UNIT - V

Key Management & Distribution

* **Symmetric Key distribution using symmetric Encryption :**

Key Distribution :

Key distribution
- Symmetric Key Distribution
  - → Symmetric encryption
  - → Asymmetric encryption
- Public key Distribution
  - → Public announcement of Public keys
  - → Publicly Available Directory
  - → Public key Authority
  - → Public key certificates

- When two parties share the same key (i.e. symmetric key) that protect from access by others, the process b/w two parties that exchanges that key called as symmetric key distribution.
- If two person wants to communicates with each other via msg. or exchange data without interference of other.

- Two parties/ person A and B achieved the key distribution in various ways:

1) A can select a key and physically deliver it to B.
2) A third party can select the key and physically deliver it to A and B.
3) If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4) If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

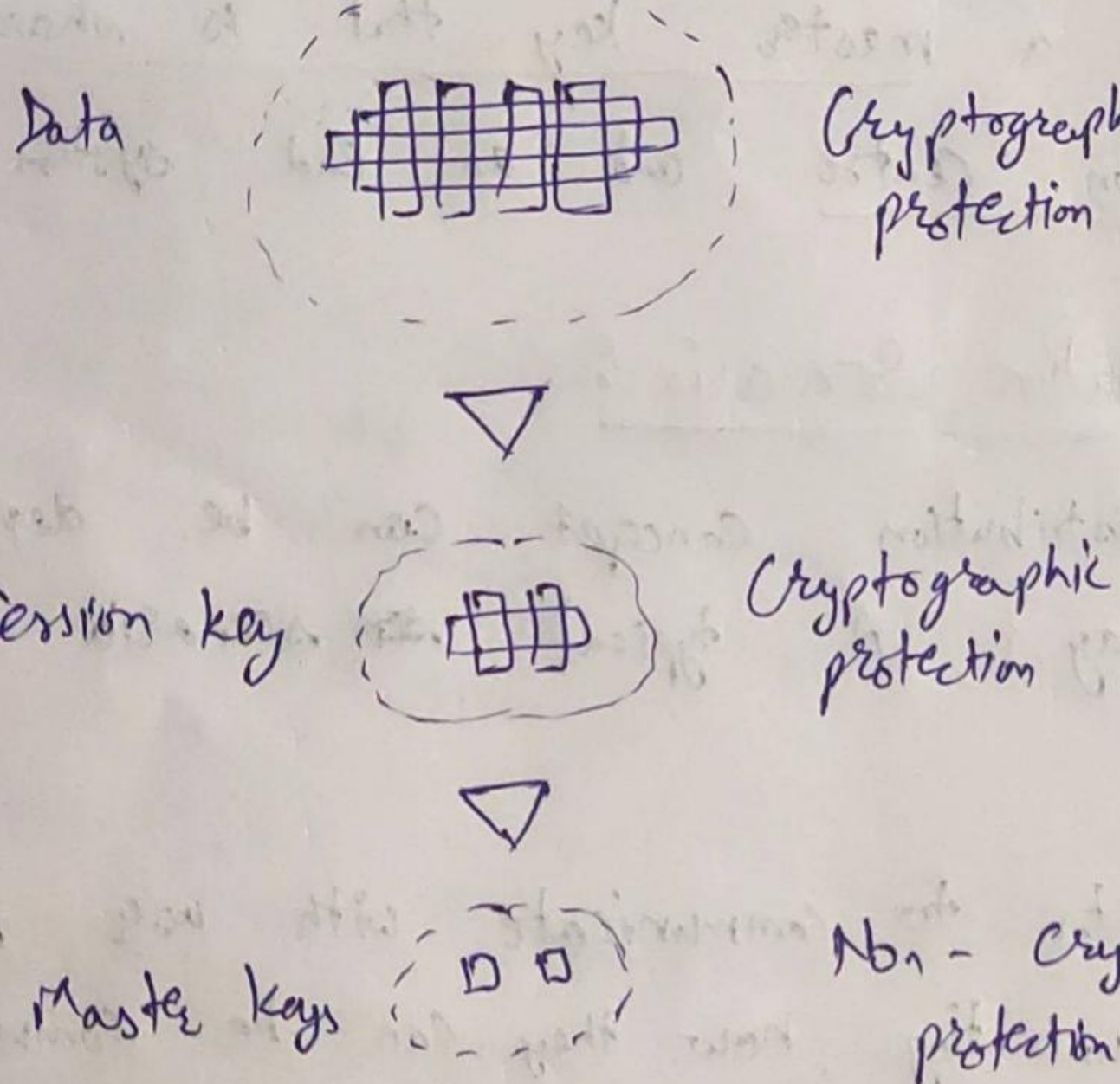
Description of all four points:

- option 1 and 2 call for manual delivery of a key to the users. In manual delivery of key is difficult in a wide-area distributed system. (for local Connection or if both the user communicate within a LAN or within a building then Option 1 & 2 is reliable.)

- Returning to our list, option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed. (attacker will capture the keys so subsequent keys will be capture by the attacker or all the msg.

decrypted by the hacker, or attacker.

- For end-to-end encryption some variation on option 4 has been widely adopted.
- In this scheme, a key distribution centre responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.
- Each user must share a unique key with the distribution centre for purpose of key distribution.
- The use of a key distribution centre is based on the use of a hierarchy of keys.



- At a minimum, two levels of keys are used (in fig) Communication b/w end systems is encrypted using a temporary key, often referred to as a session key.

(session key is mostly used for a logical connection.
(means there is a connection establish for data transfer
if data transfer is completes then session will be terminated

- Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded.

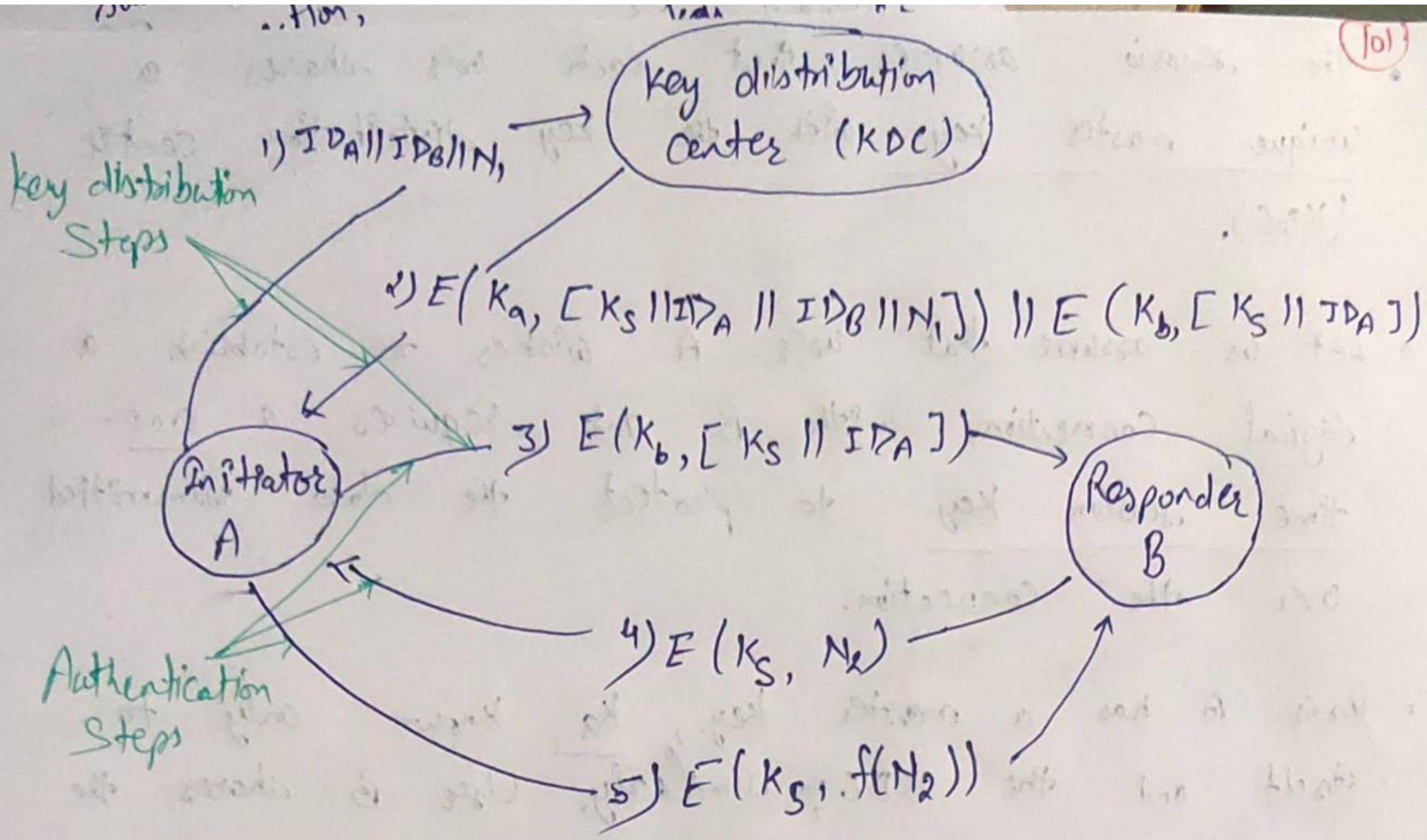
- Each session key is obtained from the key distribution centre over the same networking facilities used for end-user communication.

