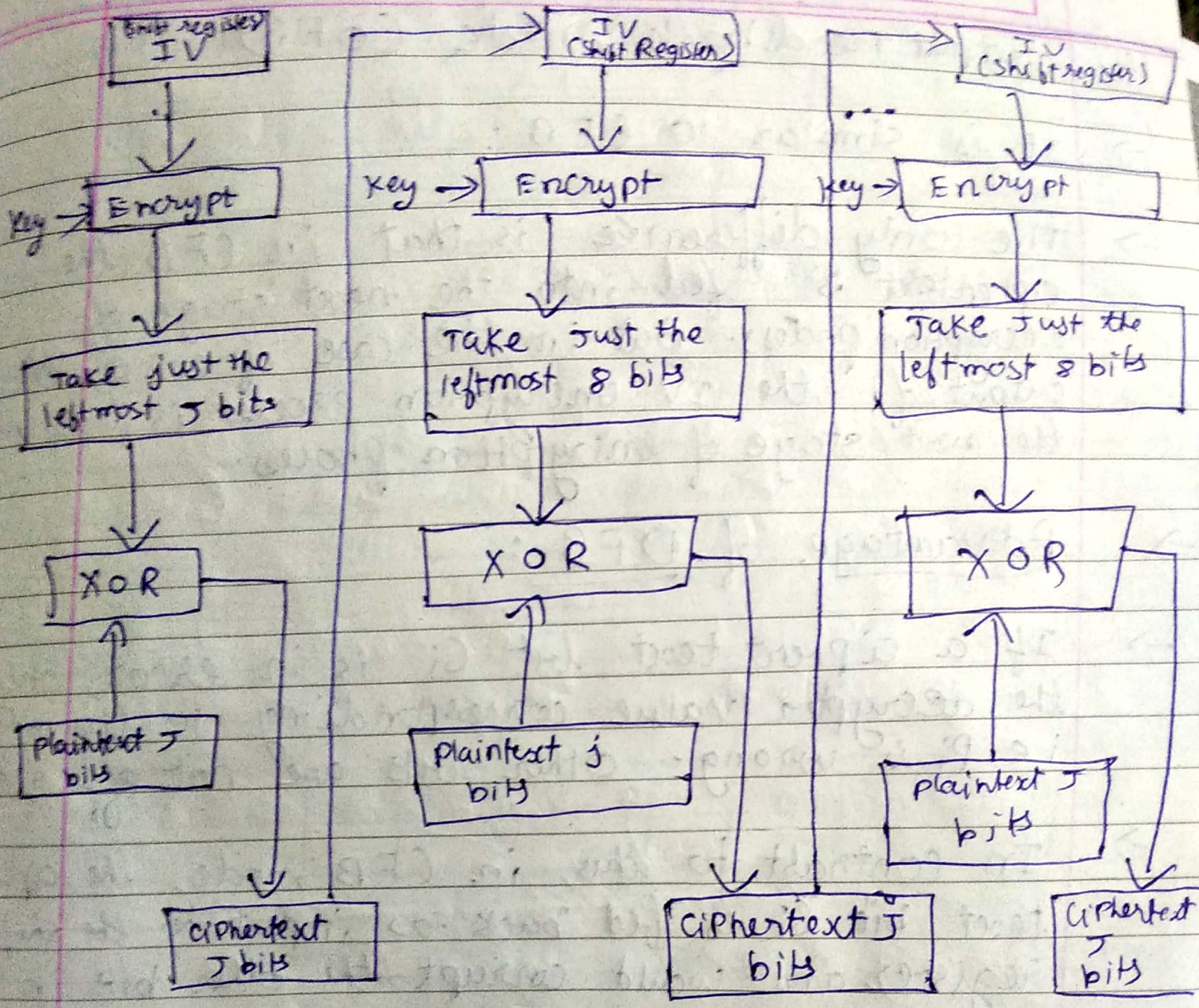


Figure: The overall encryption of CFB

Scanned by CamScanner

(4) Output Feedback Mode (OFB)

→ It is similar to CFB.

→ The only difference is that in CFB the ciphertext is fed into the next stage of encryption process. But in the case of OFB, the output of the IV encryption process is fed in the next stage of encryption process.

→ **Advantage of OFB:**

→ If a ciphert text bit C_i is in error, only the decrypted value corresponding to this bit i.e. P_i is wrong. Other bits are not affected

→ In contrast to this, in CFB mode, the cipher text bit C_i is fed back as input to the shift register and would corrupt the other bits in the message

→ **Disadvantage of OFB:**

→ The attacker changes both the cipher text and the checksum at the same time, hence there is no way to detect this change.

