Authentication

# Using Symmetric Encryption

- Using a two-level hierarchy of keys
- Usually with a trusted Key Distribution Center (KDC)
  1. Each party shares its own master key with KDC
  2. KDC generates session keys used for connections between parties.
  3. Master keys are used to distribute these to them

# ~~Needham~~

Protocols proposed in this

1. Needham - Schroeder Protocol

1. It is a original protocol for secret key distribution using a KDC that includes authentication features

2. The protocol can be summarized as follows for a session between A and B mediated by KDC

~~Left A → KDC~~
~~Right side = info of sender & reciever~~
~~Left Right side =~~

The content on :-
1. ~~Right~~ Left side = Info of sender & reciever
2. Right side = Contents of the message
3. || = means concatenation

Rainbow

PAGE:
DATE: / /

Left

1. $A \rightarrow KDC$ : $ID_A || ID_B || N_1$
2. $KDC \rightarrow A$ : $E(K_a, [K_S || ID_B || N_1 || E(K_b, [K_S ||$ $ID_A$
3. $A \rightarrow B$ : $E(K_b, [K_S || ID_A])$
4. $B \rightarrow A$ : $E(K_S, N_2)$
5. $A \rightarrow B$ : $E(K_S, f(N_2))$

Right

here $f()$ is a generic function that modifies the value of the nonce.

Now here,

A & B = Alice & Bob
KDC = Trusted server
$N_1, N_2$ = Nonces which ensure security.
They ① Prevent replay attacks
② Introduce unpredictability
③ Maintain data freshness

$K_a, K_b$ = Secret keys shared between A & KDC and B & KDC

Step 1 :- $A \rightarrow KDC$ : $ID_A || ID_B || N_1$

$ID_A$ & $ID_B$ = Identifier for Alice & Bob
$N_1$ = random nonce generated by Alice

In this step Alice informs KDC that she wants to communicate securely with Bob

Step 2

$KDC \rightarrow A : E(K_a, [K_s || ID_B || N_1 || E(K_b, [K_s || ID_A])])$

$K_a$ & $K_b$ : Symmetric keys shared between Alice (A) & KDC and Bob (B) & KDC.

$K_s$ : Session key to be used for communication between Alice & Bob.

→ This is exactly the purpose of protocol, to distribute a session key $K_s$ to A & B

Here KDC generates a session key $K_s$ and sends it to Alice, encrypted with $K_a$, along with Bob's identifier $ID_B$, the nonce $N_1$ & an encrypted token for Bob ($E(K_b, (K_s, ID_A)$)

Step 3

$A \rightarrow B : E(K_b, [K_s || ID_A])$

Here Alice forwards the encrypted token from the KDC to Bob.

Bob decrypts this using his key $K_b$ to retrieve the session key $K_s$ and Alice's identifier $ID_A$.

Rainbow
PAGE:
DATE: / /

## Step 4

$B \rightarrow A : E(K_S, N_2)$

Bob generates a new nonce $N_2$ to ensure freshness and encrypts it with the session key $K_S$.

Bob sends this to Alice to confirm that he has recieved $K_S$ and authenticated Alice.

## Step 5

$A \rightarrow B : E(K_S, f(N_2))$

Alice applies a function to $N_2$ (nonce) (e.g. incrementing it, hashing it, etc)

Alice encrypts the result using $K_S$ & sends it to Bob as proof that she has the session key and has recieved $N_2$.

Conclusion :-

Now despite all security measures in this protocol, the protocol is still vulnerable to replay attacks if a old session key has been compromised.

So to secure it further some modifications were suggested.

① timestamps (Denning 81)
② Using an extra nonce (Neuman 93)

# Remote user authentication using Assymmetric keys (Public keys)

~~① Public key~~

① This type of authentication has a range of approaches based on the ~~used~~ use of public key encryption.

② This protocol assumes that each of the two parties is in the possession of the current public key of the other party

~~③ This protocol uses~~
③ A protocol using timestamps is provided in Denning [DENN 81].

④ In this protocol, the central system is referred to as an Authentication server (AS)

⑤ The AS is not responsible for secret-key distribution, it rather provides public-key certificates.