- Accordingly, session keys are transmitted in encrypted form, using a master key that is shared by the key distribution centre and an end system or user.

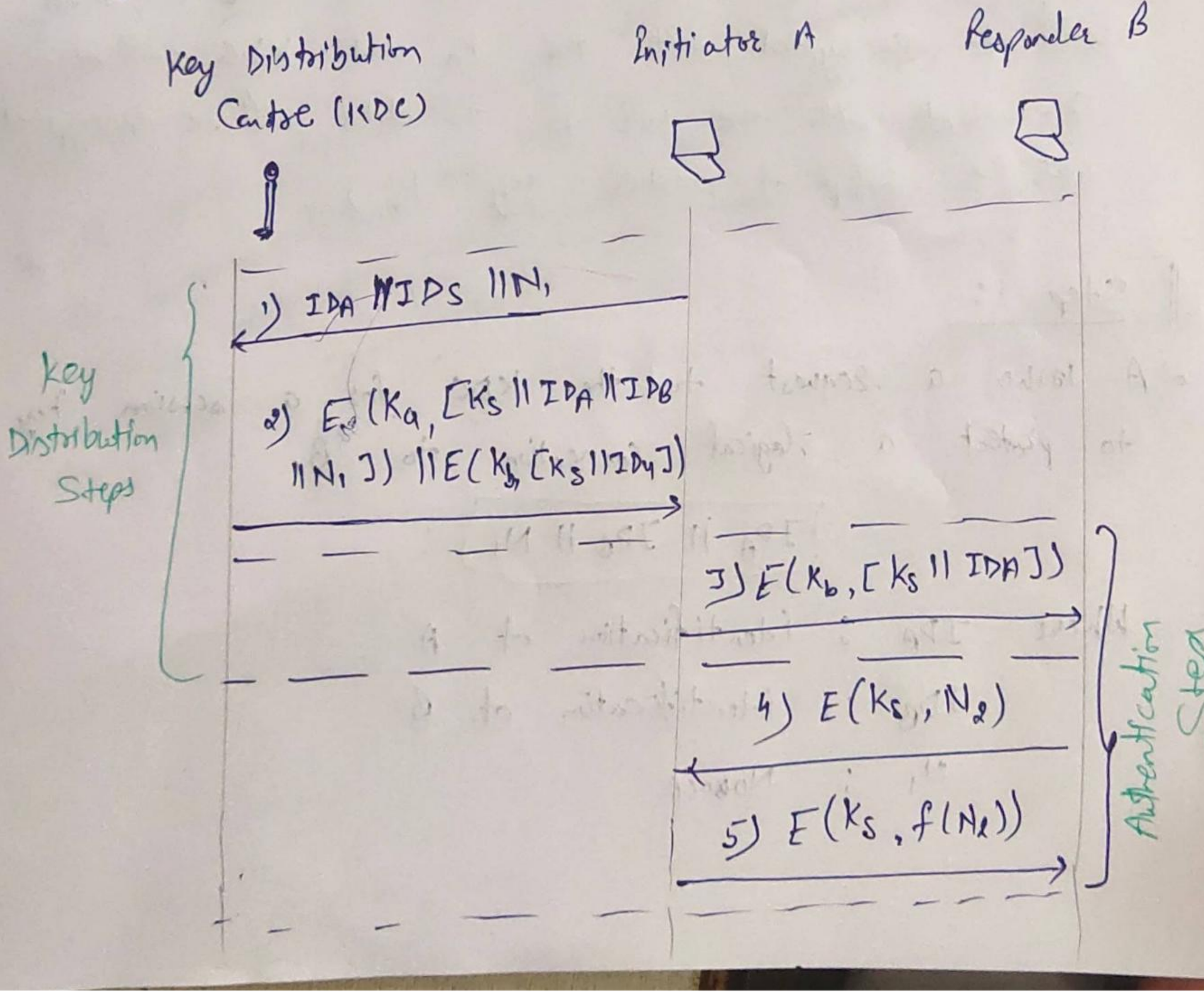
Key distribution Scenario :

- The key distribution concept can be deployed in a no. of ways. A typical scenario is illustrated in fig:

→ User A wants to communicate with user B. So before communicating, now they can be communicate with the key distribution center & get session key for the further communication with the user B.



→ Simplified version of above diagram is :



- the scenario assumes that each user shares a unique master key with the key distribution centre (KDC).

- Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.

- User A has a master key, K_a, known only to itself and the KDC; similarly, User B shares the master key K_b with the KDC.

(Whenever if KDC communicate with the user A, Data will be encrypted with the K_a and KDC will be communicate with the user B, so data will be encrypted with the help of master key K_b.)

$ Step 1:

- A issue a request to the KDC for a session key to protect a logical connection to B.

$$ID_A \,||\, ID_B \,||\, N_1$$

Where ID_A : identification of A

ID_B : identification of B

N_1 : Nonce

- the msg. includes the identity of A and B and a unique identifier, N_1 for this transaction, which we refer to as a nonce.

- the nonce may be a timestamp, a counter or a random no., the minimum requirement is that it differs with each request.

(Nonce N_1 will be appended to the with the IDA or IDB and send to the KDC).

- Also to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random no. is a good choice for a nonce.
(means nonce will be used to prevent masquerade attack).

Step 2:

- The KDC responds with a msg. encrypted using K_a.
Thus, A is the only one who can successfully read the msg. and A knows that it originated at the KDC.

$$E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$$

- The msg includes two items intended for A:
  - ↳ One time session key, K_s to be used for the session.

↳ the original request msg, including the nonce, to enable A to match this response with the appropriate request.

- Thus, A can verify that its original request was not altered before reception by the KDC and because of nonce.

- In addition, the msg includes two items intended for B:

  ↳ The one-time session key, K_S to be used for the session.

  ↳ An identifier of A (eg. its n/w add.), ID_A

- These last two items are encrypted with K_b (the master key that the KDC shares with B.).
  ⇒ They are to be sent to B to establish the connection and prove A's identity.

Step 3:

- A store the session key for use in the upcoming session and forwards to B the info. that originated at the KDC for B, namely $E(K_b [K_S || ID_A])$.

- Because this info. is encrypted with K_b, it is protected from eavesdropping.

- B now knows the session key (K_s), knowing that the other party is A (from ID_A), and knows that info. originated at the KDC (because it is encrypted using K_b).

- At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

$ Step 4:

- Using the newly minted session key for encryption, B sends a nonce, N_2 to A.

$$E(K_s, N_2)$$

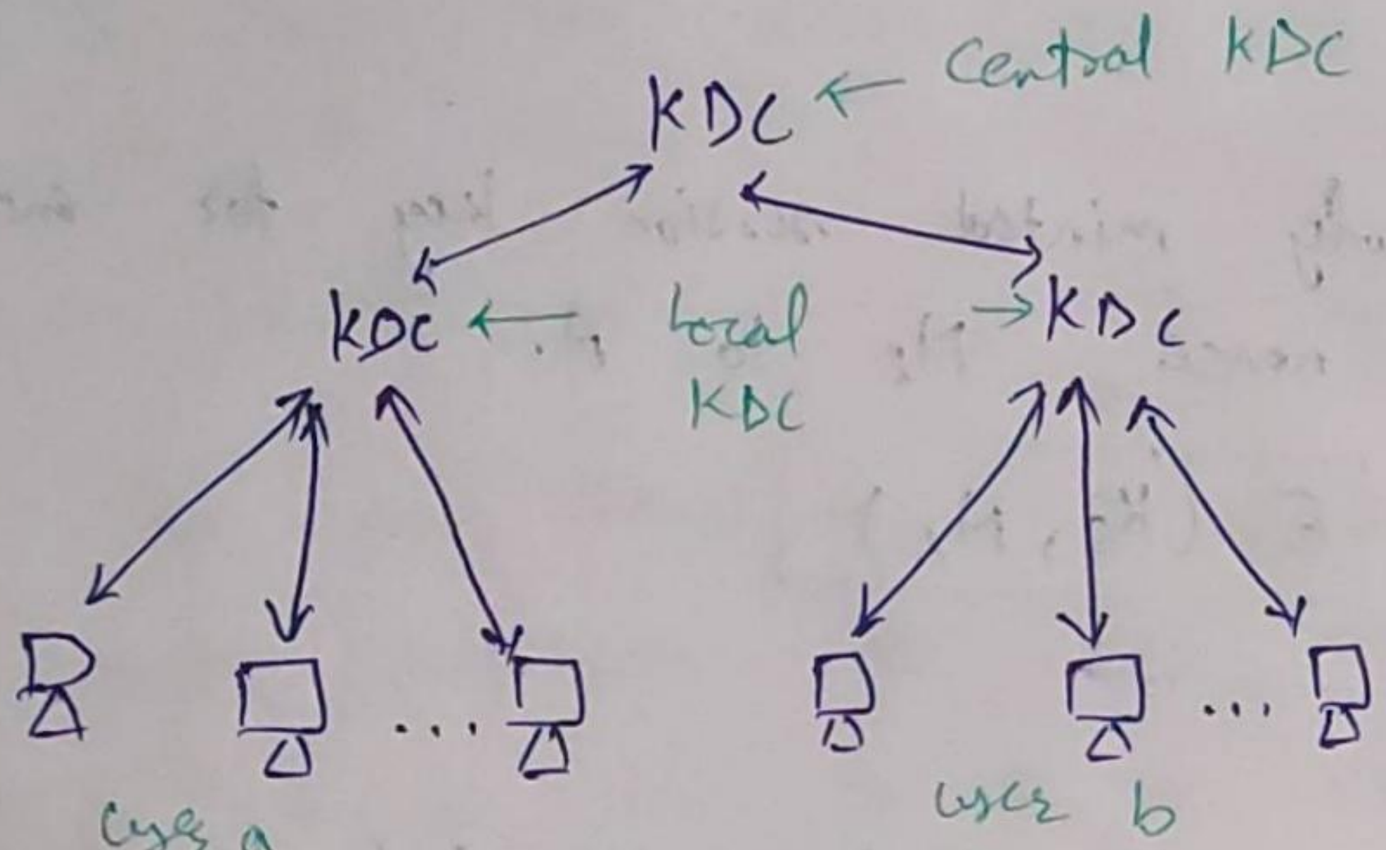
$ Step 5:

- Also, using K_s, A responds with $f(N_2)$, where f is a fun^n. that performs some transformation on N_2 (eg. adding one) or / substraction).

$$E(K_s, f(N_2))$$

Hierarchical key Control:

- It is not necessary to limit the key distribution fun^n to a single KDC. (If wide area n/w will be used there single KDC will be failure).
- Indeed, for very large n/w, single KDC is not enough to distribute keys among all users.
- As an alternative, a hierarchy of KDCs can be established.



