ASYMMETRIC AUTHENTICATION

DR. O. I. ADELAIYE

OUTLINE

- Using Asymmetric Crypto. For Authentication
- Distributing Public Keys
- X.509 Certificates

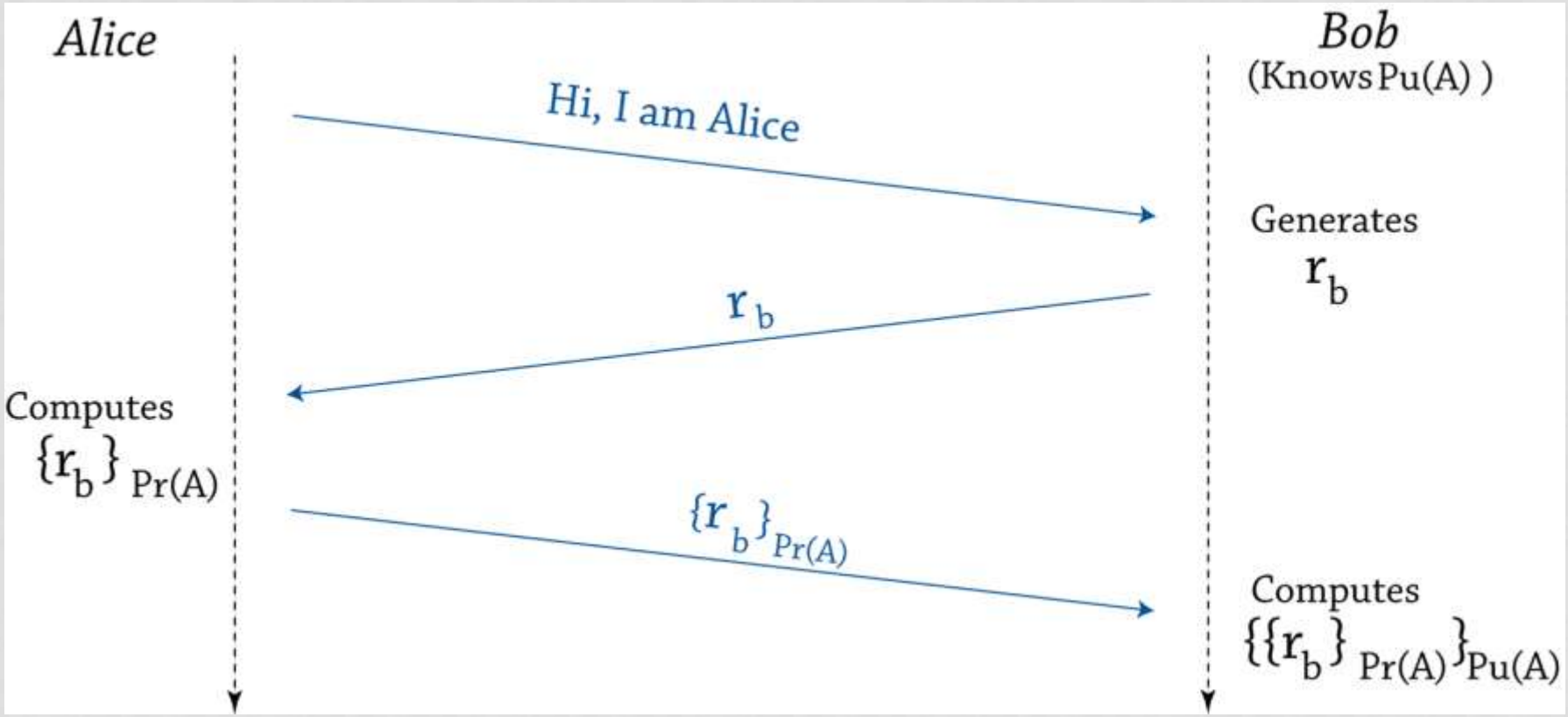
ASYMMETRIC AUTHENTICATION

- Keys are generated as a pair
  - Mathematically related, but not possible to derive one from the other
- One key is known by only one entity: private key
- The other is known by everyone: public key
- Either keys can be used for encryption
  - The other is used for decryption
- Using public keys for encryption ensures confidentiality
- Using private keys for encryption ensures authentication