Lets continue this in the Denning AS Protoc

PAGE:
DATE: / /

## 1. Denning AS Protocol.

1. $A \rightarrow AS$ : $ID_a \,||\, ID_b$
2. $AS \rightarrow A$ : $E(PR_{as}, [ID_a \,||\, PU_a \,||\, T]) \,||\, E(PR_{as}, [ID_b \,||\, PU_b \,||\, T])$
3. $A \rightarrow B$ : $E(PR_{as}, [ID_a \,||\, PU_a \,||\, T]) \,||\, E(PR_{as}, [ID_b \,||\, PU_b \,||\, T]) \,||\, E(PU_b, E(PR_a, [K_s \,||\, T]))$

(1) Here, the session key is chosen and encrypted by A (Alice), hence no risk of exposure by AS.

(2) The timestamps protect against replays of compromised keys

(3) Now timestamps ~~prot~~ prevent replays but it requireds synchronized clocks

## • One-Way Authentication :-

(1) Example of this type of authentication is email

(2) This type of authentication is required when sender and reciever are not in communication

(3) It involves a single transfer of information from 1 user (A) intended for another user (B) i.e. It establishes identity of A & B & establish that some sort of authentication token was actually generated by A & was

actually intended to sent to B.

④ It has a header in clear so that it can be delivered by an email system.

⑤ Now this method also needs to protect & authenticate the contents of the body & the sender.

→ Use of Symmetric Encryption.

① Here, we can define KDC but we can't have final exchange of nonces.

1. $A \rightarrow KDC : ID_A || ID_B || N_1$
2. $KDC \rightarrow A : E_{K_a}[K_s || ID_B || N_1 || E_{K_b}[K_s || ID_A]]$
3. $A \rightarrow B : E_{K_b}[K_s || ID_A] || K_s[M]$

② This method does not protect against replays

④ It could rely on timestamps in message but email delays make this problematic

→ Use of Assymetric (Public) key

① If confidentiality is a major concern, we can use:

$A \rightarrow B : E_{KU_b}[K_s] || E_{K_s}[M]$

It has encrypted session key and encrypted message.

② If authentication needed to use a digital signature with a digital certificate:

$A \rightarrow B: M \| E_{KRa}[H(M)] \| E_{KRas}[T \| ID_a \| KU_a]$

It has a message, a digital signature and a digital certificate.

# Digital Signature Standard (DSS)

1. US Govt. approved signature scheme FIPS 186
2. It uses a SHA Hash Algorithm.
3. It is designed by NIST and NSA in early 90's.
4. DSS = Standard, DSA = Algorithm.
5. It is ~~an~~ a variant on ElGamal and Schnorr schemes.
6. It creates a 320-bit signature, but with 512 - 1024 bit security
7. Its security depends on difficulty of computing discrete logarithms.

Advantages over RSA

① Smaller (320 bit vs 1024 bit)
② Faster (Much computation is done modulo a 160 bit number)

Rainbow
PAGE:
DATE: / /

# ~~DSA Key Generation~~

# Digital Signature Algorithm

① The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard ~~(FIPS)~~ (FIPS 186), also known as the Digital Signature Algorithm (DSA)

② The DSA makes use of the Secure Hash Algorithm (SHA)

## ⟹ The DSA Approach

① DSA uses an algorithm that is designed to provide only the digital signature function.
② Unlike RSA, it cannot be used for encryption or key exchange.
③ It is a public-key technique

## Algorithm

① Global Public-key Components [p, q, g]

p = prime number ⟹ $2^{L-1} < p < 2^L$
where $512 \leq L \leq 1024$ (bits)
and L is a multiple of 64.
i.e. L is a bit length bet^n 512 & 1024 bits.

q = prime divisor of (p-1), where $2^{N-1} < q < 2^N$
i.e. bit length of N bits

g → generator of a subgroup of order q

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

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

where h (integer) $1 < h < (p-1)$
such that $h^{(p-1)/q} \bmod p > 1$

$\therefore$ $p \neq 1$ should be always the case
or else regenerate.

(2) User's Private Key

$x \Rightarrow$ random integer
where $0 < x < q$

(3) User's Public Key

$y = g^x \bmod p$.