- for ex., there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.

if user a wants to communicate with user b so user a call request to the its local KDC and local KDC tries to find out its Local area n/w, it is not available then it is request to the Central KDC, Central KDC forward that request to the another local KDC and local KDC sends this request to user b.

finally a and b establish a connection and wants to communicate with each other.

- for communication among entities within the same local domain, the local KDC is responsible for key distribution.

(if first two PCs dont want to communicate with each other then there is no communication with central KDC, only local KDC will establish a secure connection).

- If two entities in diff. domain desire a shared key, then the corresponding local KDCs can communicate through a global KDC.

- In this case, any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the no. of users and geographic scope of internetw.

Session key Life time:

- the most frequently session keys are exchanged, the more secure they are, because the attacker has to capture session key every time to decrypt cipher text.
- Short session key life time → key exchange frequently & more secure.

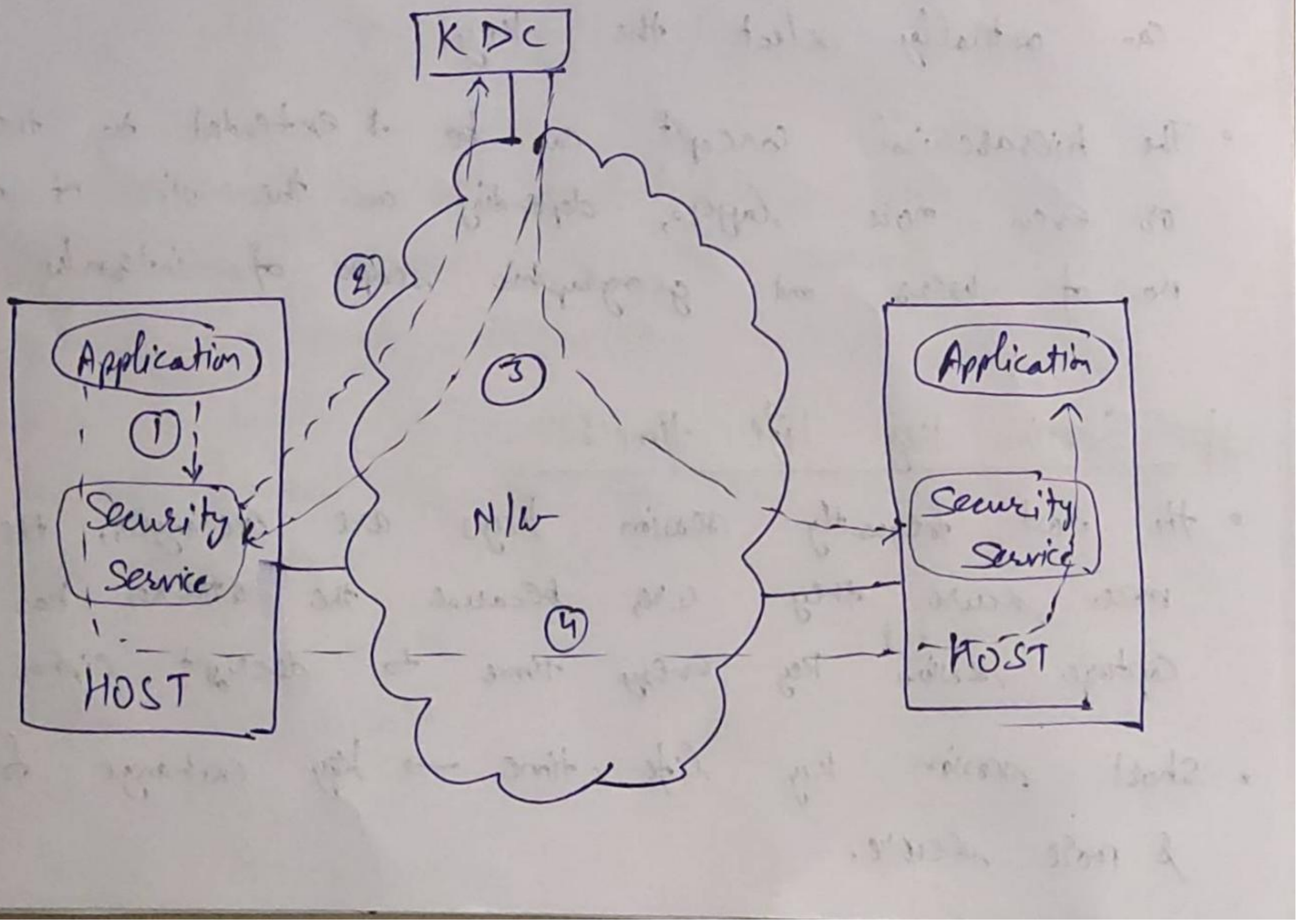
- Long session key life time → Reduce key exchange time & less n/w bandwidth used.

- For Connection - oriented protocols, new session key for each new connection. Update key periodically, if the connection has long time. This is a eg. of TCP Connection.

- For connection less protocols, not to use a new key for each session but use a given session key for a fixed period of time. (This is a eg. of UDP protocol.

A transparent Key Contol scheme:

- The steps involved in establishing a connection are shown in fig:



1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.

- When one host wants to set up a connection to another host, it transmits a connection-request packet (step 1).
- The SSM (session security module) saves that packet and applies to the KDC for permission to establish the connection (step 2).
- The communication btw the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.
- If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3).
- The requesting SSM can now release the connection request packet, and a connection is set up btw the two end systems (step 4).

- All user data exchanged b/w the two end systems are encrypted by their respective SSMs using the onetime session key.

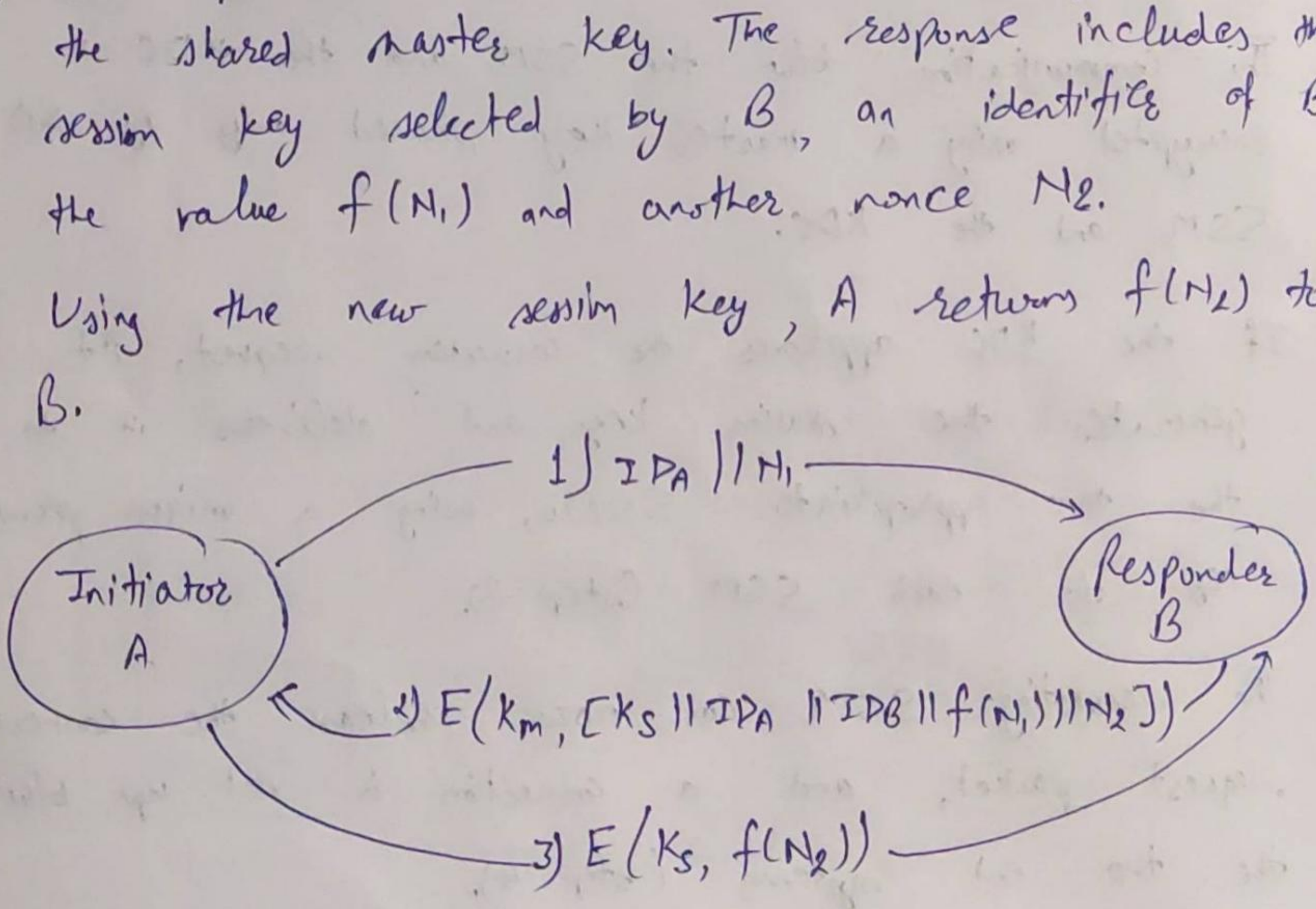
Decentralized Key Control:

- It is not practical for larger n/w using symmetric Encryption only, it may be useful within a local context.
- A session key may be established with the following sequence of steps.