AUTHENTICATION

- Asymmetric Cryptography
  - Each user has a private and a public key
  - Bob needs to know the public key of Alice in order to authenticate her
- Encrypting using a Private key: **Signature**
  - Only Alice can make it,
  - Every body can verify it

ASYMMETRIC ENCRYPTION FOR AUTHENTICATION

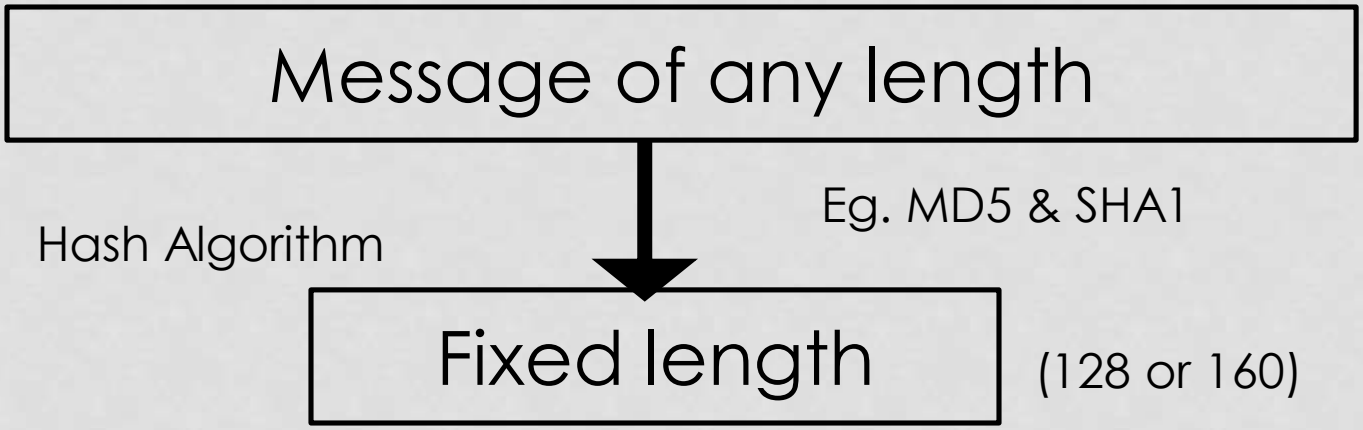


- Pu(A) (resp. Pr(A)) is Alice public (resp. private) key

ASYMMETRIC ENCRYPTION FOR AUTHENTICATION

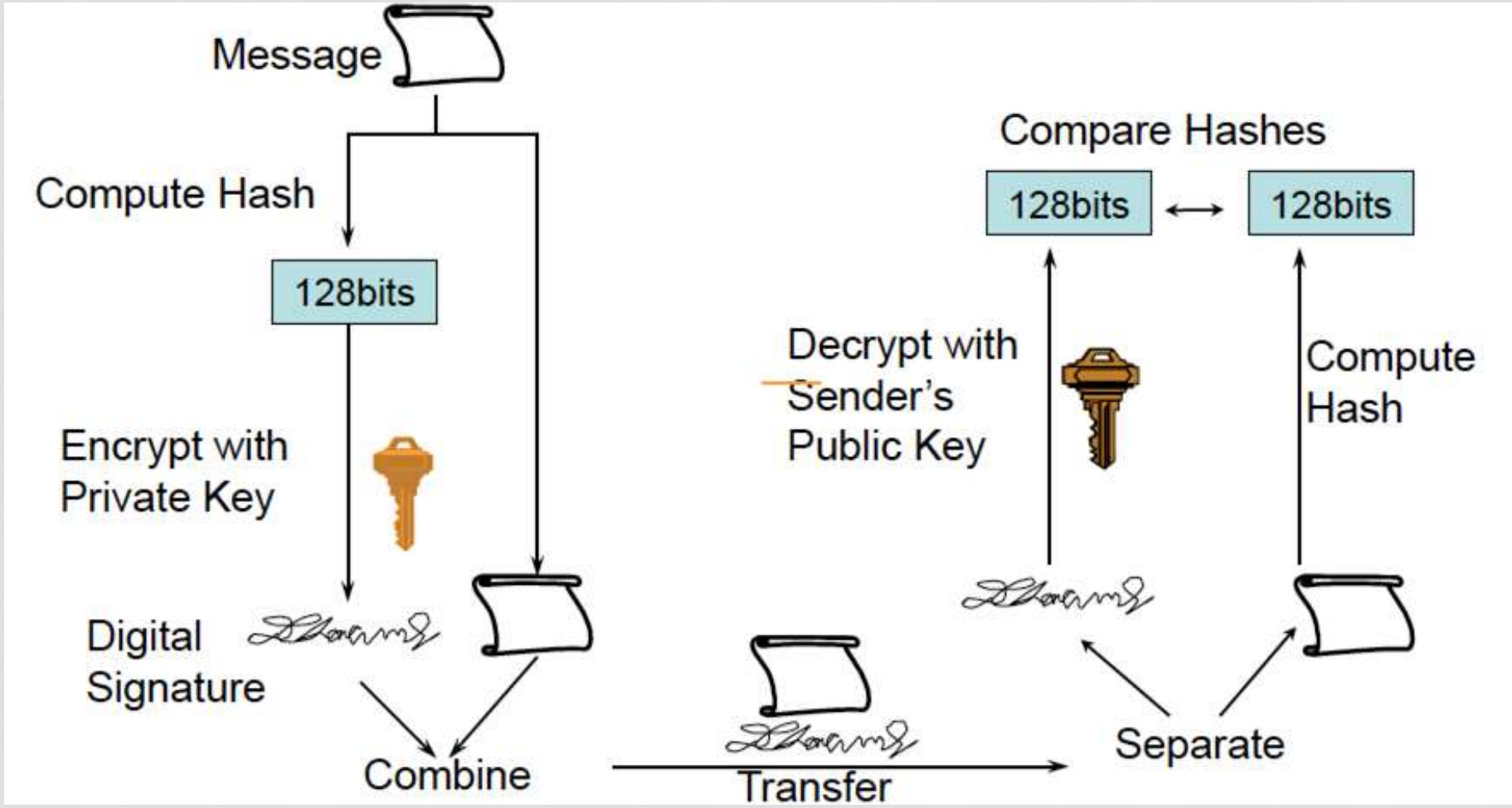
- Advantage
  - Even if Alice wants to prove her identity to more than one user, she only needs to know her private key If Alice uses symmetric encryption, she needs to generate as many keys as authenticators
- Problem
  - Poor performance, asymmetric encryption is more processor and time consuming than symmetric encryption and hash techniques
- Solution
  - Sign a summary, or a "hash" of the message

HASH??



- Every message produces a different hash
- Given a hash you cannot find the message
- "Digital fingerprint" of the message

DIGITAL SIGNATURES



STORAGE OF PRIVATE KEYS

- The signature is trusted if and only if no one except its owner can produce it
- The private key can be stored
  - In an encrypted file, protected by a password
  - In a smart card, protected by a password or PIN

DISTRIBUTING PUBLIC KEYS

- In a context where each user may need to prove his identity to others
  - Each user needs to know the public keys of the other users
- Problem: How to distribute those keys
  - In a trustable way?
    - How to be sure that K is Alice's public key?
  - Efficiently?
    - Each user needs to be able to know the public keys of all users she may authenticate
    - Give Bob the public key of Alice only when he needs to verify her identity