Scanned by CamScanner

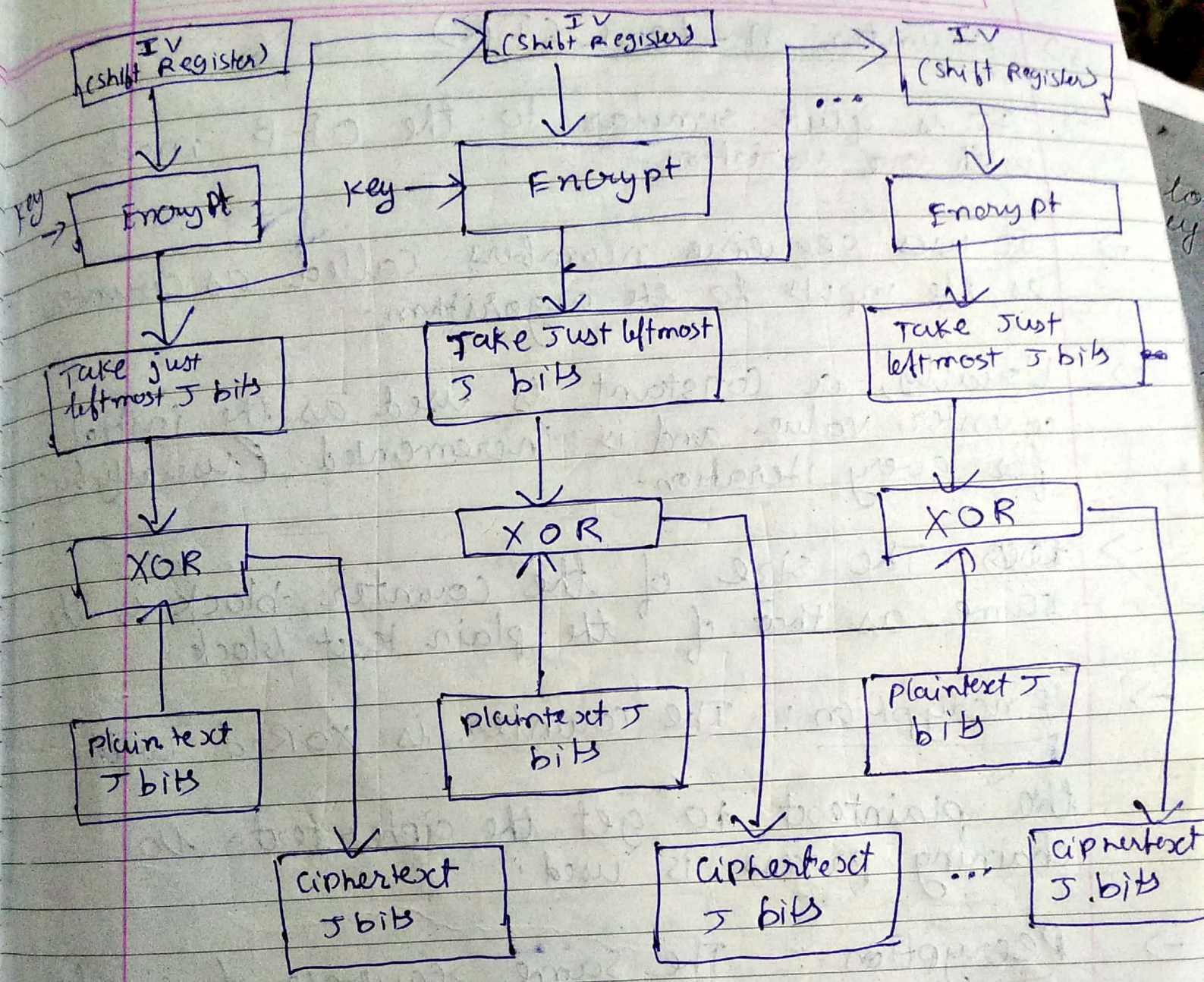


Fig: OFB Encryption process

Scanned by CamScanner

(5) Counter Mode (CTR)

-> It is quite similar to the OFB mode, with one variation.

-> It uses sequence numbers called as counter as the inputs to the algorithm.

-> Usually, a constant is used as the initial counter value and is incremented (usually by) for every iteration.

-> The size of the counter block is the same as that of the plain text block.

-> Encryption: The counter is XORed with the plaintext to get the ciphertext. No chaining process is used.

-> Decryption: The same sequence of counters is used. Here, each encrypted counter is XORed with the corresponding ciphertext block to obtain the original plaintext block.

Advantage: Encryption or decryption can be done in parallel on multiple blocks as no chaining