1) A issues a request to B for a session key and includes a nonce N_1.

2) B responds with a msg that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$ and another nonce N_2.

3) Using the new session key, A returns $f(N_2)$ to B.



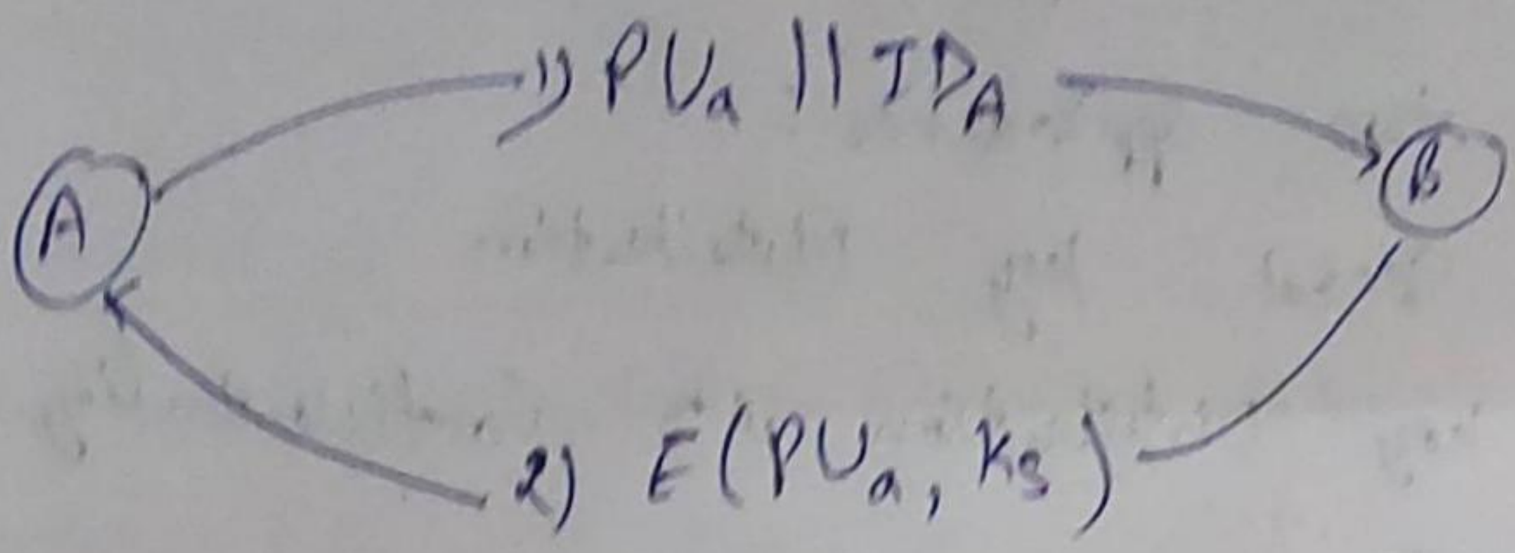
* Symmetric Key Distribution using Asymmetric Encryption:

- There are two approaches:
  1) Simple Secret key Distribution
  2) Secret key Distribution with Confidentiality and Authentication.

Simple Secret Key Distribution:

- If A wishes to communicate with B, the following procedure is employed:
  1) A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a msg 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, which is encrypted with A's public key.
- A decrypt msg using $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the msg., only A and B will know the identity of K_s.
- A discards PU_a and PR_a and B discards PU_a.
- A and B can now securely communicate using conventional encryption and the session key K_s.

At the completion of the exchange, both A and B discard K_s.



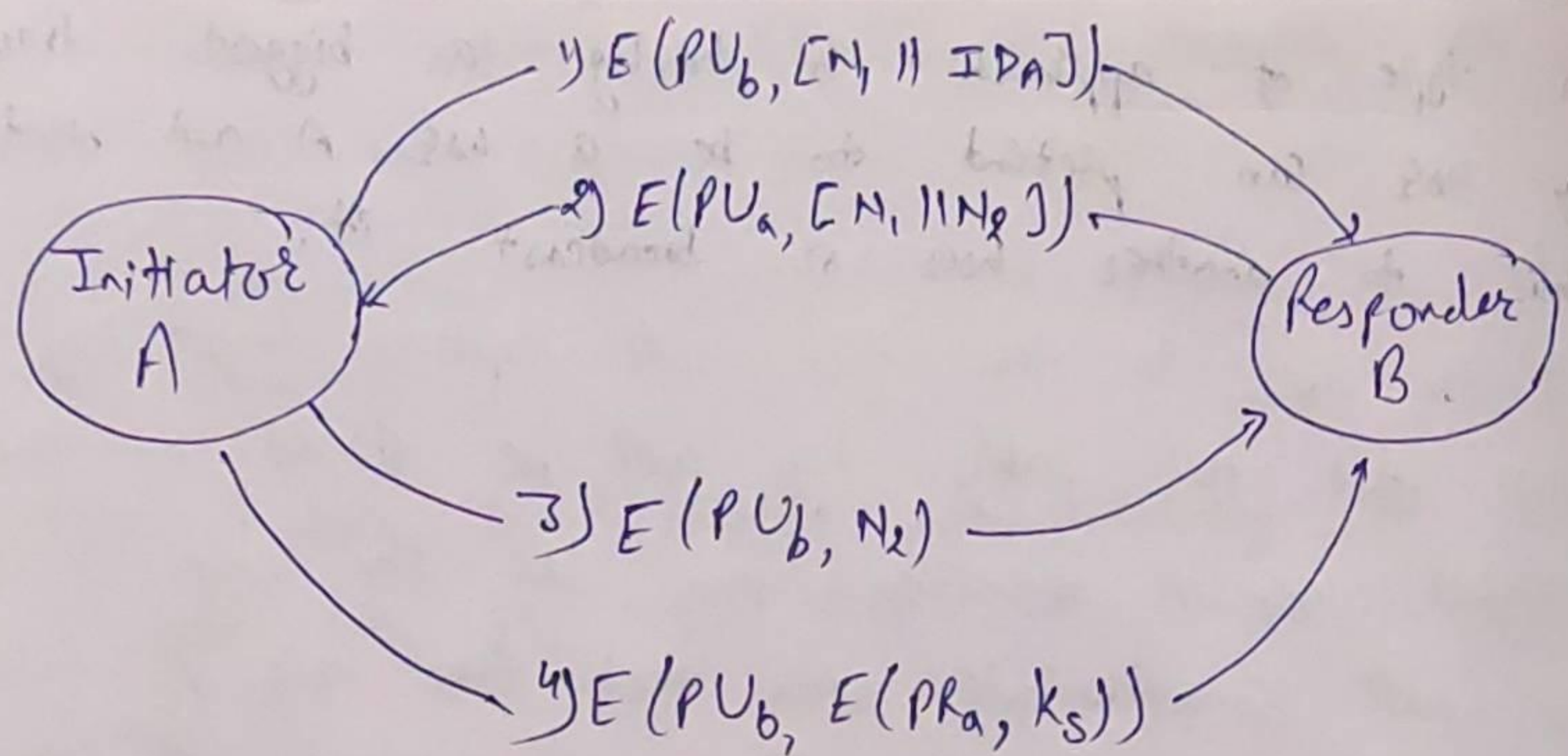
Secret Key Distribution with Confidentiality and Authentication?

1) A uses B's public key to encrypt a msg. 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 msg. to A encrypted with PU_a and containing A's nonce as (N_1) and also a new nonce generated by B (N_2). Because only B could have decrypted msg. (1), the presence of N_1 in msg (2) assures A that the correspondent is B.

3) A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

4) A selects a secret key and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this msg. with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.

