# Week 8

Lecture 1

**Key Management (Public Key)**

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8

# Key Management (Public Key)

## COMP90043

## Lecture 1



Udaya Parampalli

## Lecture 1

1.1 Public Key Management

- Public Key Address and Distribution
- Four different methods
- Public Key Authority
- Public Key Certificates and Revocation.
- Public Key Infrastructure

# Recap

The Figure Illustrates the notations
And use of Public Key functions;
We will use this notation
throughout this semester.

Public key of B : $Pu_b$
Private key of B : $PR_b$

Encryption and Decryption by A

$$Y = E(PU_b, X)$$

$$X := D(PR_b, Y)$$

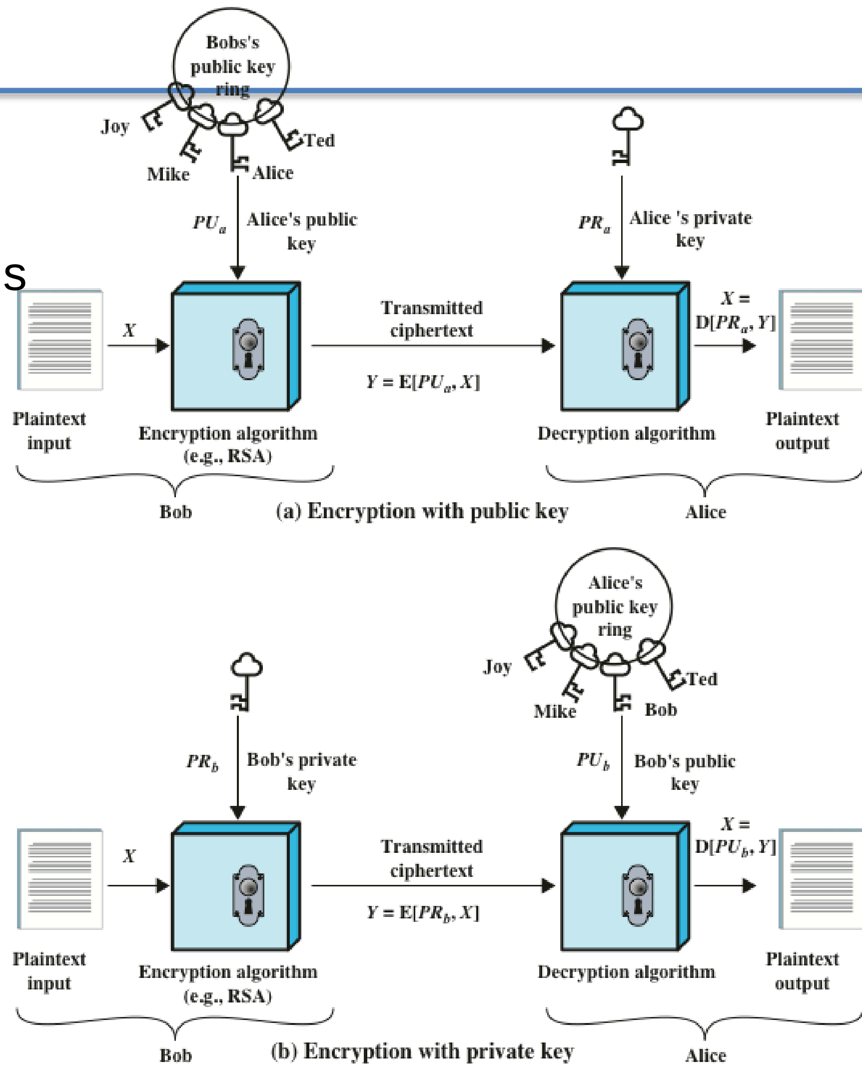



**Figure 9.1 Public-Key Cryptography**

# Public Key Address

- We saw an example of an RSA key:
- RSA-768: a 768-bit RSA modulus with 232-digit decimal representation:

*n = 1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413.*

*e = 11304953579770080757973560987543678999542356780987667899995573263034537315482685079170977477086534597989047753567946978708 13*

- Can you make out if this belongs to an identity that you are familiar?
- In fact, the value looks random and we cannot conclude anything.
- In practice, let us look at the structure of the public key formatting we considered so far.
- If Alice has a public key PU, then we can represent as
- Alice : < IDA=Alice, $Pu_a$ = (n,e)>.
- If this is in public domain, anyone can make a modification: for eg:

- Trudy : < IDA=Trudy, $Pu_t$ = (n,e)>.
- How do you ascertain the correct identity?
- This is the authentication problem. This lecture will look into these issues.