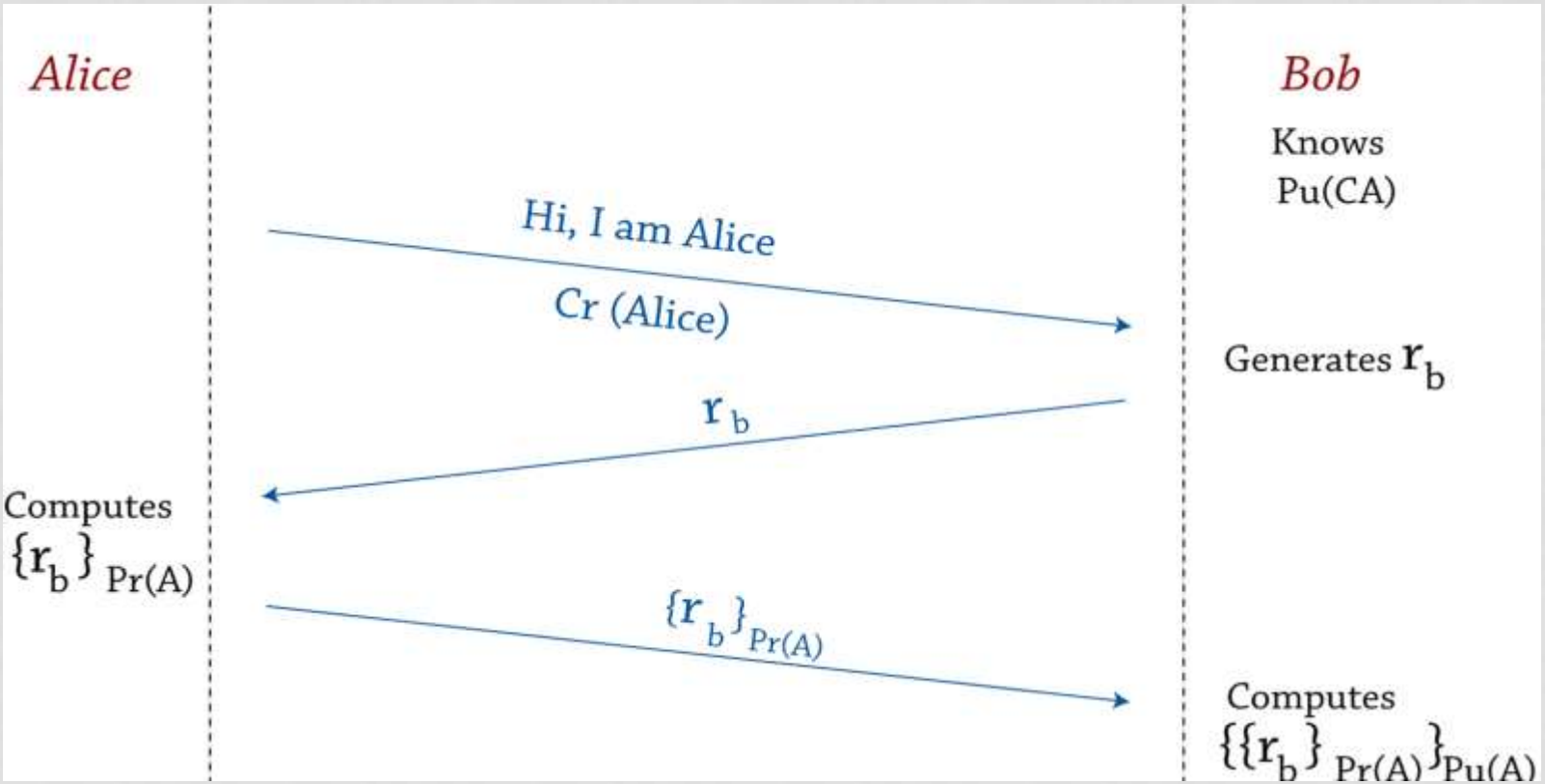
DISTRIBUTING PUBLIC KEYS

- Three ways to distribute public keys:
  1. Personally exchange public keys: not scalable
  2. Get a public key from someone you trust (trusted introducer): not scalable
  3. Get a **certified** key from a public repository: dynamic, scalable

DISTRIBUTING PUBLIC KEYS

- Certificates
  - Trust a third party
  - Every one knows its public key
  - The trustable entity *certifies* that K is the public key of Alice
  - Alice provides her certificate to Bob when he verifies her identity
  - Public key certificates cannot be altered without detection
- In practice
  - The trustable entity is called *Certification Authority (CA)*
  - Each user knows the public key of the CA
  - The CA generates a certificate for each user containing its name (*Alice*) and its key (*Pu(Alice)*)