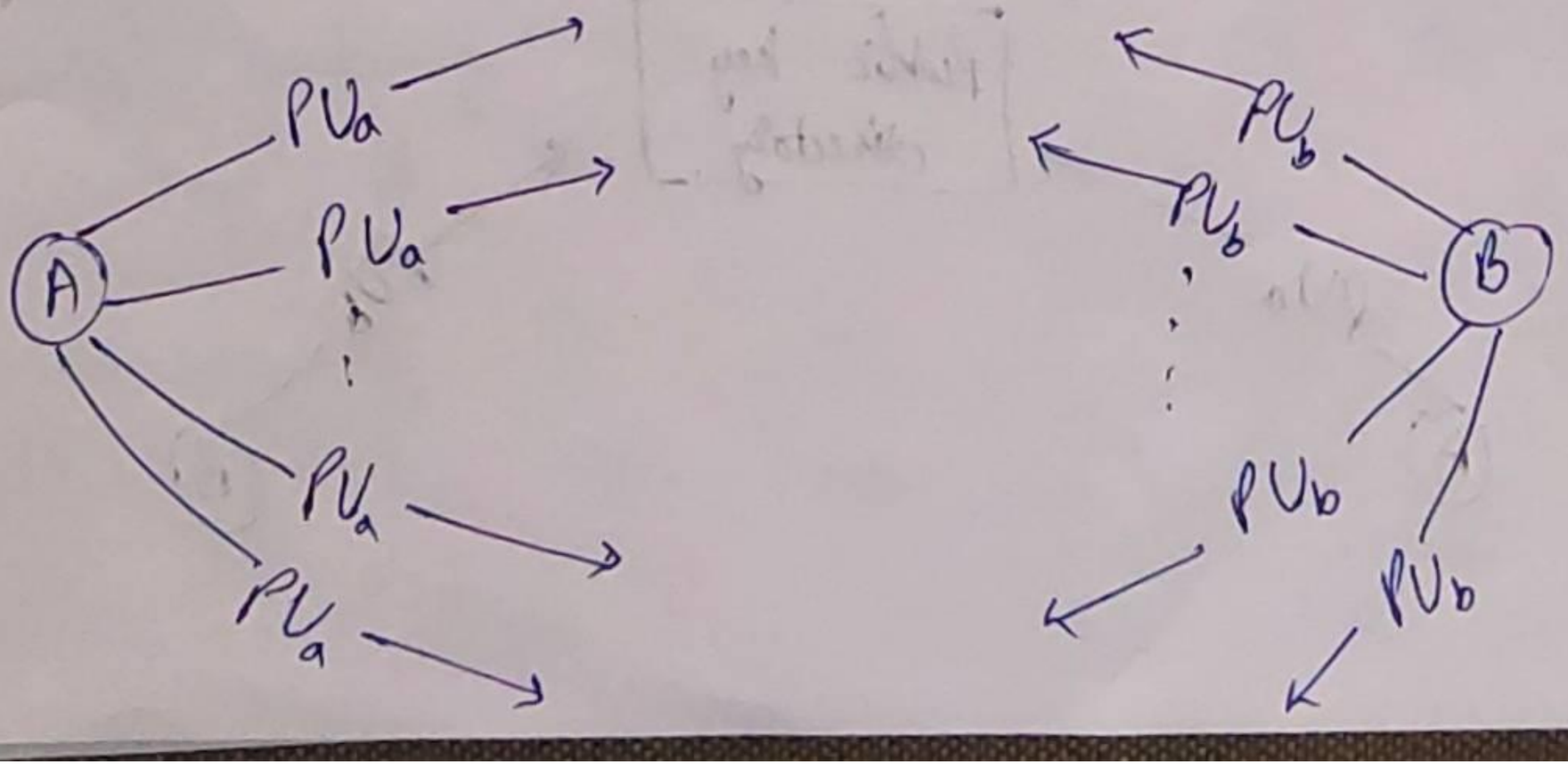


- B decrypt the msg. and get secret key Ks.
- The result is that this scheme ensures both confidentiality and authentication in the exchang of a secret key.

* Distribution of Public keys:

Public Announcement:

- In a public key cryptography, such as RSA, any user can send his/her key to any other user or broadcast it to the group as shown in fig.

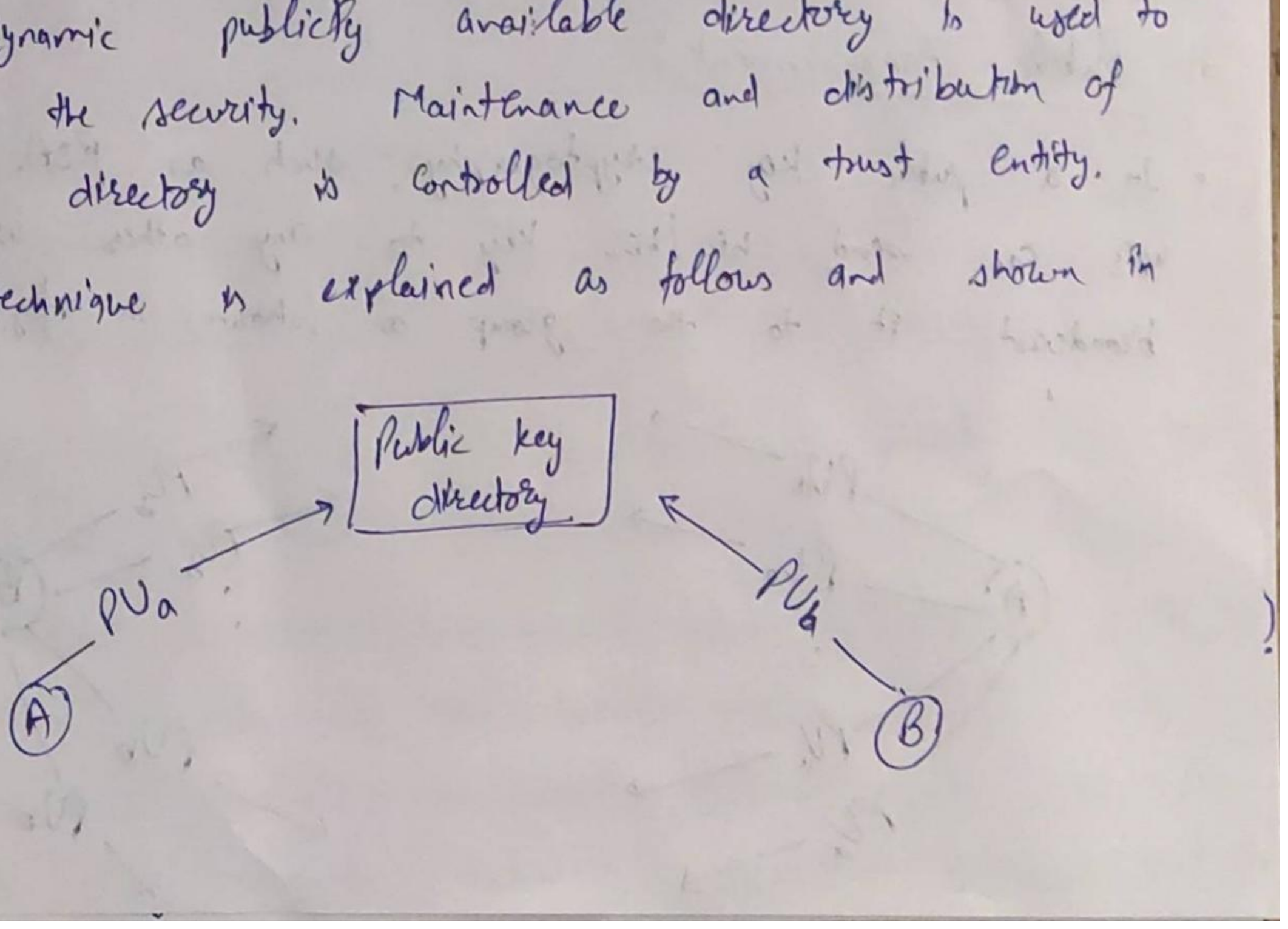


- This type of approach is having a biggest drawback. Any user can pretend to be a user A and send a public to another user or broadcast it.
(All the users in the n/w will capture that key and think that this public key will be shared by the original user A, whenever those all the users in the n/w will communicate with that fake user & fake user will get all the data and capture all the data msg and decrypt all the msg. of the another user.

- Until user A has got this thing and alerts to other user, a pretender is able to read all encrypted msg. of other users.

Publicly Available Directory:

- A dynamic publicly available directory is used to achieve the security. Maintenance and distribution of public directory is controlled by a trust entity.
- This technique is explained as follows and shown in fig.



1) A trusted entity maintains a directory for each user as <name, public key>

2) Each user has to register a public key with the directory.

3) A user can replace the existing key with a new one at any time for any particular reason.

- It is more secure than public announcement but still having some weakness. A hacker can obtain the private key of directory or temper with the info. kept by directory.

Public Key Authority :

- It gives strong security, as shown in fig. a central authority keeps a dynamic directory of public keys of all users. Additional, each user knows the public key of authority.

1) A sends a time stamped msg. to public-key authority containing a request for the current public key of B.

2) Authority responds with a msg. i.e. encrypted using the authority's private key, PR_{auth}. Thus, A is able to decrypt the msg. using the authority's public key,

Therefore, A is assured that the msg. originated with the authority. The msg. includes the following:

↳ B's public key, PU_b, which A can use to encrypt msg. destined for B.