# Public Key Distribution Problem

- How does Alice advertise her public key so that Bob and others can use it to encrypt information to her?
  - Alice : < IDA=Alice, $Pu_a$ = (n,e)>.
- Note that the above format may appear specific to RSA, but we can extend the idea by including explicit information about public parameters
- Alice : < IDA=Alice, $Pu_a$ = (n,e), Algorithm Public Parameters>.

- Even if Alice signs the public address, as long as the public address is authenticated no one can believe that it belongs to Alice.

- Because as we saw before, any one can replace with a new public key and signature and masquerade as Alice.
  - Alice : < IDA=Alice, $Pu_a$ = (n',e')>.

# Public Key Distribution

- Stallings discusses four important methods:
  - Through Public announcement
  - By Using publicly available directory
  - With Public-key authority
  - Using Public-key certificates
- Most of the existing methods can be mapped to one of the above.

# Notation

- We use the conventions associated with RSA schemes while explaining public key protocols.
- Public Address: PU Private Address: PR
- Public Key Encryption/Decryption:
  - Encryption: E(PU,M) = C;
  - Decryption: M = E(PR,C)
- Public Key Signature/Verification
- Signing:
  - s = E(PR,M); (M,s) is a signature pair
- Verification
  - M eq E(PU,s)?

NOTE: the notation E(key, message) is used for symmetric key encryption also; the meaning depends on the context.

# Through Public announcement

- A simple strategy, users distribute to those who need by any means (broadcasting or email etc)
- Example: PGP keys
- Main issue is that they can be easily forged as we explained before.


**Figure 14.10 Uncontrolled Public Key Distribution**

From the textbook Fig 14.10

# By Using publicly available directory

- A directory service is established,
- Each user contacts the directory through secure means and places his public address to be downloaded by other users.
- Each user can update his public key and details. Think, why do you nee[d] this feature?
- Sometime keys may be compromised.
- Users can contact the directory electronically.
- Security is better than the previous method, but still vulnerable.

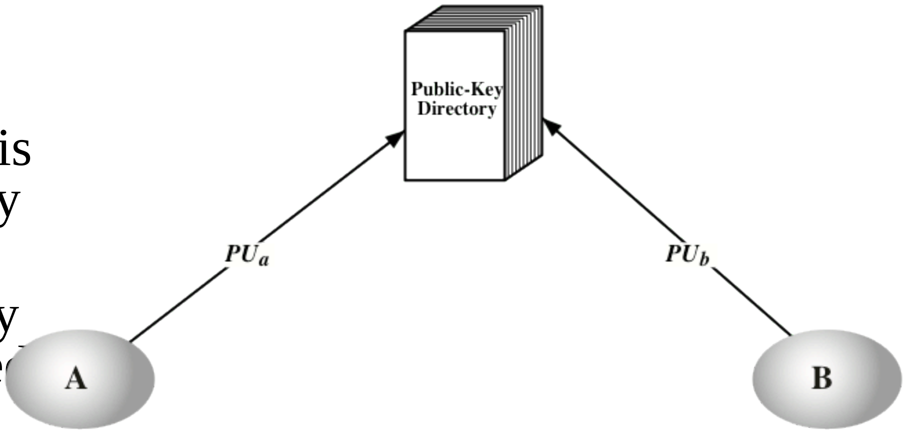



Figure 14.11 Public Key Publication

From the textbook Fig 14.11

# With Public-key authority

- This method is a further improvement to the directory service. It has following properties:
- The authority server is always online with tight control over the distribution and maintenance of keys.
- Authority also has a public and private key: $<PU_{auth}, PR_{auth}>$
- Users will contact the authority whenever they need key service.
- Issues:
  - Server needs to be online always.
  - Still there is a possibility of tampering and attacks.

- Next, we discuss the protocol as in the textbook:



Udaya Parampalli

School of Computing and Information Systems

# Public-Key Authority: A simple scenario

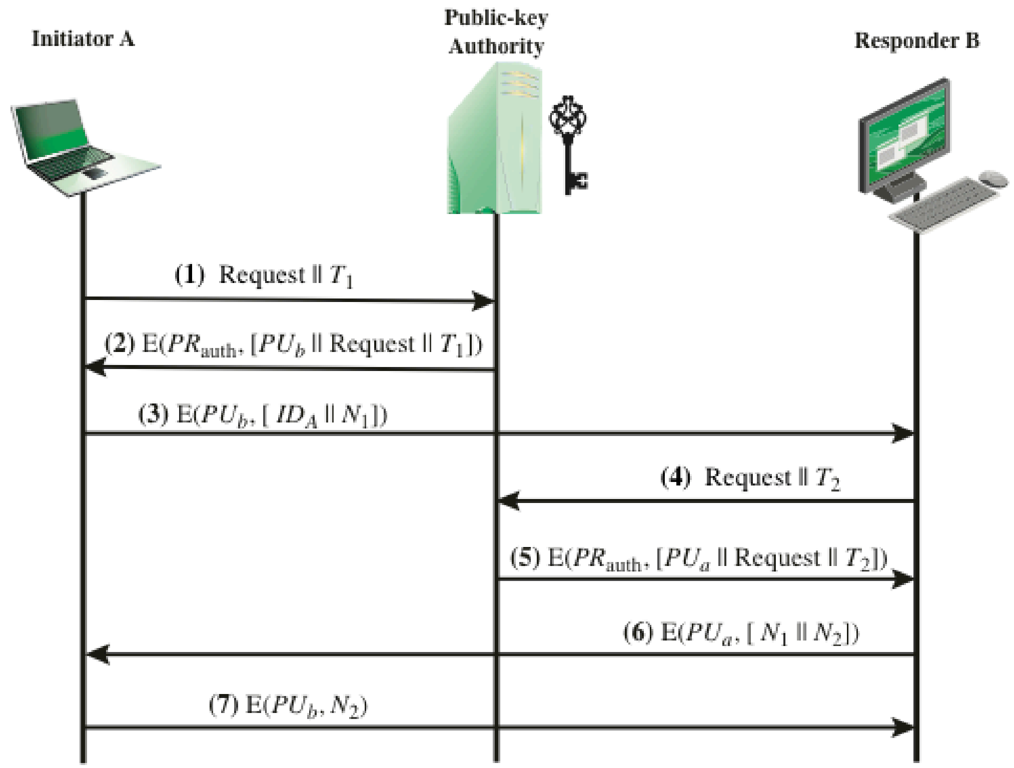



**Figure 14.12 Public-Key Distribution Scenario**

From the textbook Fig 14.12

# Using Public-key certificates

- This is the most sophisticated method, first suggested by Kohnfelder. Users get real access to keys.
- Here, key authority need not be online all the time, at least theoretically.
- What is a certificate?
- A form which binds identity of a users with its public key.
- The method allows others to verify the validity of the certificates.

- A certificate should have minimum of this form:

- Alice : < IDA=Alice, $PU_a$ , Signature of $PU_a$ by the Authority, $PU_{auth}$ >.

