CS Revision Lecture 11, 13

- ## Lecture 11 - Secure communication: public key infrastructure
  - ### Public keys

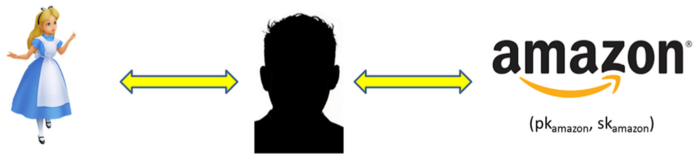
    

    Figure: How does Alice trust that pk_{Amazon} is Amazon's public key?

    - Public-key encryption schemes are secure only if the authenticity of the public key is assured
  - ### Distribution of public keys
    - Public announcements - participants broadcast their public key
      - **does not defend against forgeries**
    - Publicly available directories - participants publish their public key on public directories
      - **does not defend against forgeries**
    - Public-key authority - participants contact the authority for each public key it needs
      - **bottleneck in the system**
    - Public-key certificates - CAs issue certificates to participants on their public key
      - **as reliable as public-key authority but avoiding the bottleneck**
  - ### Public key certificates

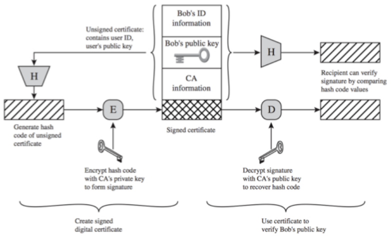


Figure: image from Cryptography and Network Security - Principles and Practice - William Stallings

- A certificate consists mainly of
  - a **public key**
  - a **subject** identifying the owner of the key
  - a **signature** by the CA on the key and the subject binding them together
    - **The CA is trusted**

- X.509 certificates

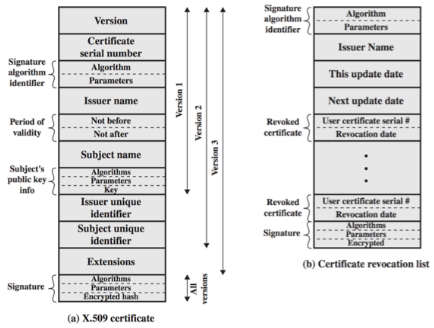


Figure: image from Cryptography and Network Security - Principles and Practice - William Stallings

  - X.509 defines a framework for the provision of authentication services
  - Used by many applications such as TLS

- Public key certificates

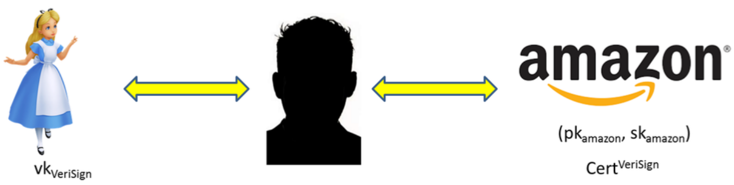


Figure: Alice can now verify Amazon's certificate

- China of trust

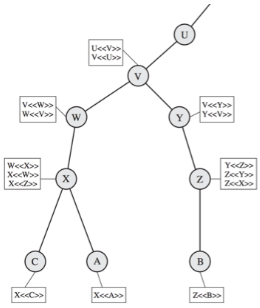


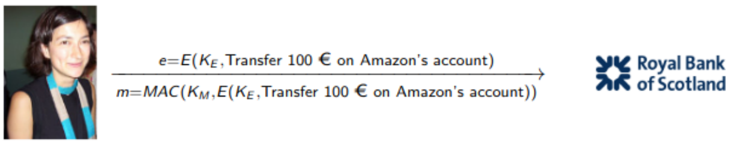
Figure: X.509 Hierarchy - image from Cryptography and Network Security - Principles and Practice - William Stallings

- Having a single CA sign all certificates is not practical
- Instead a root CA signs certificates for level 1 CAs, level 1 CAs sign certificates for level 2 CAs, etc.

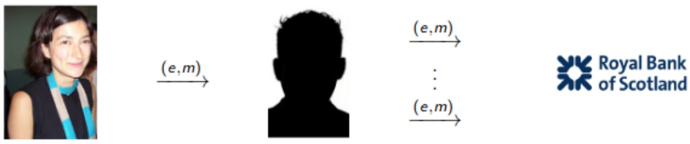
- Revocation
  - A certificate needs to be revoked if the corresponding private key has been compromised
  - Certificate Revocation Lists (CRLs) are the solution adopted in X.509
  - Online Certificate Status Protocol (OCSP) stapling is the modern solution to this problem

- **Lecture 13 - Secure communications: Cryptographic protocols**
  - Context
    - Applications **exchanging sensitive data** over a **public network**
      - eBanking
      - eCommerce
      - eVoting
      - eHealth
      - Blockchains
      - Mobile phones
      - ...

- A malicious agent can
    - record, alter, delete, insert, redirect, reorder, and reuse past or current messages, and inject new messages
        - $\longrightarrow$ **the network is the attacker**
    - Control dishonest participants
- More complex systems needed…

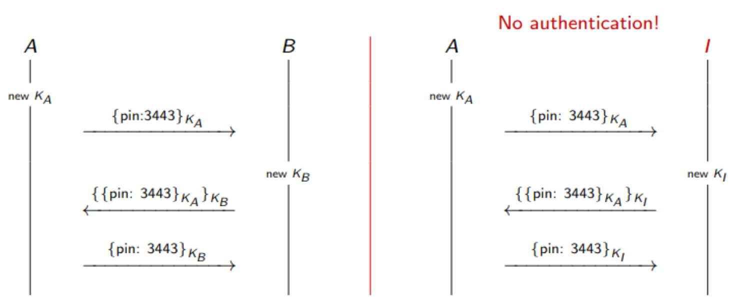


Replay attack



- Attacker sending the same message multiple times
- To achieve ore complex properties
    - **Confidentiality:**
        - Some information should never be revealed to unauthorised entities.
    - **Integrity:**
        - Data should not be altered in an unauthorised manner since the time it was created, transmitted or stored by an authorised source.
    - **Authentication:**
        - Ability to know with certainty the identity of an communicating entity.
    - **Anonymity:**
        - The identity of the author of an action (e.g. sending a message) should not be revealed.
    - **Unlinkability:**

        - An attacker should not be able to deduce whether different services are delivered to the same user
    - **Non-repudiation:**
        - The author of an action should not be able to deny having triggered this action.
- Cryptographic protocols
    - Distributed program relying on cryptographic primitives and whose goal is the establishment of "secure" communications
    - But!
        - Many exploitable errors are due not to design errors in the primitives, but to the way they are used, i.e. bad protocol design and buggy or not careful enough implementation
- Logical attacks
    - Many of these attacks do not even break the crypto primitives
    - Example
        - Assume a commutative symmetric encryption scheme
            - $\{\{m\}_{k_1}\}_{k_2} = \{\{m\}_{k_2}\}_{k_1}$
        - where $\{m\}_k$ denotes the encryption of message m under the key k
        - Ex: Stream Ciphers



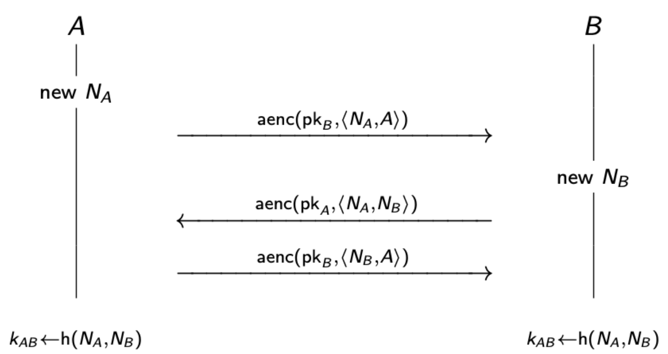
since $\{\{\text{pin: 3443}\}_{K_A}\}_{K_B} = \{\{\text{pin: 3443}\}_{K_B}\}_{K_A}$ by commutativity