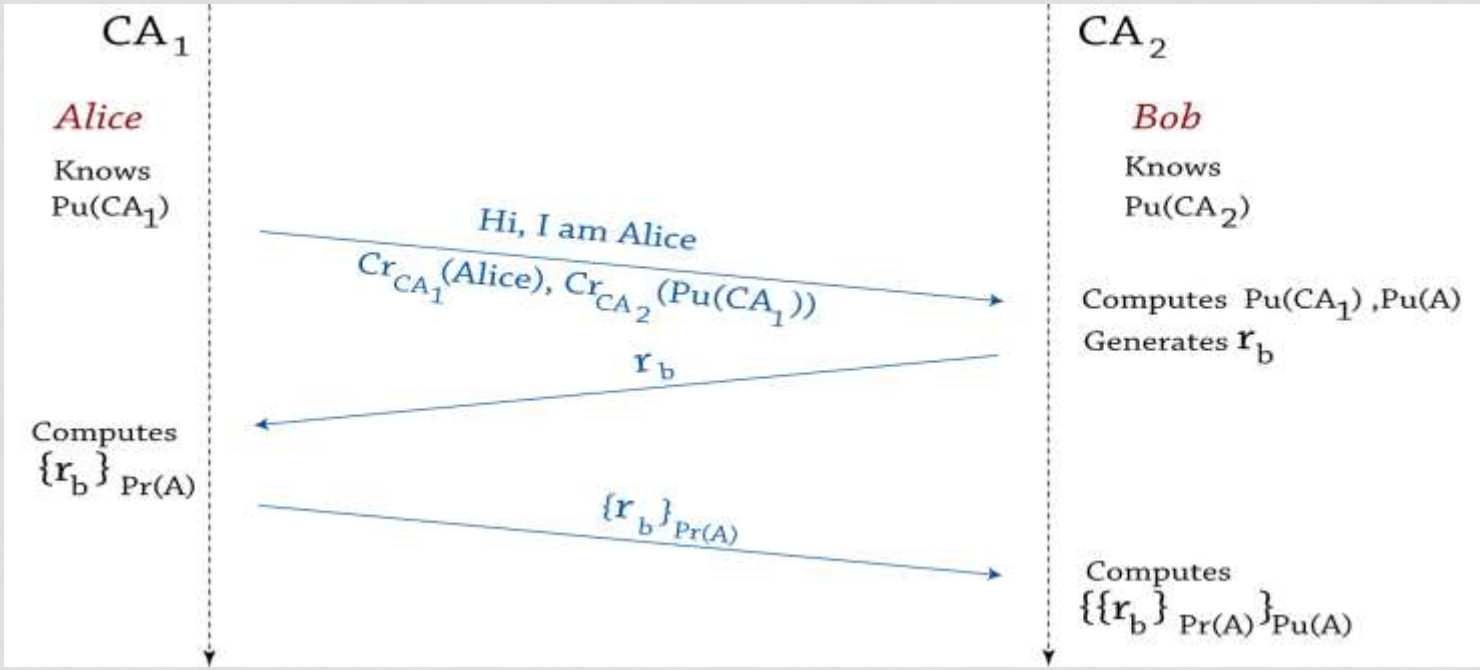
CERTIFICATES



- Why does Bob need the challenge? What happens if it is Alice who generates the challenge