Scanned by CamScanner

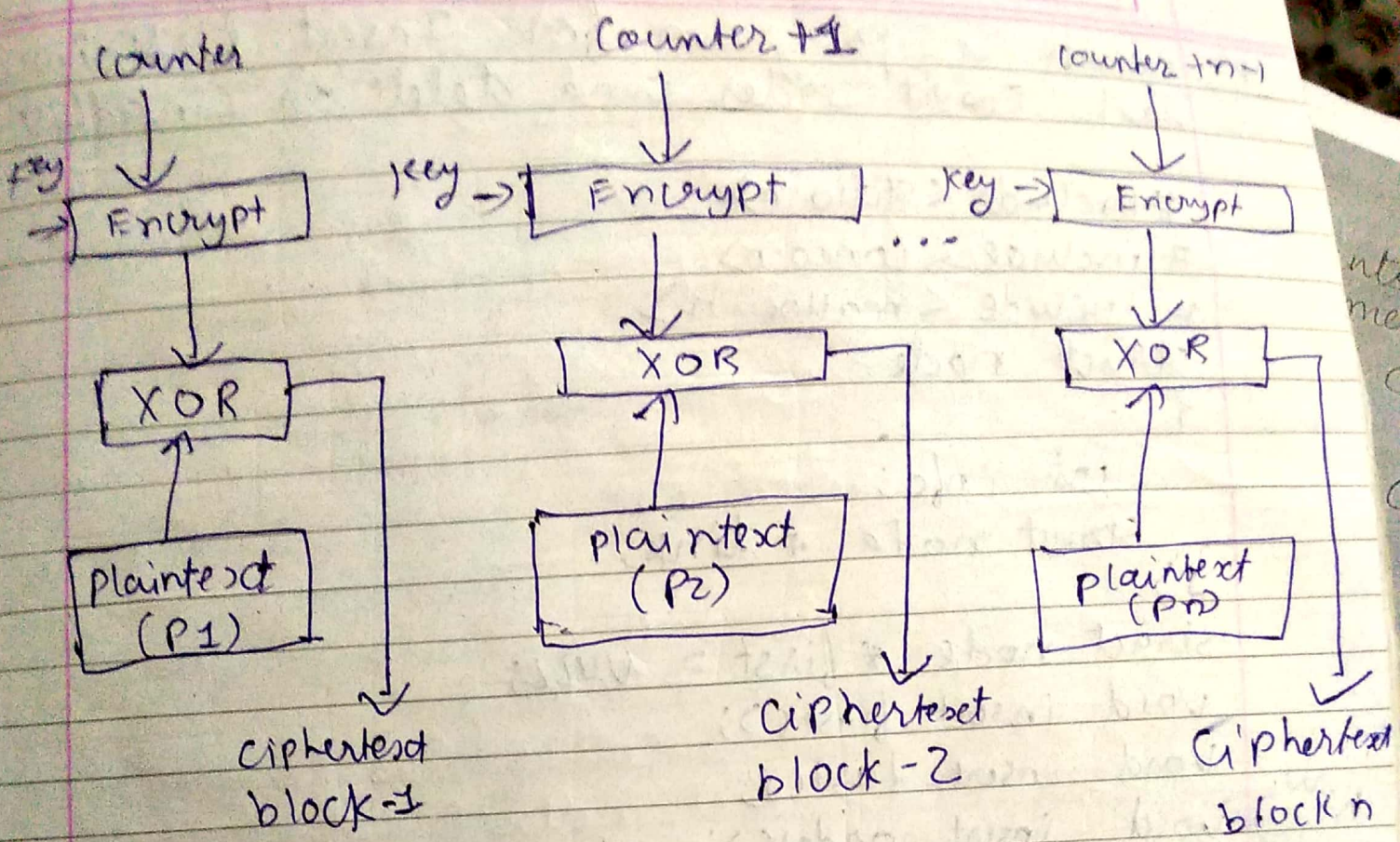


Fig: Encryption process CTR

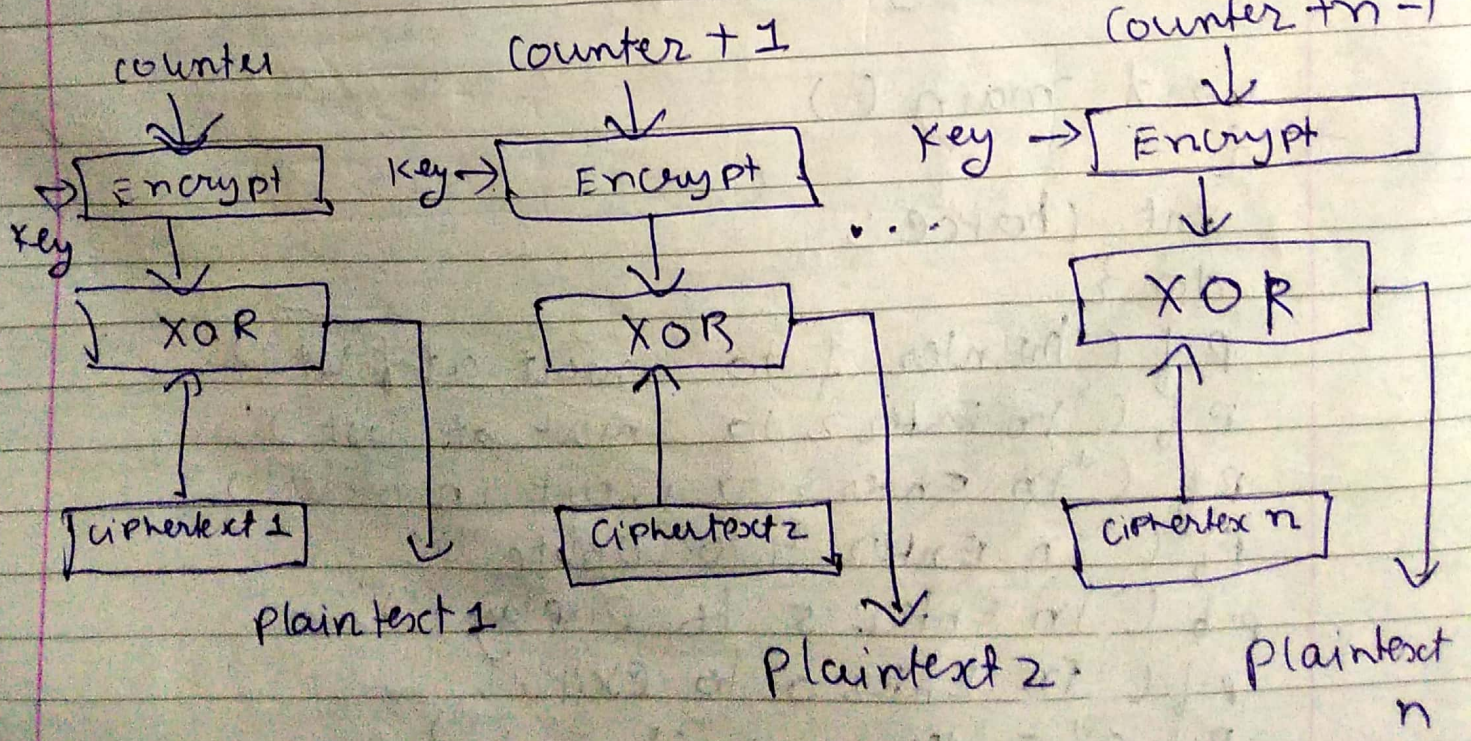


Fig: Decryption process CTR.

Scanned by CamScanner

* write a program for Insert first, insert last, Insert order and delete in LinkedList.

```
#include <stdio.h>
# include <conio.h>
# include <malloc.h>
struct node
{
    int info;
    struct node * link;
};
struct node * first = NULL;
void insert_first();
void insert_last();
N.W void insert_order();
void delet();
void display();

void main()
{
 int choice;
 do {
 p.f ("\nEnter 1 to insert at first");
 p.f ("\n Enter 2 to insert at last");
 p.f ("\n Enter 3 to insert in order");
 p.f ("\n Enter 4 to Delete");
 p.f ("\n Enter 5 to Display");
 p.f ("\n Enter 0 to exit");
 p.f ("\n Enter your choice:");
 s.f ("%d", & choice);
```

Scanned by CamScanner

```
switch (choice)
{
 case 1:
        insert_first();
        break;
 case 2:
        insert_last();
        break;
 case 3:  insert_order();
        break;
 case 4:  delete();
        break;
 case 5:  display();
        break;
 case 0:   break;

 default:   p.f ("Invalid choice");
        break;
    }
 } while (choice != 0);
}

void insert_first()
 {
  struct node *n
   int info_temp;
  n= (struct node *) malloc (sizeof (struct node
  if (first == NULL)
    {
      first = n;
    } first -> link = NULL;
```