↳ The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority,

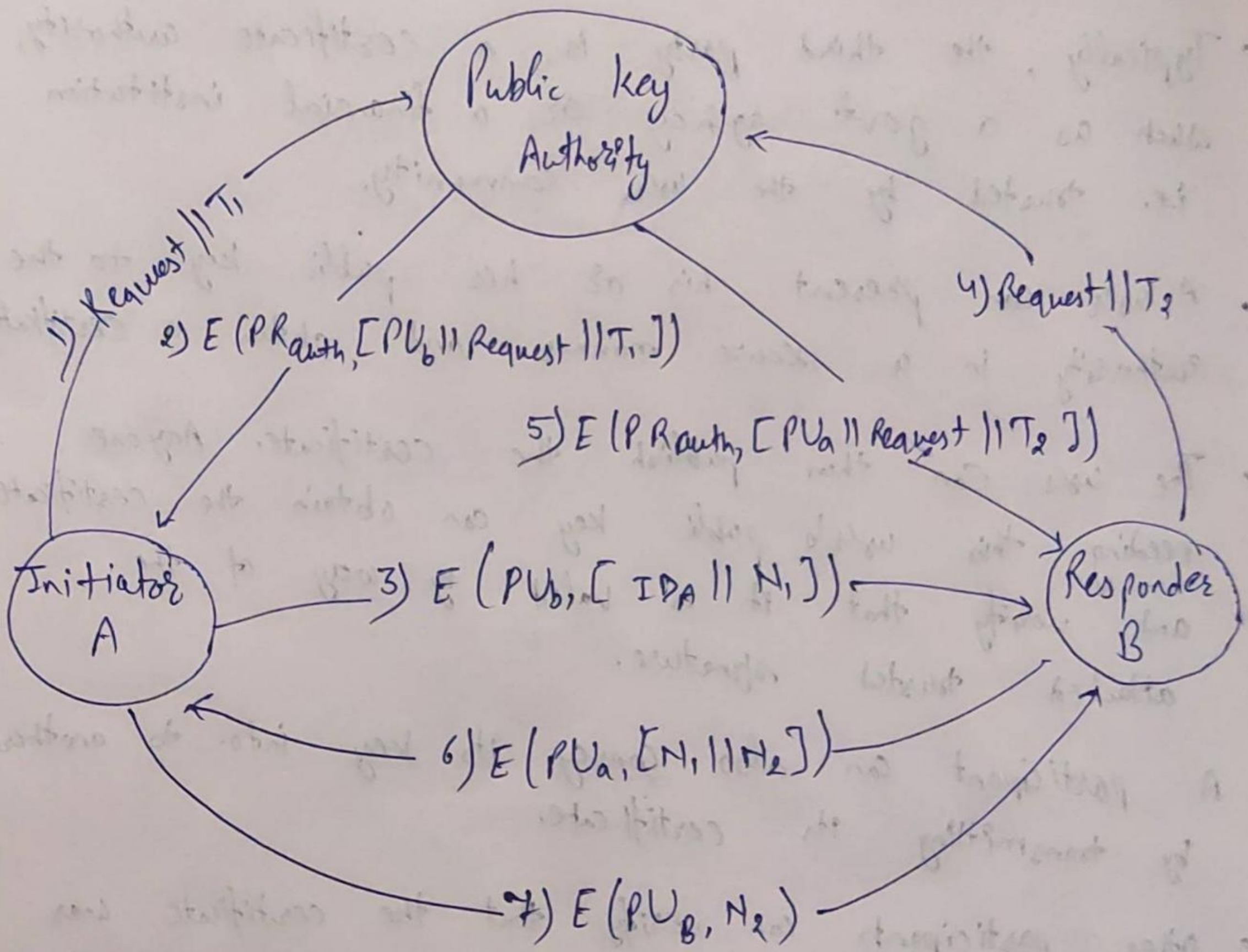
↳ The original time stamp given so A can determine that this is not an old msg. from the authority containing a key other than B's current public key.

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

4) 5) B retrieves A's public key from the authority in the same manner as A received B's public key.

6) B sends a msg 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 B could have decrypted msg. (3), the presence of in msg (6) assure A that the correspondant is B.

7) A return N_2, which is Encrypted using B's public key to assure B that its correspondent is A.



Public Key Certificates:

- The directory of names and public keys maintained by the authority is vulnerable to tempering.
  (It is more secure as compare to the first approach).
- An alternative approach, first suggested by Kohnfelder, is to use certificates.
- In essence, a certificate consists of a public key,

an Identifier of the key owner, and the whole block signed by a trusted third party.
It means PU_a, ID_A ; PR_{auth}. respectively.

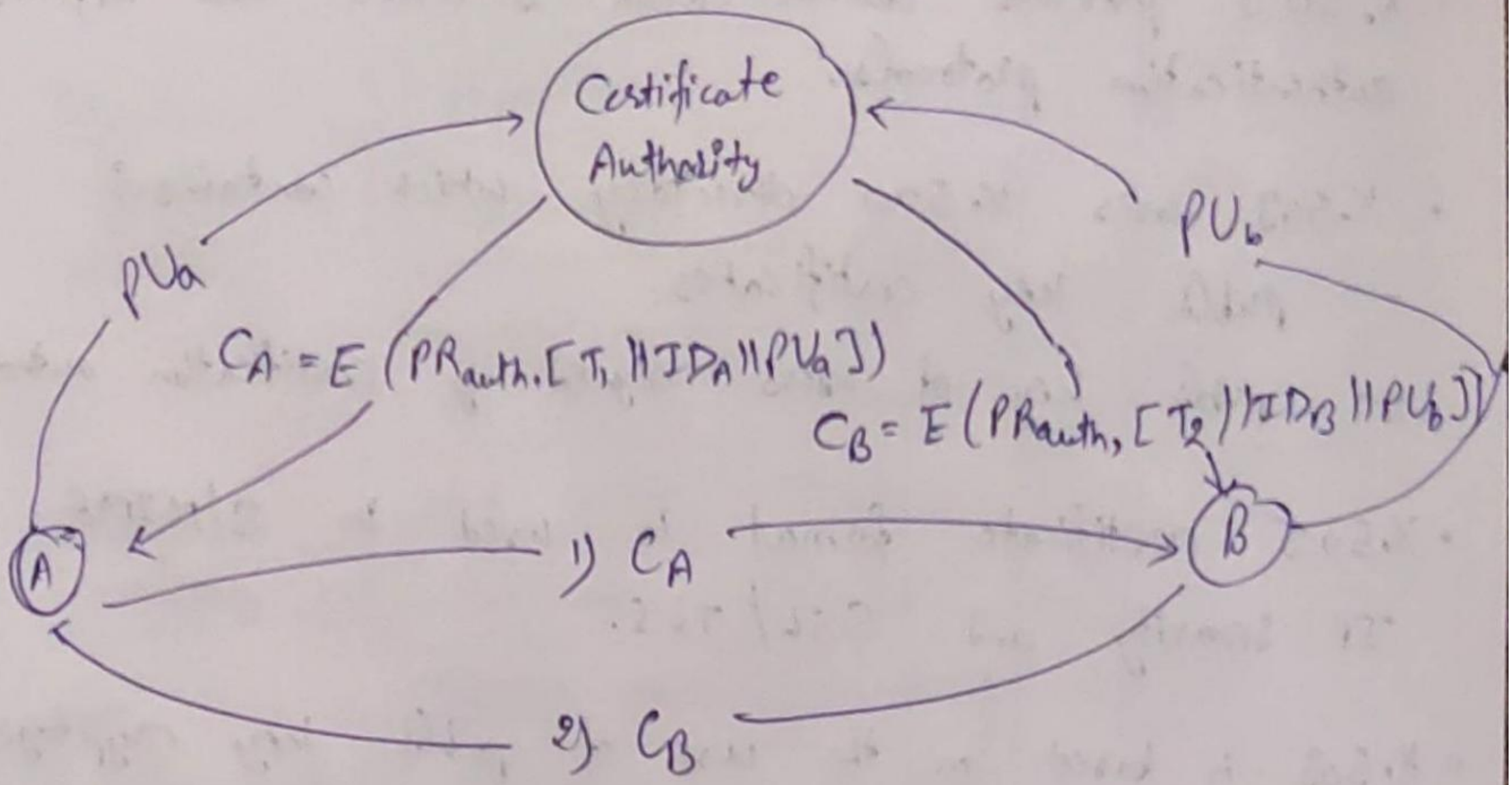
- Typically, the third party is a certificate authority, such as a govt agency or a financial institution i.e. trusted by the user community.
- A user can present his or her public key to the authority in a secure manner and obtain a certificate.
- The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- A participant can also convey its key info. to another by transmitting its certificate
- Other participants can verify that the certificate was created by the authority.
- We can place the following requirements on this scheme:

1) Any participant can read a certificate to determine the name and public key of the certificate's owner. 1)

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.