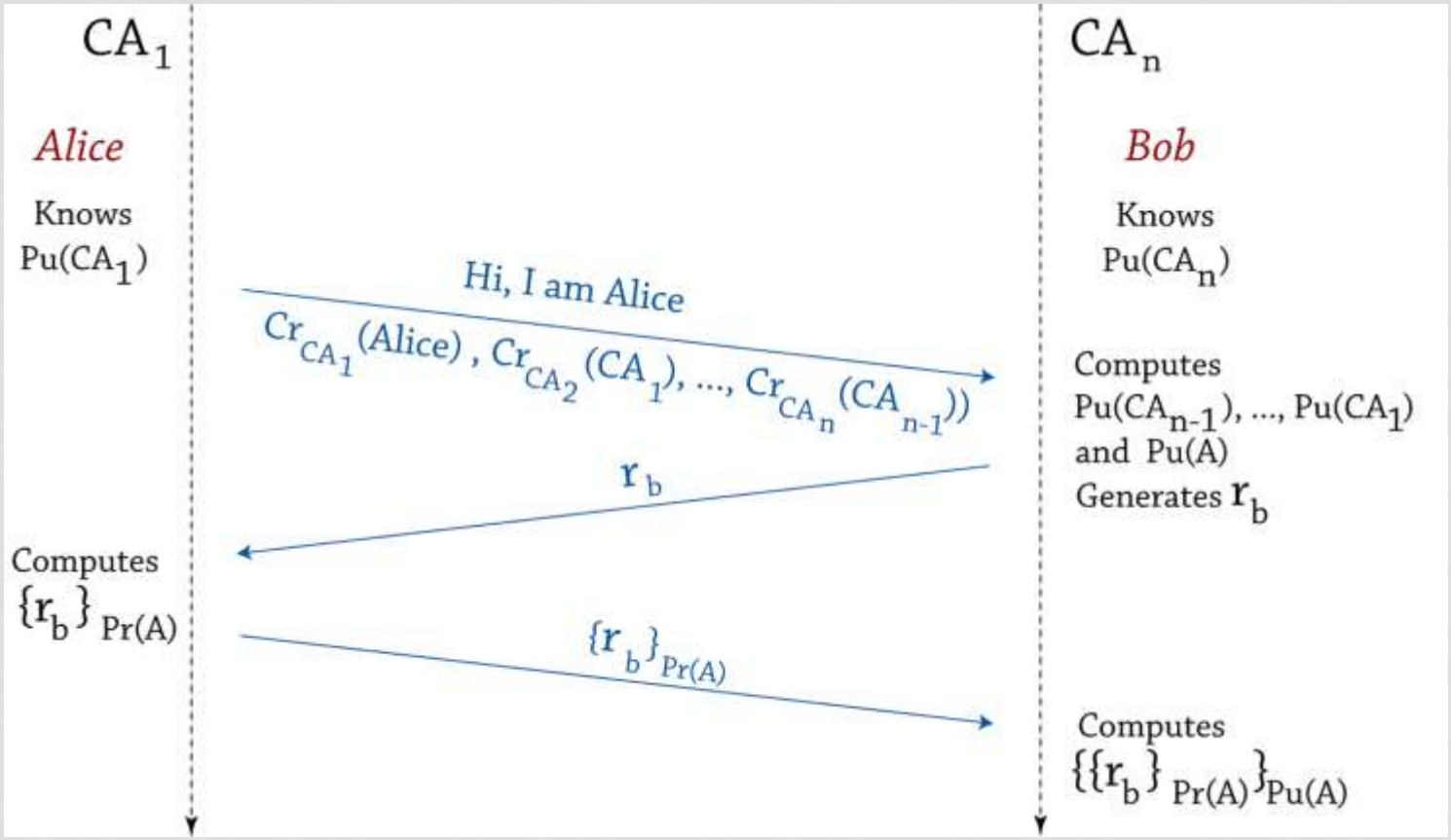
CERTIFICATES

- In practice, there is no universally trusted entity (guess why?!): users are divided into domains
  - Domains are managed by different CA
- How to allow Alice (CA_1) to prove her identity to Bob (CA_2)?
  - CA_2 signs a certificate for CA_1



CERTIFICATES

- More generally, certificate chains
  - CA_n is the *trust anchor*



CERTIFICATE FORMATS

- X.509 format
  - Widely accepted international standard format
  - Used by Microsoft, Verisign, etc.
  - Used by S/MIME email
  - Signed by a single Certification Authority that has a globally unique name
- PGP format
  - Allows multiple owner identities for a key
  - Allows multiple certifiers (CAs) for a key
  - User certifies his own key
  - Anyone else can also be a certifier
    - a user or a CA

TRUST MODELS

- Monopoly Model (Centralized): one universally trusted entity
  - With Registration Authorities (RA): to check identities
  - With delegated CA: can issue certificates
- Oligarchy: many trust anchors
  - Used in Browsers
- Anarchy (Web of Trust): anyone can sign a certificate for anyone else
  - Pretty Good Privacy (PGP)