Scanned by CamScanner

```
else {
        n -> link = first;
        first = n;
    }
    p.f("Enter the Info : ");
    s.f("%d", & Info_temp);
    first -> info = info_temp;
}
void insert_last()
{
    struct node *n
    struct node *temp
n = (struct node *) malloc (sizeof (struct node));
temp = (struct node *) malloc (sizeof (struct node));
    if (first = = NULL)
        {
            first = n;
            first -> link = NULL;
        }
    else {
            temp = first;
            while (1)
            {
                if (temp -> link = = NULL)
                    {
                        break;
                    }
                else { temp = temp -> link; }
            }
            temp -> link = n;
```

Scanned by CamScanner

```
        n -> link = NULL;
    p.f ("Enter info");
    s.f ("%d", & info_temp);
    n -> info = info_temp;

void display( )
  {
    struct node * temp;
    temp = (struct node *) malloc (sizeof (struct node))
    temp = first;
    if (first = = NULL)
      {
        p.f (" Empty list");
      }
    else {
        p.f (" Link List = \n");
        while (1)
        {
          p.f ("%d \t", temp -> info);
          if (temp -> link = = NULL)
            {
              break;
            }
          else {
                temp = temp -> link;
            }
        p.f ("\n");
      }
  }
```

Scanned by CamScanner

```
else {
        n -> link = first;
        first = n;
    }
p.f("Enter the Info : ");
s.f("%d", & Info_temp);
first -> info = info_temp;
}
void insert_last()
{
  struct node *n
  struct node *temp
n = (struct node*) malloc (sizeof (struct node));
temp = (struct node*) malloc (sizeof (struct node));
    if (first = = NULL)
        {
          first = n;
          first -> link = NULL;
        }
else {
        temp = first;
        while (1)
        {
          if (temp -> link = = NULL)
            {
              break;
            }
          else { temp = temp -> link; }
        }
        temp -> link = n;
```

Scanned by CamScanner

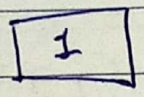
```
        n -> link = NULL;
    p.f ("Enter info");
    s.f ("%d", & info_temp);
    n -> info = info_temp;

void display( )
 {
  struct node *temp;
 temp = (struct node *) malloc (sizeof (struct node));
 temp = first;
 if (first = = NULL)
    {
        p.f (" Empty list");
    }
else {
        p.f (" Link List = \n");
       while (1)
        {
         p.f ("%d" \t", temp -> info);
           if (temp -> link = = NULL)
             {
                break;
             }
           else {
                  temp = temp -> link;
                }
          p.f ("\n");
        }
    }
```

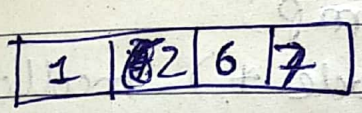
Scanned by CamScanner

B - tree of order 5

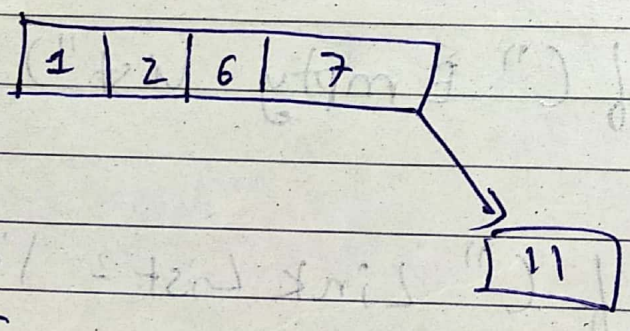
Step 1: Insert 1



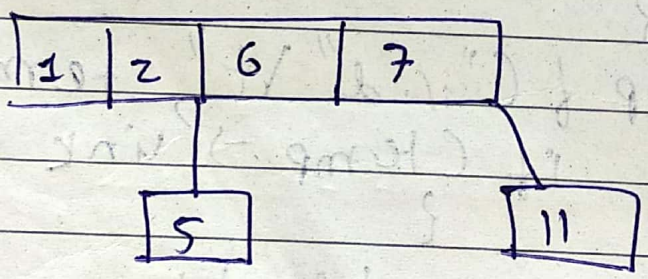
Step 2 : Insert 7, 6, 2



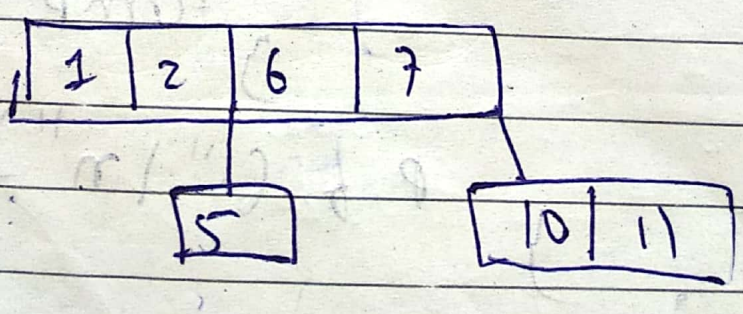
Step 3 : Insert 11



Step 4: Insert 5

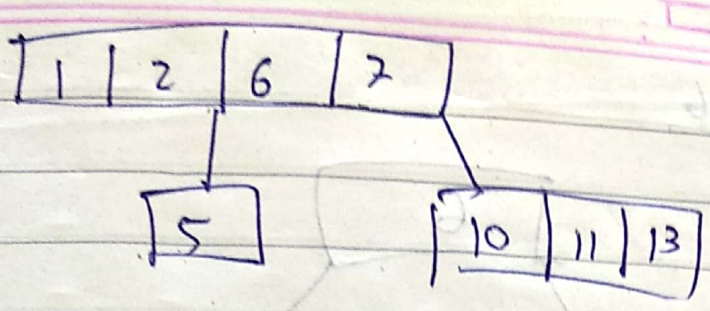


Step 5: Insert 10

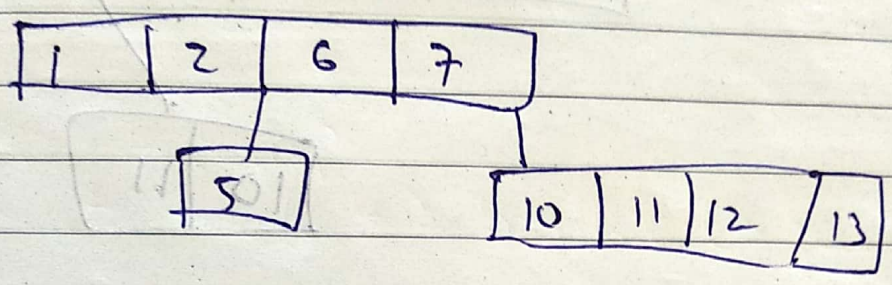


Step 6: Insert 13

Scanned by CamScanner



Insert 12



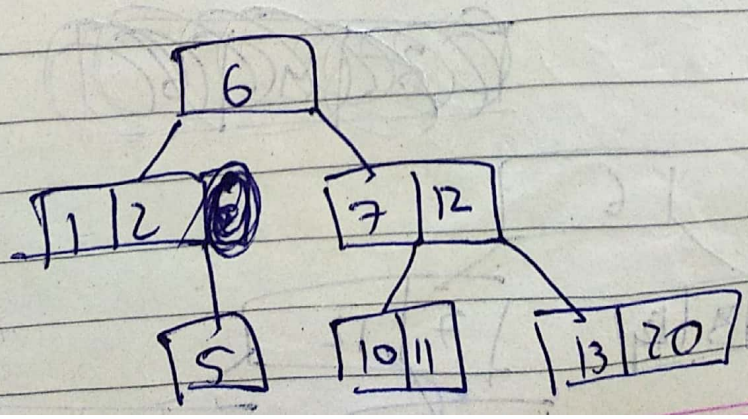
Insert 20

10, 11 → left
12 → mid
13, 20 → Right

Move 12 to parent. Here also full, so

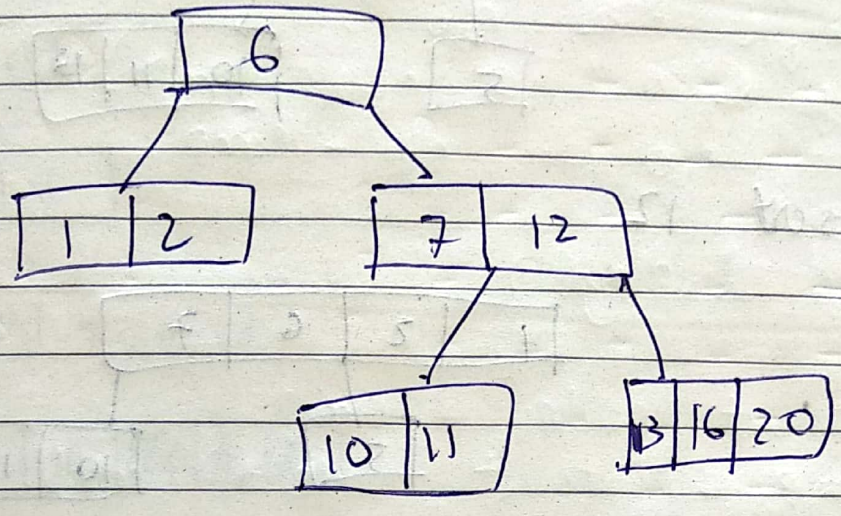
1, 2 → left
6 → mid
7, 12 → Right

move 6 to root

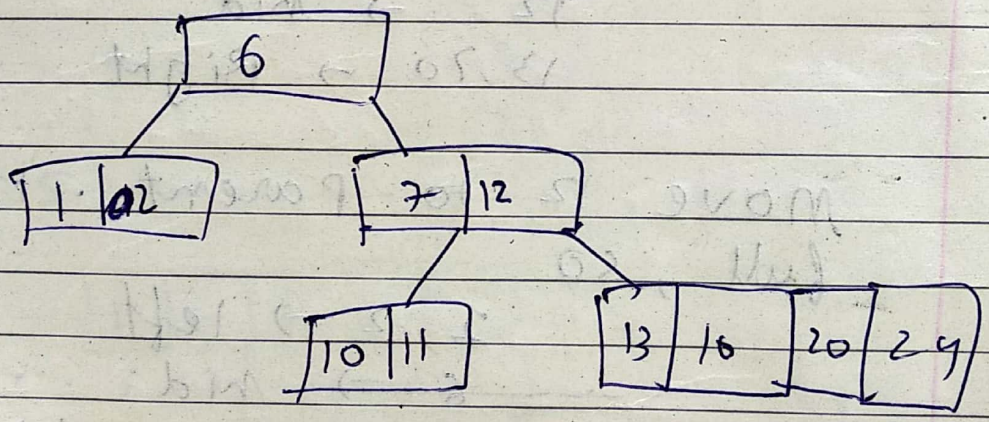


Scanned by CamScanner

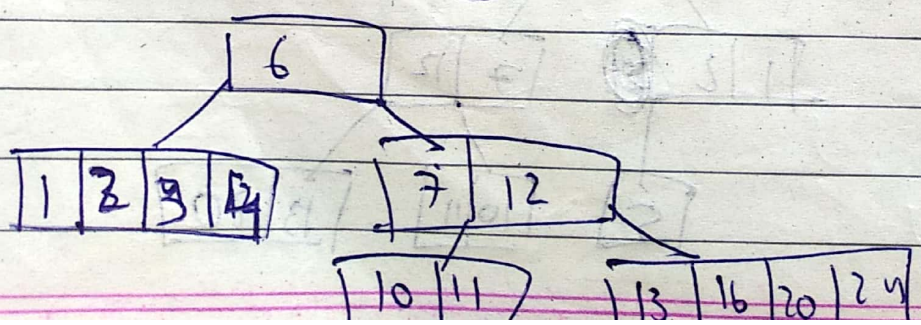
Step 9 Insert 16



Step 10 Insert 24



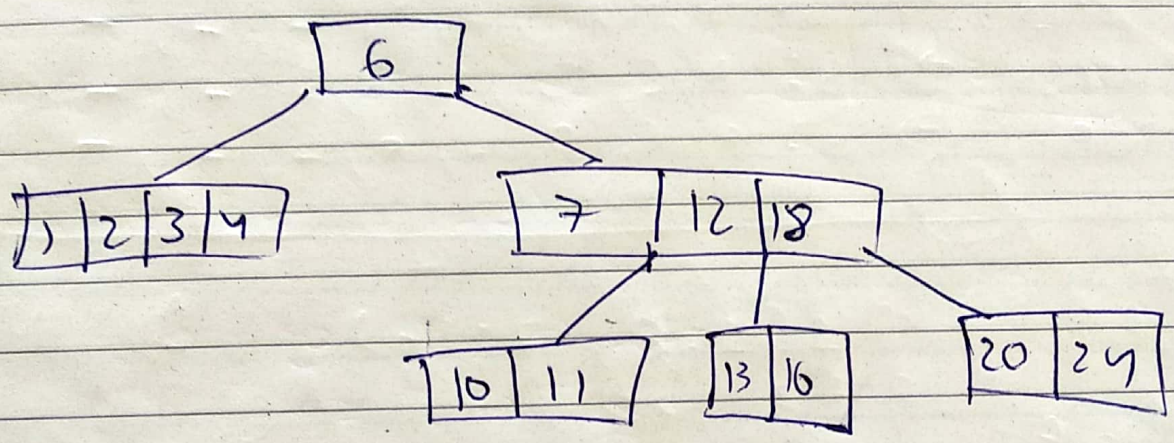
Step 11: Insert 3, 4



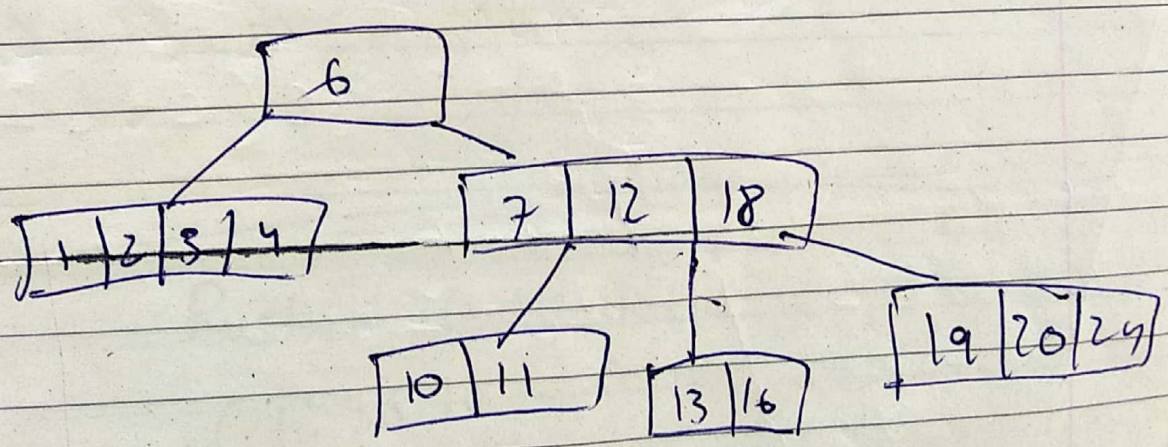
Scanned by CamScanner

Step 12: Insert 18

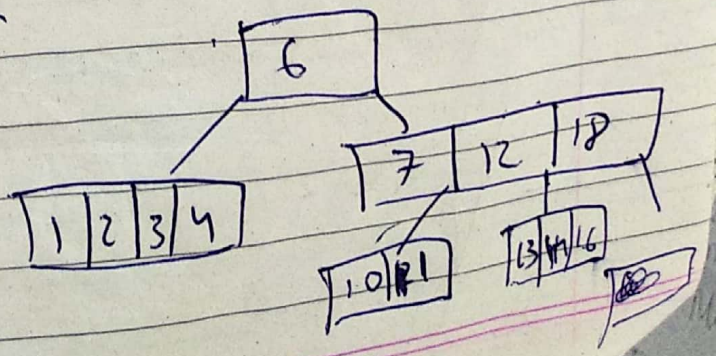
13, 16 → Left
18 → Mid
20, 24 → Right
move 18 to parent



Step 13: Insert 19



Step 14: Insert 14, 25



Scanned by CamScanner

Q2 (B)

((A - (B + C)) X D) / (E + F)

↳ Answer is

ABC+ - D X EF+ /

section-B

(((A - (BC+)) X D)

(ABC+ -) X D / (EF+)

(ABC+ - D X) / EF+

ABC+ - D X EF+ /

Rushin Makwana, 98 DIVD

(1:35 to 2:10) M.No: 9377804320

Scanned by CamScanner

Section A

Q1 (B) S: A, C, D, F, K, -, -, -

(1) POP() → S: A, C, D, F, -, -, -, -
(2) POP() → S: A, C, D, -, -, -, -, -
(3) Push(L) → S: A, C, D, L, -, -, -, -
(4) Push(P) → S: A, C, D, L, P, -, -, -
(5) POP() → S: A, C, D, L, -, -, -, -
(6) Push(R) → S: A, C, D, L, R, -, -, -
(7) Push(S) → S: A, C, D, L, R, S, -, -
(8) POP() → S: A, C, D, L, R, -, -, -

Q2 (A) A1(1:8, 7:14)
B.A = 100
size = 4
A1(4, 12) = ?

Section A

Answer is 216

$A[I,J] = B.A + size(c*(I - L_r) + (J - L_c))$

$C = U_c - L_c + 1$

Here $L_r = 1$, $U_r = 8$, $L_c = 7$, $U_c = 14$

So $C = 14 - 7 + 1$

$\boxed{C = 8}$

$A1[\overset{I}{4}, \overset{J}{12}] = 100 + 4(8(4-1) + (12-7))$
$= 100 + 4(8(3) + 5)$
$= 100 + 4(24 + 5)$
$A[4,12] = 100 + 4($
$= 216$

Q 3 (A) As we have quadratic nested loop

time complexity = $n * n$
$= n^2$

section A

---

section B Q 3 (A)

$a + b * c - d / e * h$

$\rightarrow a + (bc*) - d / e * h$

$\rightarrow a + (bc*) - (de/) * h$

$\rightarrow a + (bc*) - (de/h*)$

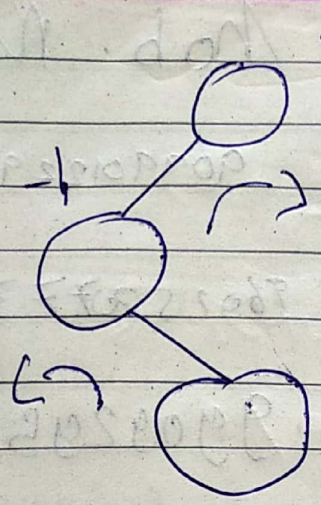
$\rightarrow (abc*+) - (de/h*)$

$\rightarrow \boxed{abc*+de/h*-}$

Scanned by CamScanner

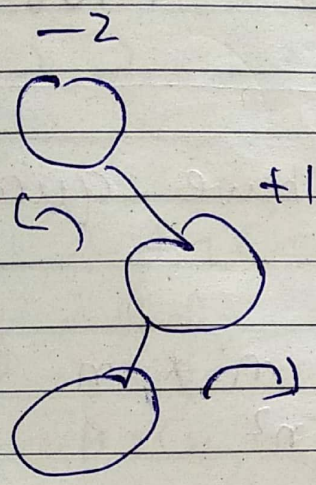
Case

(4)



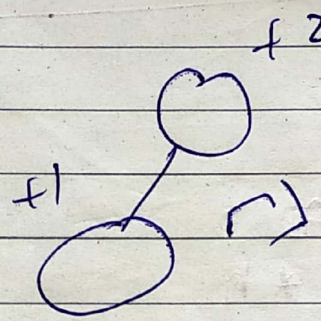
→ L R

(3)



→ R L

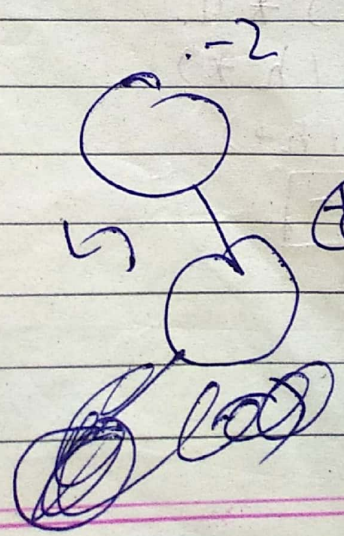
(2)



Left heavy

Right Rotation

(1)



Right heavy

Left rotation

Scanned by CamScanner

To find inverse of $M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 5 & 6 & 0 \end{bmatrix}$

Step 1: Find determinant. If should not be zero.

$\det(M) = 1(0-24) - 2(0-20) + 3(0-5)$
$= 1 \neq 0$ — So inverse is possible

Step (2): Transpose original matrix

$$M^T = \begin{bmatrix} 1 & 0 & 5 \\ 2 & 1 & 6 \\ 3 & 4 & 0 \end{bmatrix}$$

Step 3: Find determinant of each minor.

$\begin{vmatrix} 1 & 6 \\ 4 & 0 \end{vmatrix} = -24$ $\quad \begin{vmatrix} 2 & 6 \\ 3 & 0 \end{vmatrix} = -18$ $\quad \begin{vmatrix} 2 & 1 \\ 3 & 4 \end{vmatrix} = 5$

$\begin{vmatrix} 0 & 5 \\ 4 & 0 \end{vmatrix} = -20$ $\quad \begin{vmatrix} 1 & 5 \\ 3 & 0 \end{vmatrix} = -15$ $\quad \begin{vmatrix} 1 & 0 \\ 3 & 4 \end{vmatrix} = 4$

$\begin{vmatrix} 0 & 5 \\ 1 & 6 \end{vmatrix} = -5$ $\quad \begin{vmatrix} 1 & 5 \\ 2 & 6 \end{vmatrix} = -4$ $\quad \begin{vmatrix} 1 & 0 \\ 2 & 1 \end{vmatrix} = 1$

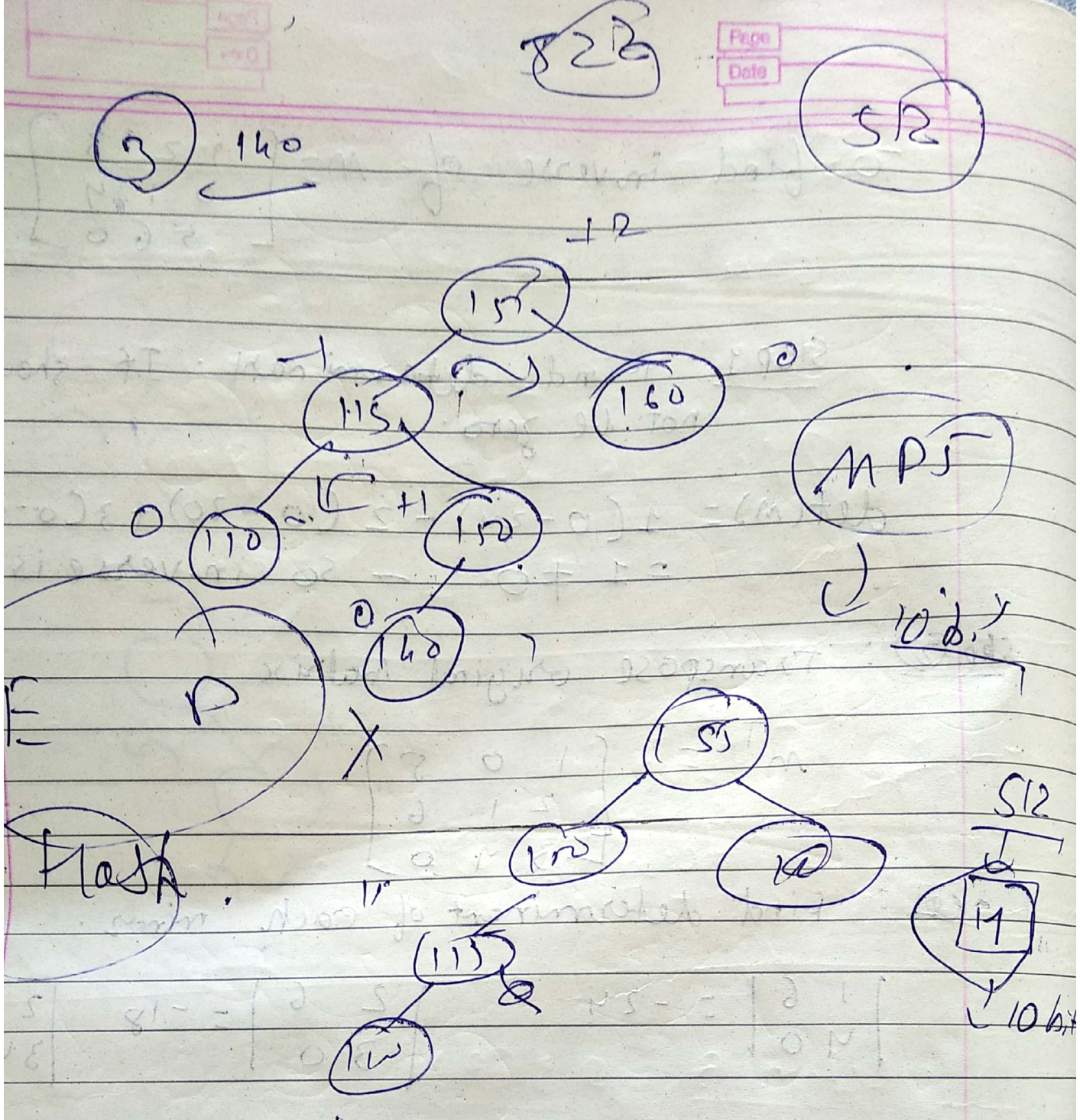
Step 4:

$$\text{Adj}(M) \begin{bmatrix} -24 & -18 & 5 \\ -20 & -15 & 4 \\ -5 & -4 & 1 \end{bmatrix} \times \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}$$

$$\text{Adj}(M) = \begin{bmatrix} -24 & 18 & 5 \\ 20 & -15 & -4 \\ -5 & 4 & 1 \end{bmatrix}$$

$$M^{-1} = \frac{1}{\det(M)} \times \text{Adj}(M)$$

answer as det is 1

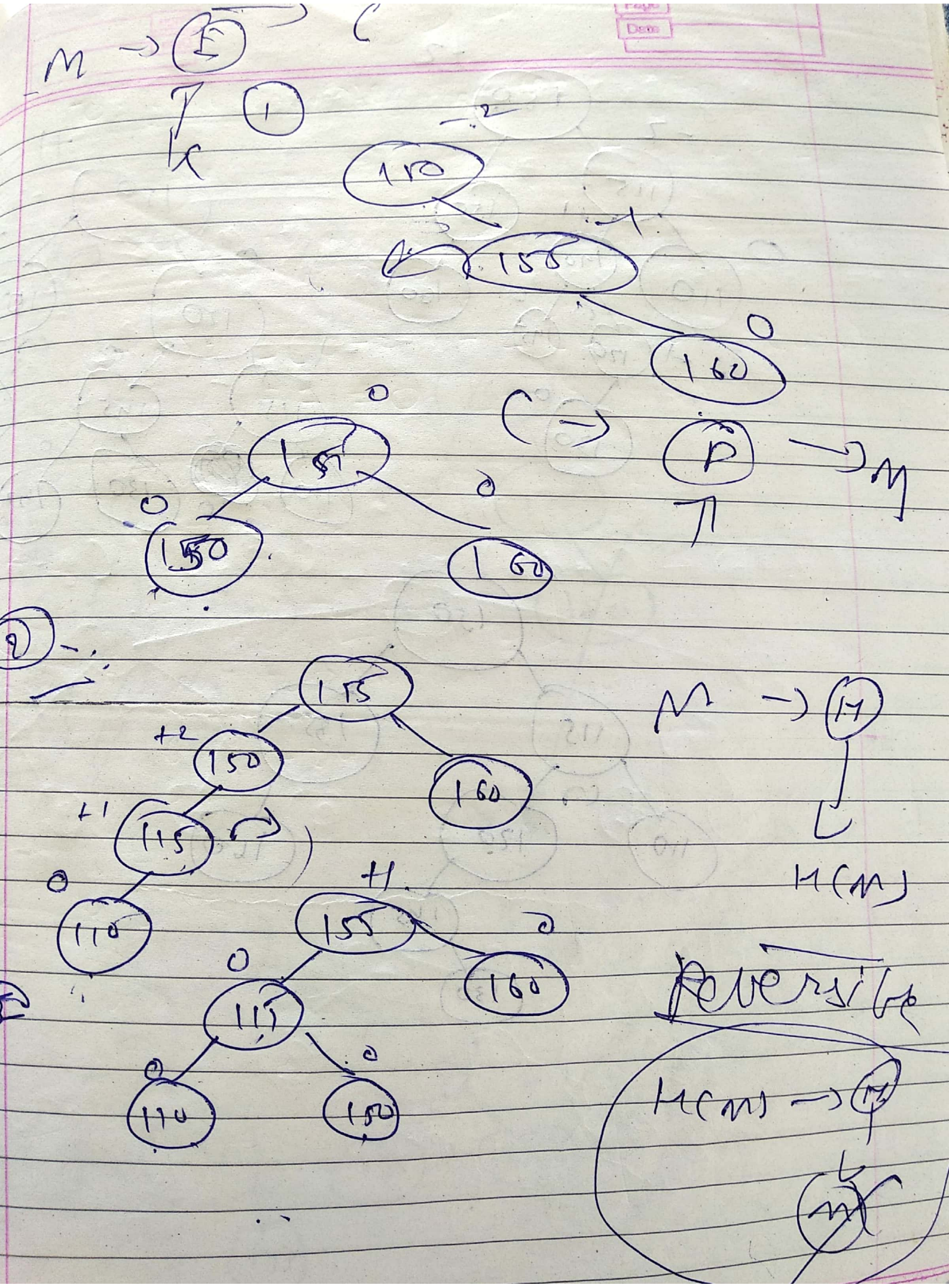


M → 512 bit + concate.

Hash → 10 → Receiver

512	10 bit

→ Hash



Scanned by CamScanner