- Authentication and key agreement protocols

- Long-term keys should be used as little as possible to reduce "attack-surface"
- The use of a key should be restricted to a specific purpose
    - e.g. you shouldn't use the same RSA key both for encryption and signing
- Public key algorithms tend to be computationally more expensive than symmetric key algorithms
- Long-term keys are used to establish short-time session keys
    - e.g. TLS over HTTP, AKA for 3G, BAC for epassports, etc.

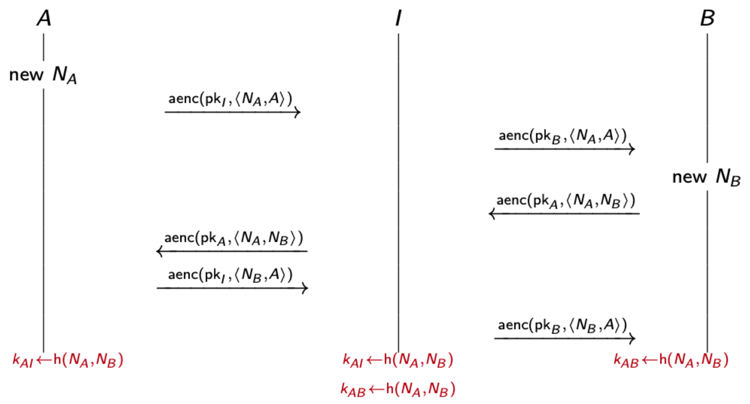
- Needham-Schroeder Public Key (NSPK)

NSPK: authentication and key agreement protocol

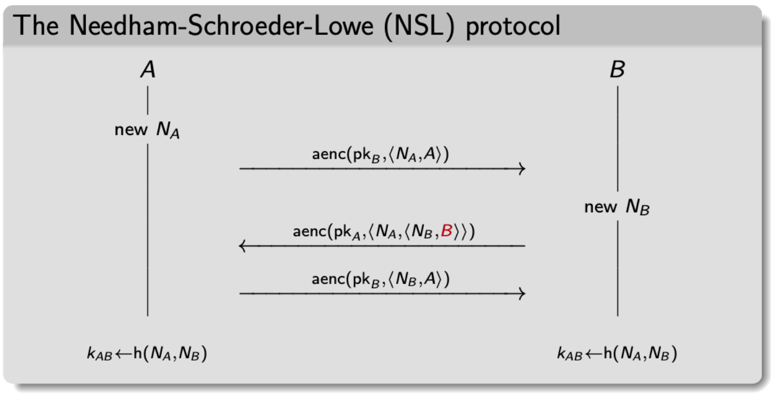


- security requirements
    - **Authentication**
        - if Alice has completed the protocol, apparently withBob, then Bob must also have completed the protocol with Alice.
        - If Bob has completed the protocol, apparently withAlice, then Alice must have completed the protocol with Bob.
    - **Confidentiality**
        - Messages sent encrypted with the agreed key $(k \leftarrow h(N_A, N_B))$ remain secret.
- Lowe's attack on authentication

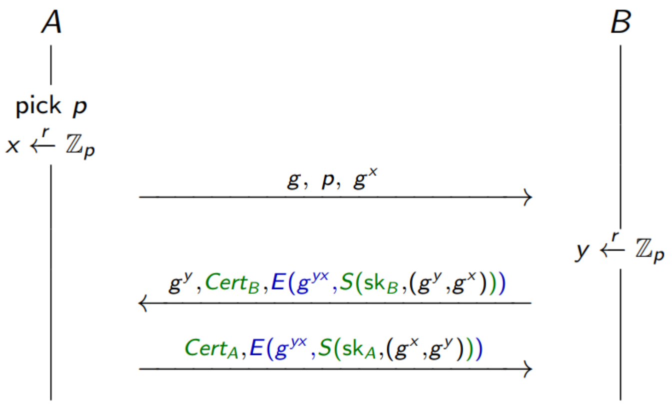
- Attack found 17 years after the publication of the NS protocol