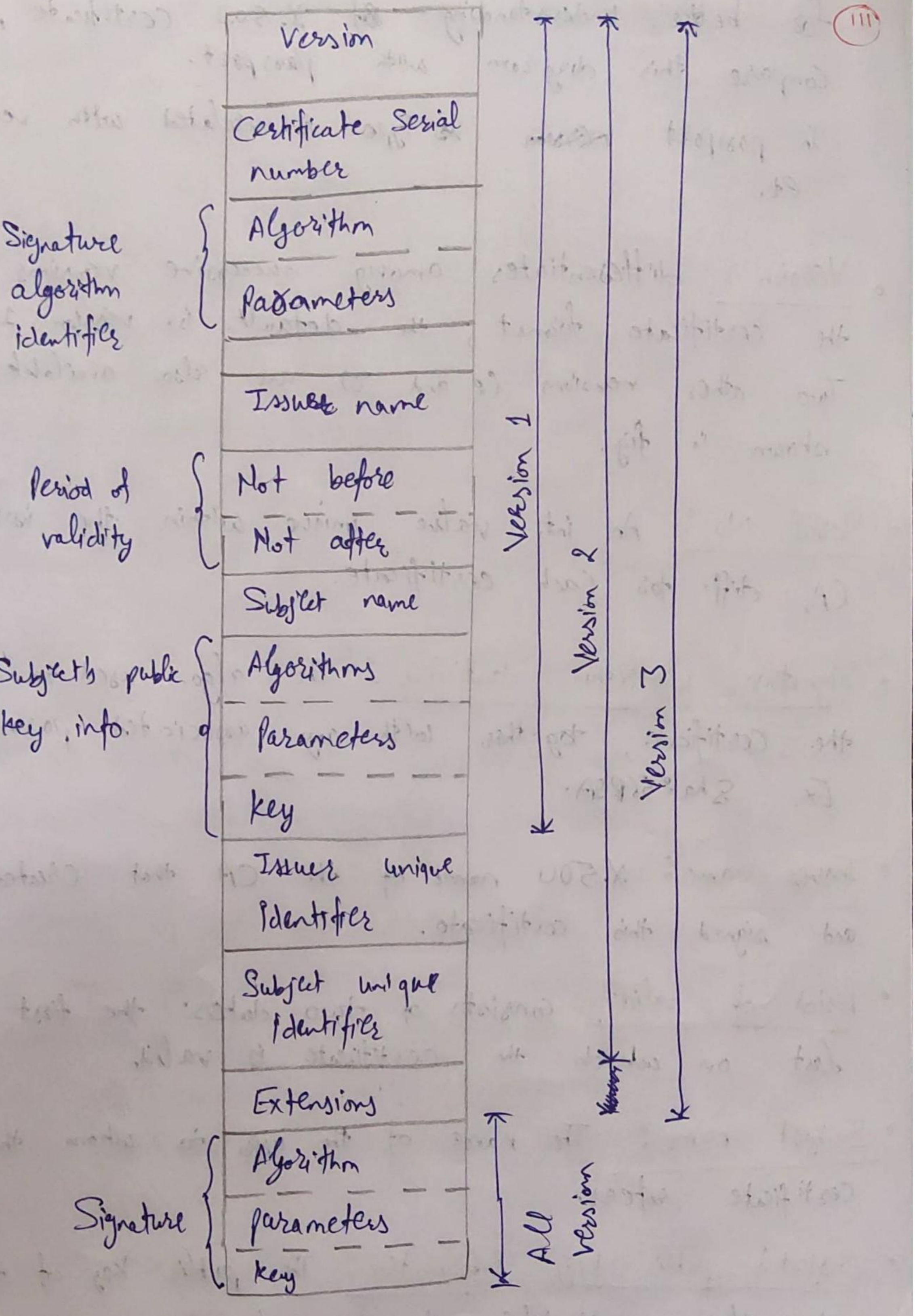
4) Any participant can verify the certificate.



* X.509 Certificate / Digital Certificate :

Introduction :

- X.509 provides authentication services and defines authentication protocols.
- X.509 uses X.500 directory which contains :
  - → public key certificates
  - → public key of users signed by certification authority
- X.509 certificate format is used in S/MIME, IP security and SSL/TLS.
- X.509 is based on the use of public-key cryptography (preferably RSA) and digital Signatures.
- X.509 includes the following elements :
  - → Version
  - → Serial no.
  - → Signature algo. Identifier
  - → Issuer name
  - → period of validity
  - → Subject name
  - → Subject's public-key info
  - → Issuer unique identifier
  - → Subject unique Identifier
  - → Extensions ⟶ Signature



for better understanding of X.509 certificate, Compare this diagram with passport.

In passport ~~version~~ Type is related with version. ex.

- Version : differentiates among successive versions of the certificate format; the default is version 1. Two other versions (2 and 3) are also available as shown in fig.

- Serial No : An int. value unique within the issuing CA, diff. for each certificate.

- Signature algorithm identifier : The algo. used to sign the certificate, together with any associated parameters. Ex. Sha256RSA.

- 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 whom this certificate refers.

- Subject's public-key information : The public key of the sub., plus an identifier of the algo. for which this

is to be used, together with any associated parameters.

- Issuer unique identifier: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for diff. entities.

- Subject unique identifier: An optional bit string field used to identify uniquely the sub. in the event the X.500 name has been reused for diff. entities.

- Extensions: A set of one or more extensions fields.

- Signature: Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algo. identifier.

Purpose of X.509 Certificate:

- The main purpose of Digital Certificate (SSL / TLS certificates), is to identify people and resources over nlw such as the Internet & also to provide secure, confidential communication b/w two parties using Encryption.

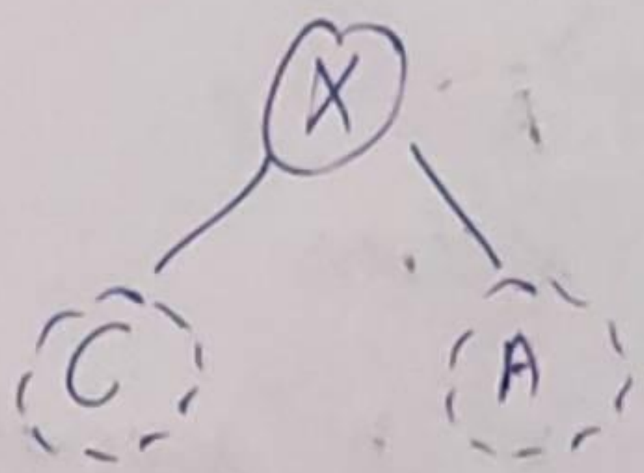
Summary of X.509 certificate:

Version	Version of X.509 to which the certificate conforms.
Serial No.	A no. that uniquely identifies certificates.
Signature Algo. ID	The names of the specific public key algo. that the CA has used to sign the certificate (Eg. RSA with SHA-1)
Issuer (CA) X.500 Name	The identity of CA server who issued the certificate
Validity Period	The period of time for which the certificate is valid with start date and expiration date.
Subject X.500 Name	The owner's identity with X.500 Directory formate.
Subject Public key Info. — Algo. ID	Public key of the owner of the certificate and the specific public key algo. associated with the public key.
Subject Public key Info. — Public key value	
Issuer Unique ID	Info. used to identify issuer of the certificate.
Subject Unique ID	Info. used to identify owner of the certificate.
Extension	Additional info. like alternate name, CRL distribution Point (CDP)
CA digital Signature	The actual digital signature of the CA.

* Obtain & Revocation of Digital (X.509) Certificate:

How to Obtain Digital certificate (X.509 certificate)?

- Any user can verify a certificate if he/she has the public key of the CA that issued the certificate.

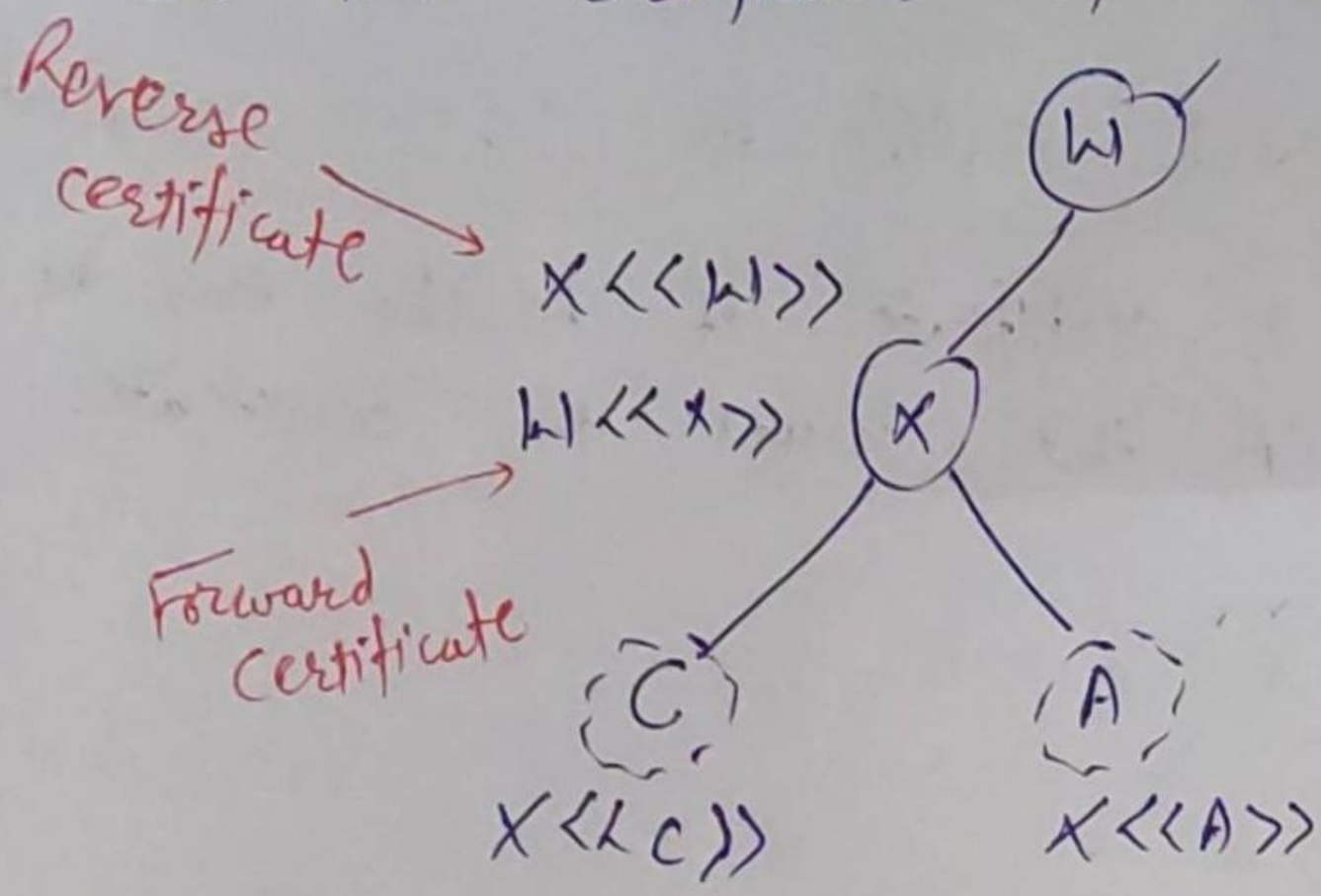


in this fig X is the certification authority and A and C both are the users. for eg. X is share his public key to the certification authority (CA) X then X will generate the certificate of A and send to the user A so X<<A>> is the notation of certificate. and X<<C>> is the certificate of C generated by X.