(4) User's Per-Message Secret Number

$K \Rightarrow$ random integer
with $0 < K < q$

(5) Signing

$r = (g^k \bmod p) \bmod q$
$s = [K^{-1}(H(M) + xr)] \bmod q$

Signature = $(r, s)$

Rainbow
PAGE:
DATE: / /

⑥ Verifying

$w = (s')^{-1} \bmod q$
$u_1 = [H(M')w] \bmod q$
$u_2 = (r')w \bmod q$
$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$

TEST : $v = r'$

M = message to be signed.
H(M) = hash of M using SHA-1
M', r', s' = recieved versions of M, r, s

Working of DSA algorithm

1. There are 3 parameters that are common to a group of users. An N-bit prime number q is chosen.
2. Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (p-1)
3. Finally g is chosen to be the form $h^{(p-1)/q} \bmod p$, where h is an integer between 1 and (p-1) with the restriction that g must be greater than 1.
4. With these parameters in hand, each user selects a private key and generates a public key.
5. The private key x must be a number from 1 to (q-1) and should be chosen randomly or pseudorandomly.
6. The public key is calculated from the private key as $y = g^x \bmod p$.

Rainbow

PAGE:
DATE: / /

(7) The signature of a message M consists of the pair of numbers r and s, which are functions of the public key components (p, q, g), the user's private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.

(8) Let M', r', s' be received versions of M, r, s

(9) Verification is performed using the formulas in verification section mentioned before.

(10) The receiver generates a quantity v that is a function of the public components, the sender's public key, the hash code of the incoming message, and the received versions of r and s.

If this quantity matches the r component of the signature, then the signature is validated.

# Authentication Applications.

- It considers authentication functions
- They are developed to support application-level authentication and digital signatures.
- So we consider Kerberos - a private-key authentication service
- Then we consider the X.509 directory authentication service

## 1. Kerberos

1. A trusted key server system from MIT
2. It provides centralised private-key thirdparty authentication in a distributed network
   → a. It allows users access to the services that are distributed through network, w/o needing to trust all workstations
   b. Rather all users just need to trust a central authentication server.
3. There are 2 Kerberos versions in use. Version 4 and version 5.
4. Relies exclusively on symmetric encryption,
5. Kerberos was a 3-headed dog, whose 3 heads represented
   a. the client/principal
   b. The n/w resource which is the application server that provides access to the network.
   c. KDC, which acts as Kerberos's trusted third-party authentication service.

# Kerberos requirements

1. First published report identified its ~~required as~~ requirements as:
   (a) security
   (b) Reliability
   (c) Transparency
   (d) Scalability

2. It is implemented using an authentication protocol based on Needham-Schroeder.

## Kerberos 4 overview

1. It is a third party authentication scheme which mates use of DES
2. As it is difficult to see the needs for ~~its~~ many of its elements, we build up to the full protocol by looking first at several hypothetical ~~dialouges~~ dialogues.
3. Each successive dialogue adds complexity to counter security vulnerabilities revealed in the preceding dialogue.
4. Kerberos 4 using an Authentication Server (A
   a. Here, users initially negotiate with AS to identify itself.
   b. AS provides a non corruptible authentication credential (ticket granting ticket (TGT))
5. Kerberos 4 using a Ticket Granting Server (TGS
   ~~a. It is a basic third party authentication scheme~~

Rainbow

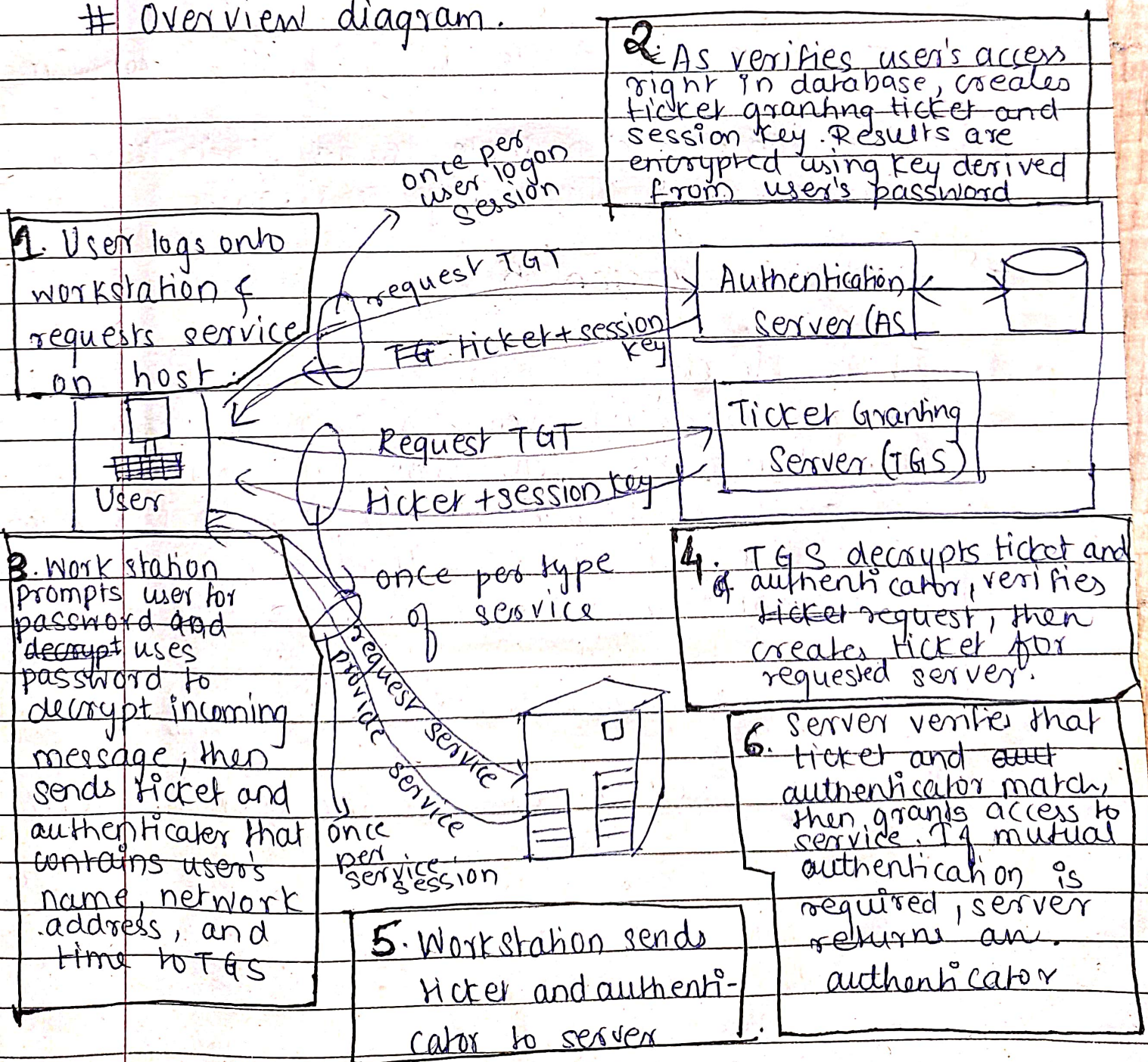
PAGE:
DATE: / . /

a. Users subsequently request access to other services from TGS on basis of user's TGT.

6. The core of Kerberos is the Authentication and Ticket Granting Servers.

- These are trusted by all users and servers and must be securely administered

# Overview diagram.



## Kerberos Realms

1. A kerberos environment consists of
   a. A kerberos server
   b. A number of clients, all registered with server
   c. Application servers, which share key with server.

2. The entire environment is called "a Realm"
   - typically a single administrative domain

3. If there are multiple realms, their kerberos servers must share keys and trust

## # Kerberos Version 5

1. It is developed in mid 1990's
2. It provides improvements over version 4
   a. It addresses environmental shortcoming
      → encryption algo, n/w protocol, byte order, ticket lifetime, authentication forwarding, interrealm authentication.
   b. It also addresses technical deficiencies
3. It is specified as Internet Standard RFC 1510

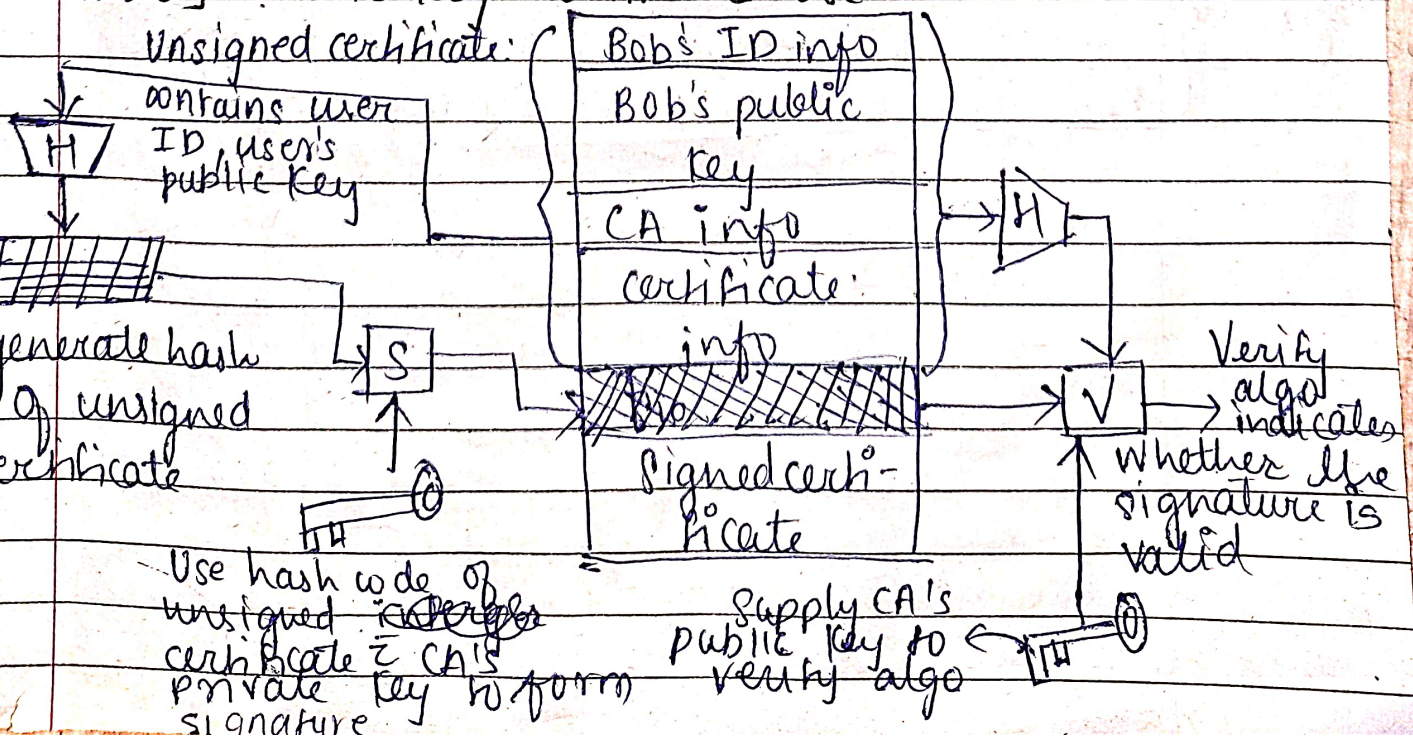
Rainbow

PAGE:
DATE: / /

# X.509 Authentication Service

1. This service is a part of CCITT X.500 directory service standards
2. The directory is a server or distributed set of servers that maintain a database of information about users
3. X.509 defines a framework for the provision of authentication services by the X.500 directory to ~~the u~~ its users.
4. The directory may serve as a repository of public-key certificates
   a. Each certificate contains the public key of the user and is assigned with the private key of a trusted certification authority
5. X.509 also defines alternative authentication protocols based on the use of public-key certificate
6. X.509 is based on the use of public-key cryptography and digital signatures. It does not use standardised algorithms but RSA is recommended

# X.509 Public key Certificate Use

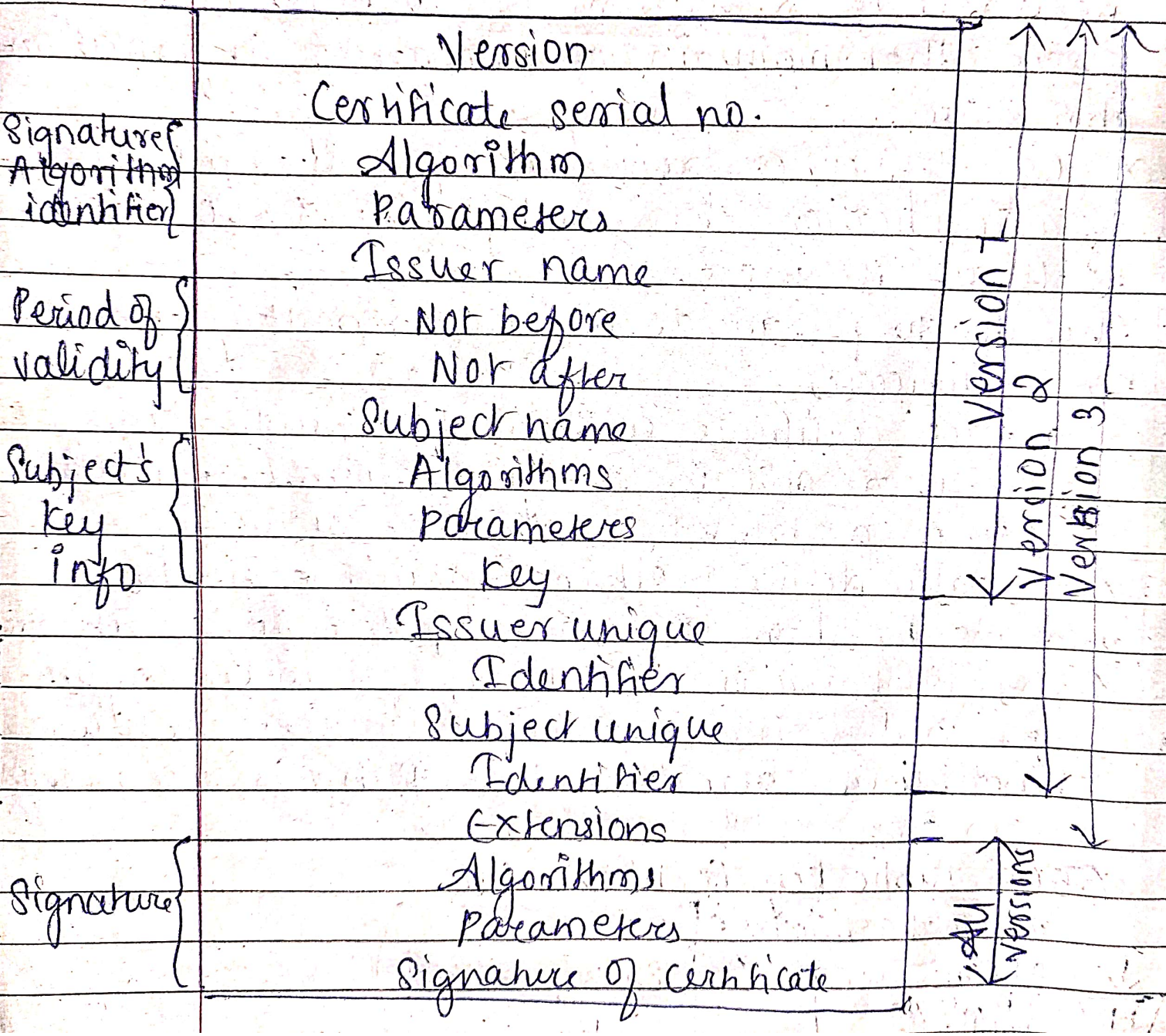


# CA = Certification Authority Certificate

Rainbow
PAGE:
DATE: / /

## X.509 Formats

a) X.509 certificate



Rainbow

PAGE:
DATE: / /

b) Certificate revocation list.

| | |
|---|---|
| Signature algo identifier | Algorithm<br>Parameters |
| | Issuer name |
| | This update date |
| | Next update date |
| Revoked certificate | User certificate serial # |
| | Revocation date |
| | . |
| | . |
| | . |
| Revoked certificate | User certificate serial # |
| | Revocation date |
| Signature | Algorithms |
| | Parameters |
| | Signature of certificate |

# Why revocation of a certificate

1. The user's private key is assumed to be compromised
2. The user is no longer certified by the CA
3. The CA's certificate is assumed to be compromised

Obtaining a certificate

(1) Any user with access to a CA can get any certificate from it.
(2) Only the CA can modify the certificate
(3) Because the certificates cannot be forged, the certificates can be placed in a public directory.