- Lowe's fix



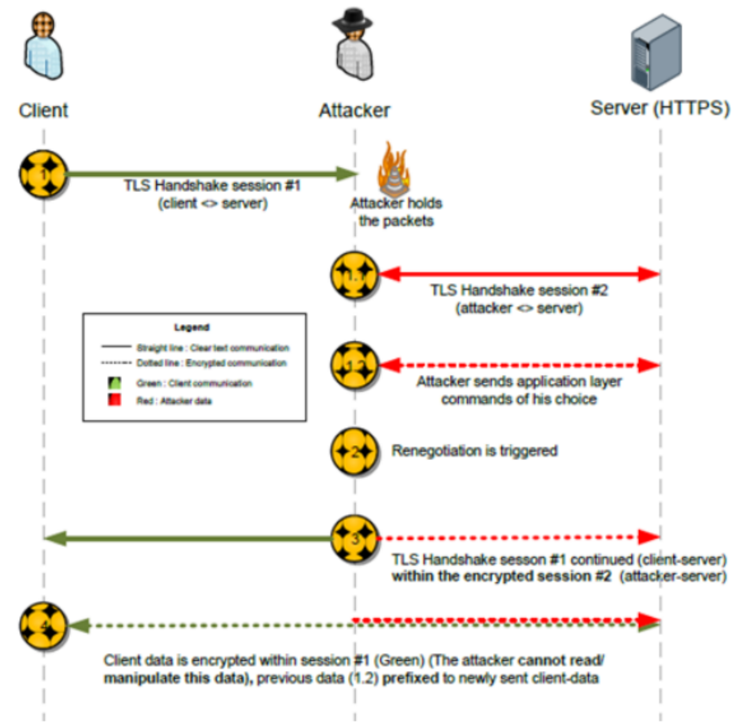
- Forward secrecy
  - The NSL protocol is secure against an attacker that controls the network
  - What if the Alice's and Bob's private keys get compromised?
  - What if the government forces Alice and Bob to reveal their private keys?
  - Can we still protect confidentiality?
  - A protocol ensures forward secrecy, if even if long-term keys are compromised, past sessions of the protocol are still kept confidential, and this even if an attacker actively interfered
- The Station-to-Station (StS) protocol



- p is a large prime
- g : generator of $\mathbb{Z}_p^*$
- The StS **ensures mutual authentication, key agreement, and forward secrecy**

- Old SSL/TLS handshake protocol

- SSL/TLS renegotiation weaknesses
  - Renegotiation has priority over application data
  - Renegotiation can take place in the middle of an application layer transaction



- Incorrect implicit assumption: the client doesn't change through renegotiation

- ## Marsh Ray's plaintext injection attack on HTTPS

Attacker:
```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:(no carriage return)
```

Victim:
```
GET /pizza?toppings=sausage;address=victim_str HTTP/1.1
Cookie:victim_cookie
```

Result:
```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:GET /pizza?toppings=sausage;address=victim_str HTTP/1.1
Cookie:victim_cookie
```

⇒ **Server uses victim's account to send a pizza to attacker!**

- ## Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to http://twitter.com/statuses/update.xml, as well as the user name and password

Attacker:
```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password
User-Agent: curl/7.19.5
Host: twitter.com
Accept:*/*
Content-Length: 140
Content-Type: application/x-www-form-urlencoded
status=
```

Victim:
```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password...
```

⇒ **the attacker gets the user name and password of the victim!**