# Exchange of Certificates

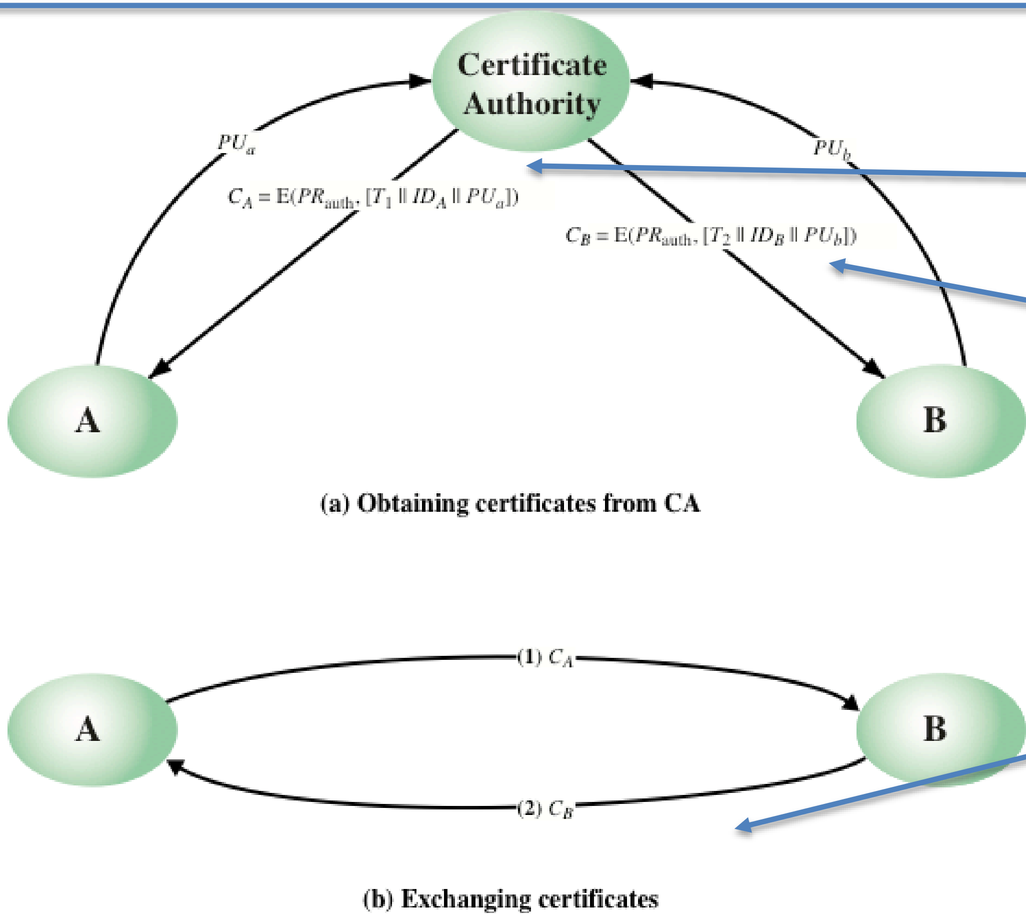


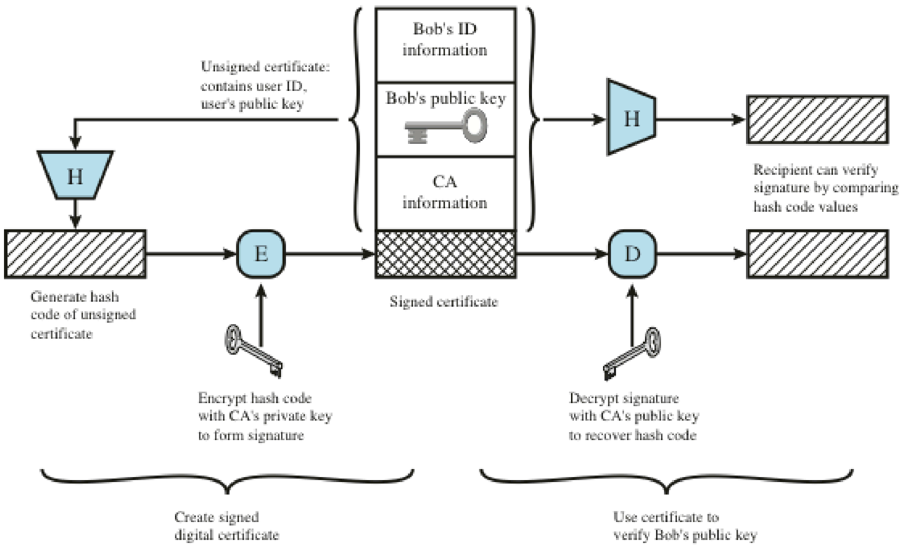

**Figure 14.13 Exchange of Public-Key Certificates**

Look at the structure of the certificate,
we follow this method in workshops and exam

Since A and B already have a trusted relation with the authority, the public Keys are implicitly authenticated After the exchange.

From the textbook Fig 14.13

# X.509 Certificates




Read Section 14.4 for the details of X.509 certificates

From the textbook Fig 14.14

**Figure 14.14 Public-Key Certificate Use**

# X.509 Structure

- The standard notation for a certificate of:

CA<<A>>

= CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}.

- with the meaning CA signs the certificate for user A with its private key.

- Please refer to a small document that I uploaded for the X.509 and the PKI Infrastructure,

- Version
- Serial number
- Signature algorithm identifier
- Issuer name
- Period of validity
- Subject name
- Subject's public-key information
- Issuer unique identifier
- Subject unique identifier
- Extensions
- Signature

# Certificate Advantages

- When Bob receives a certificate from Alice, how does he know that is authentic?
- $C_A$ : < IDA=Alice, $PU_a$ , Signature of $PU_a$ by the Authority, $PU_{auth}$ >.
- In the absence of any other information, he is still in the dark as he was when Alice have him pubic key directly.
- However, now he can also obtain a certificate from the authority for his public key:
- $C_B$ : < IDA=Bob, $PU_a$ , Signature of $PU_b$ by the Authority, $PU_{auth}$ >.
- Now, when he received $C_a$, he can verify that the authority is same as in his certificate ($C_{AB}$).
- Then he can verify the signature of ($PU_a$) by using the public key of $PU_{auth}$ found on his certificate, thus clearly establishing the authenticity of Alice's public key.

# Ah problem again!

- Are we now solved the problem of authentication of public key?
- As long as Alice's certificate has not changed or compromised he would be fine.
- But situations could change at Alice's side: certificate gets expired, compromised or Alice wants to change the public key.
- In such situations, you are now back to square one.

- **How does this problem can be solved?**

- A simplest way for Bob to determine somehow that the certificate he received is still valid or not.
- This is achieved by what is knows as "revocation list" maintained by the authority.
- Hang on, you wanted to solve the problem of authority not being a bottle neck but now authority still need to back again online.
- However, this service is only required for a fraction of users not everyone.

# Revocation

- Revocation of certificates is a very important practical problem that Industry is faced with.
- This has resulted in a large business to maintain public infrastructure.
- Certificate maintenance and verification is an important topic. We will look at the issues in one of the workshop problems.

- Also, not all users share a same CA, then we need to solve the problem of verifying certificates issues by different CA's.

- These are achieved through an organization of CA's in hierarchical fashion and each CA certify other CA's.
- Stallings explanation is given in the next slide. More details can be obtained from Additional material I have placed on LMS.

# CA Hierarchy Use

**Forward certificates:** Certificates of X generated by other CAs, and
**Reverse certificates**
Certificates generated by X that are the certificates of other CAs.

A acquires B certificate using chain:
X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

B acquires A certificate using chain:
Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

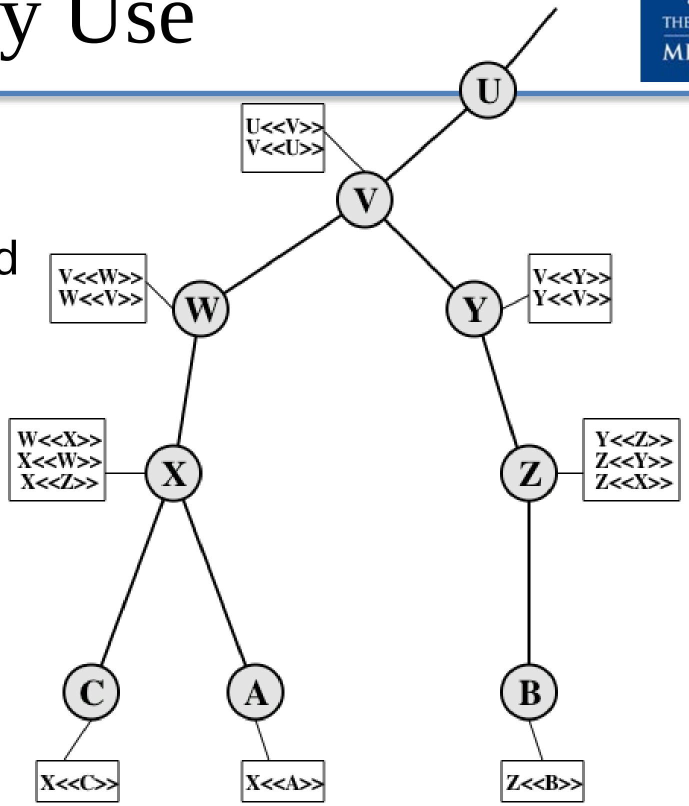
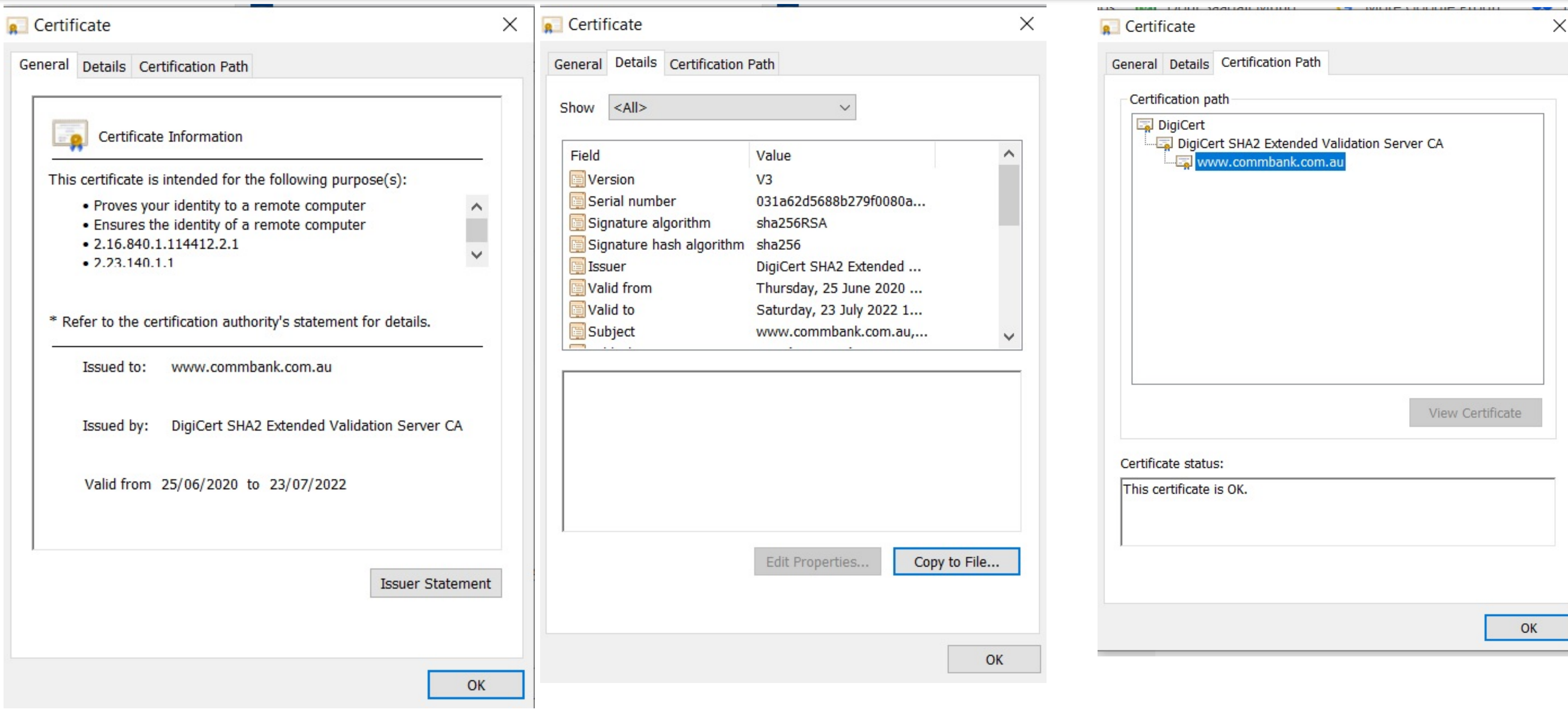



**Figure 14.16 X.509 CA Hierarchy: a Hypothetical Example**

From the textbook Fig 14.16

# Example



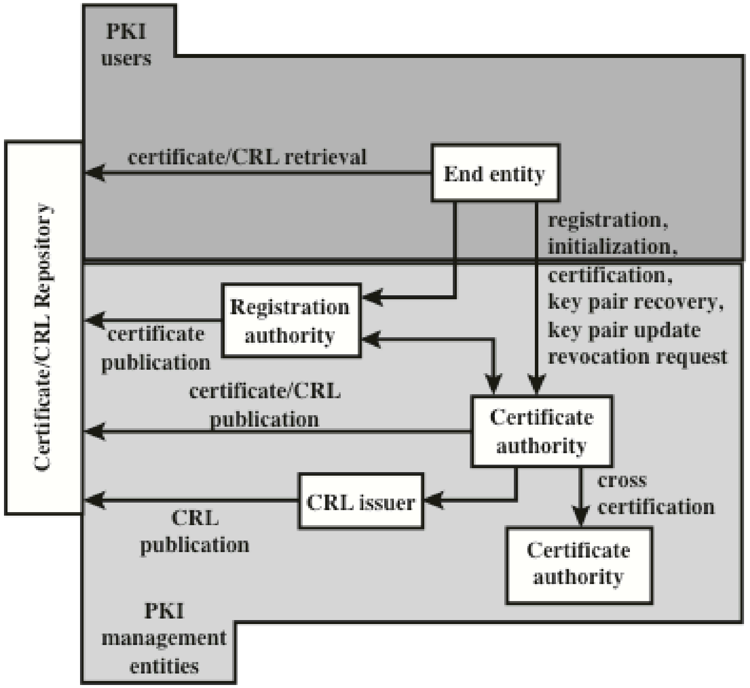




Udaya Parampalli

# Public Key Infrastructure

- **The textbook discusses the issues in detail, please refer to Section 14.5**




From the textbook Fig 14.17

**Figure 14.17 PKIX Architectural Model**

# Week 8

Lecture 1

**Key Management (Public Key)**

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8