CA Hierarchy

1. If both users share a common CA then they are assumed to know its public key
2. Otherwise CA's must form a hierarchy.
3. It uses certificates linking members of hierarchy to validate other CA's
   a. Each CA has certificate for clients (forward) and parent (backwards)
   b. Each client trusts parent certificate

Principles of CA hierarchy

1. Shared CA Public key
   - If two users trust the same CA, they can authenticate each other using the CA's public key
   - The public key of CA is known to the users beforehand.

Rainbow
PAGE:
DATE: / /

2. CA Hierarchy
- If users trust different CAs, those CAs must establish a trust relationship
- This is done through a hierarchy of CAs, where higher-level (parent) CAs certify lower level (child) CAs.
- The hierarchy ensures that all entities under different CAs can still verify each other using cross certification.

3. Forward & Backward Certificates

a. Forward certificates are issued by CA's to its clients
b. Backward certificate is held by a CA issued by its parent

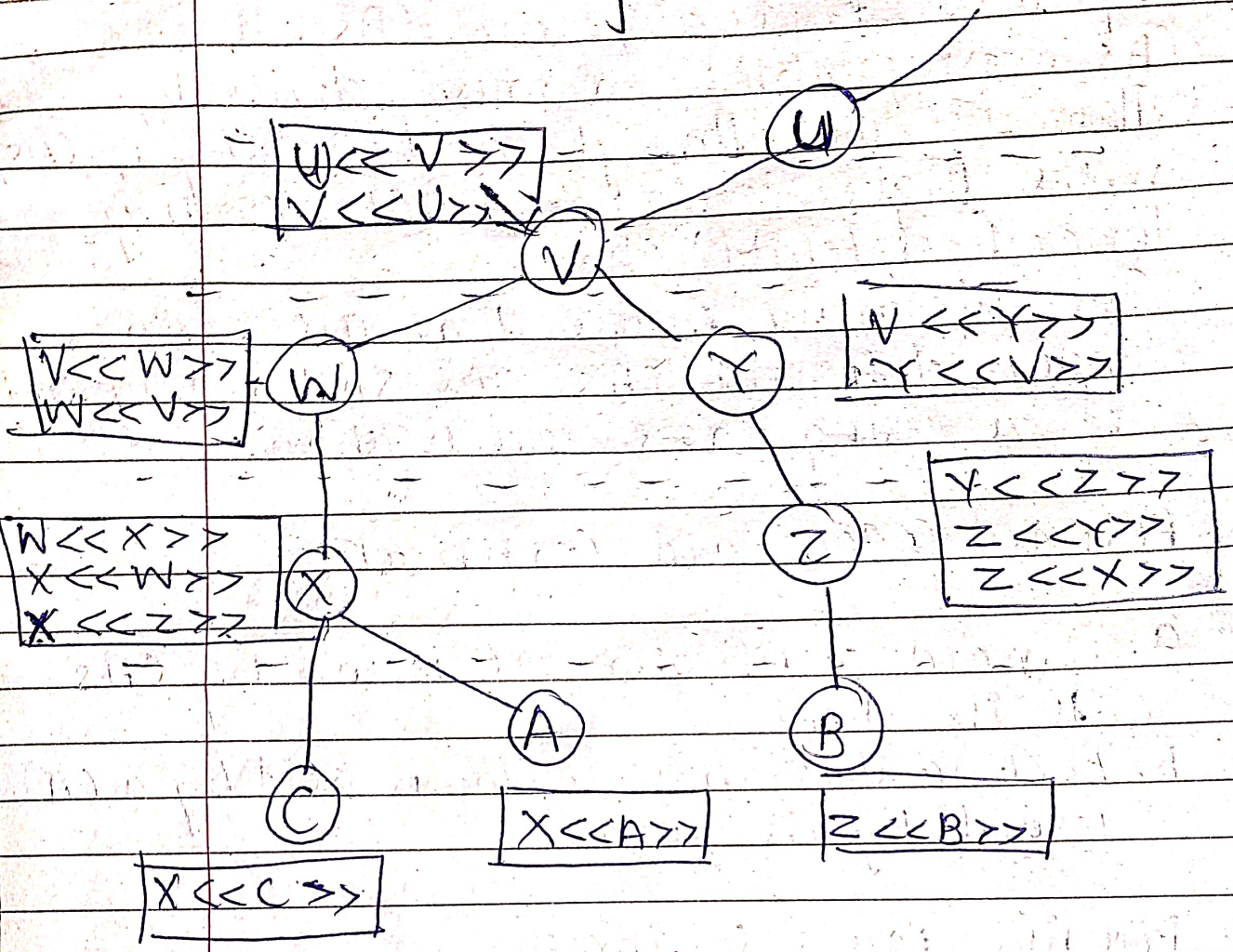
4. Trusting Parent Certificates:

a. Each client trusts the certificate of its CA's parent in the hierarchy

5. Cross-Verification

a. Any certificate issued by one CA can be verified by other users of another CA because their certificate chain can be traced upward to a common trusted root CA.

CA hierarchy Use.




Here X<<C>>

↓

means that X has issued a certificate for C.

~~2 to~~ types

1. Issued to child
2. Cross certificate for parent
3. Cross certificate for same level node

PAGE:
DATE: / /

# Certificate Revocation

1. Certificates have some period of validity
2. They may need to be revoked before expiry for :-
   a) User's private key is compromised
   b) User is no longer certified by this CA
   c) CA's certificate is compromised
3. CA's maintain list of revoked certificates
   i.e. The Certificate Revocation List (CRL)
   - Users should check certificates with CA's CRL

# Authentication Procedures:

X.509 includes 3 alternative authentication procedures:

① One-Way Authentication
② Two-Way Authentication
③ Three-Way Authentication

all use public-key signatures.

## 1. One-Way Authentication

- 1 message $[A \rightarrow B]$ is used to establish
  ① identity of A and that the message is from A
  ② that the message was intended for B
  ③ integrity and originality of message

- message must include
  (1) a timestamp
  (2) nonce
  (3) B's identity
  (4) is signed by A

2. Two-Way Authentication

- 2 Messages [(A → B, B → A)] which also establishes in addition to one way:-
  (1) Identity of B and that the reply is from B
  (2) That the reply is intended for A
  (3) Integrity and originality of reply.

- Reply includes original nonce from A, also timestamp and nonce from B.

3. Three-Way Authentication:

- 3 messages (A → B, B → A, A → B) which enables above authentication W/O synchronized clocks.

In addition to one-way & two-way, it also has
a. has reply from A back to B containing signed copy of nonce from B.

- This means that timestamps need not be checked or relied upon.

Rainbow
PAGE:
DATE: / /

# X.509 : Version 3

1. Why the need?
- It has been recognised that additional info is needed in a certificate
  e.g. email/URL, policy details, usage constraints

2. Rather than explicitly naming new fields, they defined a general extension method

3. Extension consists of:
   (1) Extension Identifier
   (2) Criticality indicator
   (3) Extension value.

# Certificate Constraints

1. The key and policy information:
   → it conveys info about subject & issuer keys, plus indicators of certificate policy.

2. Certificate subject and issuer attributes
   → they support alternative formats for certificate subject and/or issuer.

3. Certificate path constraints
   - allows constraints on use of certificates by other CAs

Rainbow

PAGE :
DATE : / /

# Additional Notes according to class

1] Kerberos vs Firewall
2] Difference between Kerberos 4 and 5
3] Kerberos brute force (sweep) detection)
4] SSL vs Kerberos
5] X.509 authentication service diagram
6] Advantages of CA over KDC
7] PKI Infrastructure
   - Issuance diagram
   - Usage diagram
8] Types of Certificates
   ① Organizational
   ② Residential
   ③ Personal
9] PKI Architecture
   ① Central Trust based
   ② Hierarchial based
   ③ Mesh based
   ④ Bridge based