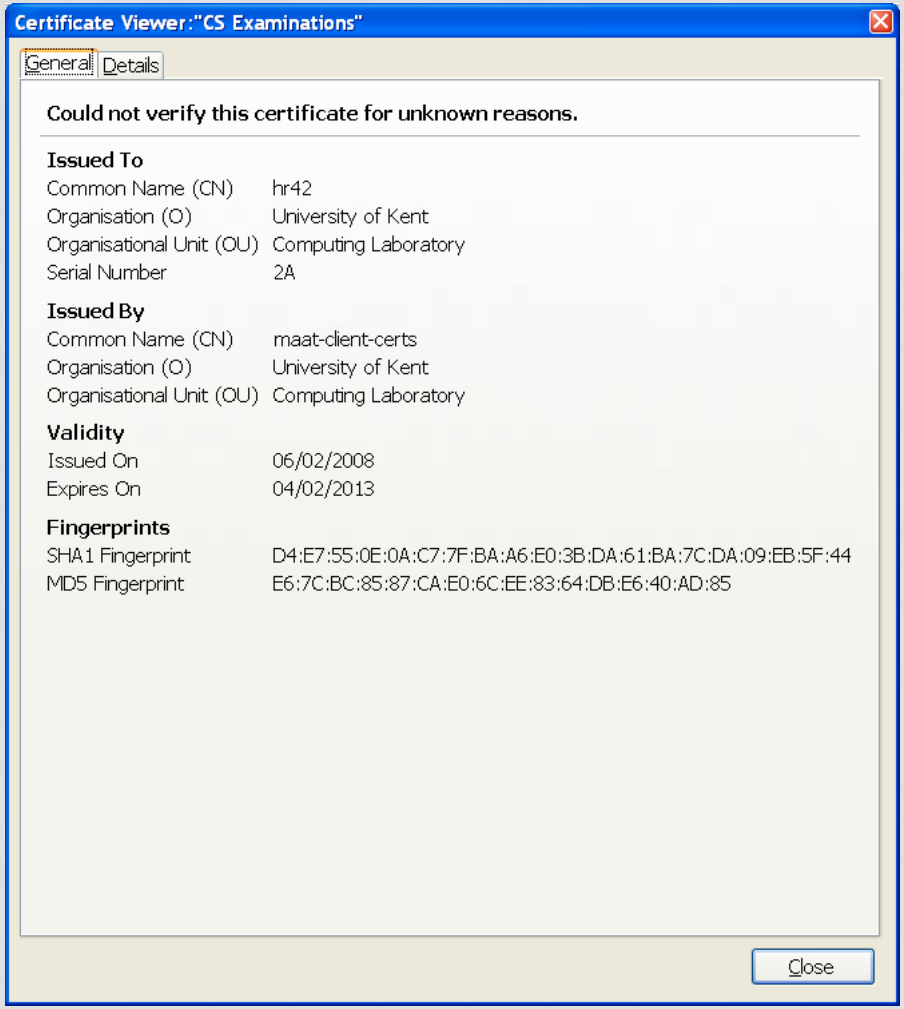
X.509 CERTIFICATES

PUBLIC KEY CERTIFICATE CONTENTS

- Name/Identifying information of the key pair owner
- The public key
- Name of the authority that vouches for this binding
- Validity period of the certificate
- All the above Digitally Signed by the certifying authority

X.509 CERTIFICATES

- My X.509 Certificate
  - Displayed in Firefox
  - Used to prove my identity to the exams system



X.509 CERTIFICATES

- Moving X.509 Certificates and key pairs between applications
  - **Certificates and key pairs** can be moved in standard X.509 binary format (**DER** or **Base64** encoded)
  - **PKCS#12** is another standard format for moving **key pairs** (private keys and public key certificates) between applications e.g. Mozilla and IE
    - This file is encrypted and protected by a user provided PW
- **PKCS#7** is a standard format for moving public key **certificates** (and certificate chains to the root CA) between applications
- Use the Import and Export capabilities of the applications

CERTIFICATE ISSUANCE

- X.509
  - A CA issues certificates to its users and to subordinate CAs
- PGP
  - User issues her own self signed certificate
  - Anyone else may choose to certify it by adding her signature on the certificate

CERTIFICATE REVOCATION

- Certificates are good for a predefined period
- It may happen that a certificate revocation is necessary
  - Private key is stolen
  - An employee is fired
  - A user forgets her password
- In X.509: Similar to credit cards revocation
  - Old technique: publish a list of revoked cards (Certificates Revocation List)
  - New technique: check the validity of the card on-line (On-line Certificate Status Protocol)
- In PGP: each signer can revoke her signature

WHO CAN PERFORM REVOCATION?

- X.509 – only the CA can revoke the certificates it has issued
  - Revocation can be requested by the user, the CA administrator, or other trusted entity
- PGP - key signers can revoke their individual signatures on a public key
- PGP – only the key owner can revoke her own public key
  - This was a problem if you forgot your private key password, so in PGP 6 you can specify a designated revoker to act on your behalf

DISTRIBUTION OF REVOCATION INFORMATION

- X.509
  - CRLs are published and distributed in the same way as the certificates, and by storing in LDAP directories and on Web pages
- PKI X.509 (PKIX) group
  - Defined an Online Certificate Status Protocol so that a relying party can query an OCSP server to see if a certificate is valid. This is similar to how credit cards are checked by shopkeepers today.
- PGP
  - Key signers and key owners should send their revoked signatures to key servers and to their PGP friends

END