- Since certificates are unforgeable, they are simply stored in the directory.
- The directory entry for each CA includes two types of certificates:

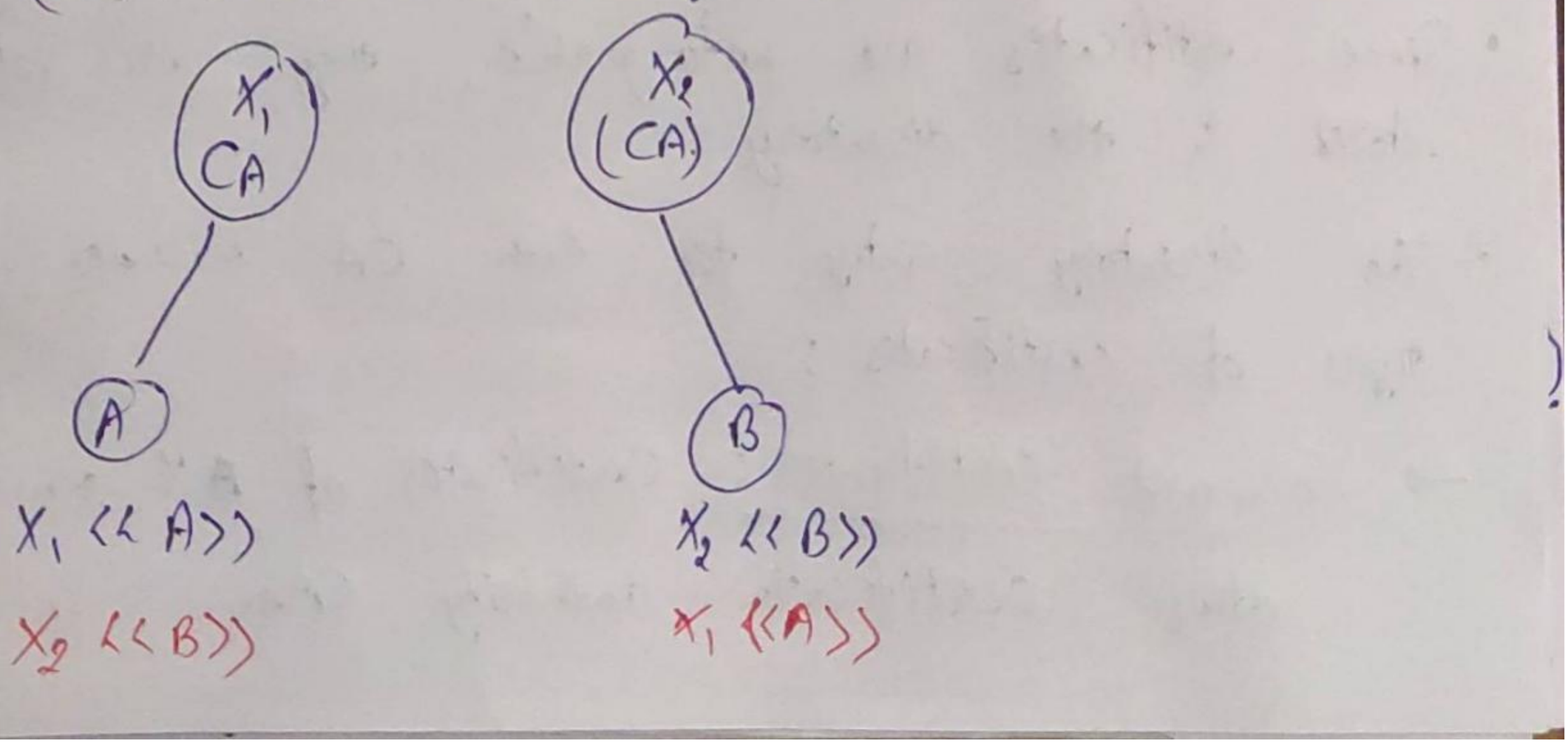
→ Forward certificates: Certificates of X generated by other certification authority CAs.

→ Reverse certificates: certificates generated by X that are the certificates of other CAs.

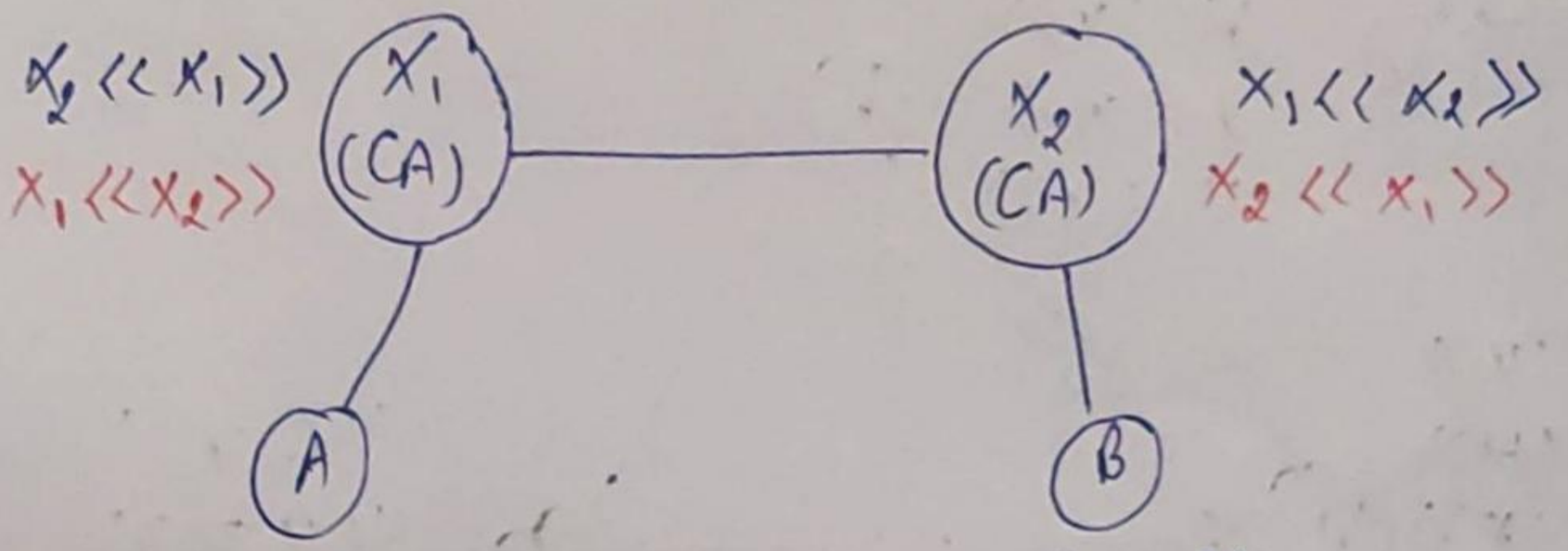


in this fig. certificate X is generated by W, so this is called forward certificate (W<<X>>). and certificate of W is generated by X is called Reverse certificate (X<<W>>).

- User subscribed to same CA can obtain certificate from the directory.
- Suppose, A has obtained a certificate from certification authority (CA) X_1 and B has obtained a certificate from (CA) certification authority X_2.



- A user may directly send the certificate to the other user. (see in the last fig.)
- If A does not know the public key of X_2, then B's certificate, issued by X_2, is useless to A because A can read B's certificate, but A cannot verify the signature.
- However, multiple CAs are there and users subscribed to diff. CAs may want to communicate with each other.
- But if the two CAs have securely exchanged their own public key, the following procedure will enable A to obtain B's public key:



→ $X_2 \langle\langle B \rangle\rangle$ • $X_1 \langle\langle A \rangle\rangle$
→ $X_1 \langle\langle A_2 \rangle\rangle$

→ A obtains the certificate of X_2 signed by X_1 from the directory. A securely knows X_1's public key, so A can obtain X_2's public key from it's certificate and verify X_1's signature on the certificate.

→ A then obtains the certificate of B, signed by X_2. A now has a copy of X_2's public key, so A can verify the signature and securely obtain B's public key.

→ In this case, A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as:

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

- Any level of hierarchy can be followed to produce a chain in this way. For example, in the given fig, A can establish a certification path to B in the following way: