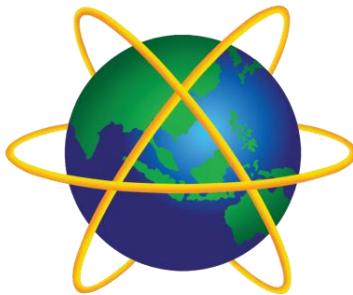


System and Network Administration



A • P • U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Public Key
Infrastructure

Secure Systems

1. Security policy
 - What needs to be protected
 - Kinds / level of protection
 - Responsibilities
 - Auditing policy
2. Security environment
 - Physical environment
 - Physical security
 - Hardware, operating system
 - firewalls, etc
3. Security mechanisms
 - **cryptography**
 - **authentication**
 - **security protocols**
4. Monitoring and auditing procedures
 - monitor access
 - audit trails
 - feedback on failures, security breaches
 - containment & recovery

Security Protocols

In practice, no single **mechanism** is adequate to address all goals, so a mix of mechanisms will be required to enforce **security policies**.

A **protocol** is an orderly sequence of steps that two or more parties follow in order to accomplish some joint task

e.g. protocols for

- authentication of participants in an exchange of messages
- data integrity checks

Encryption is a **mechanism** that can be incorporated into security **protocols**

Mechanisms to Protocols: Digital Signatures



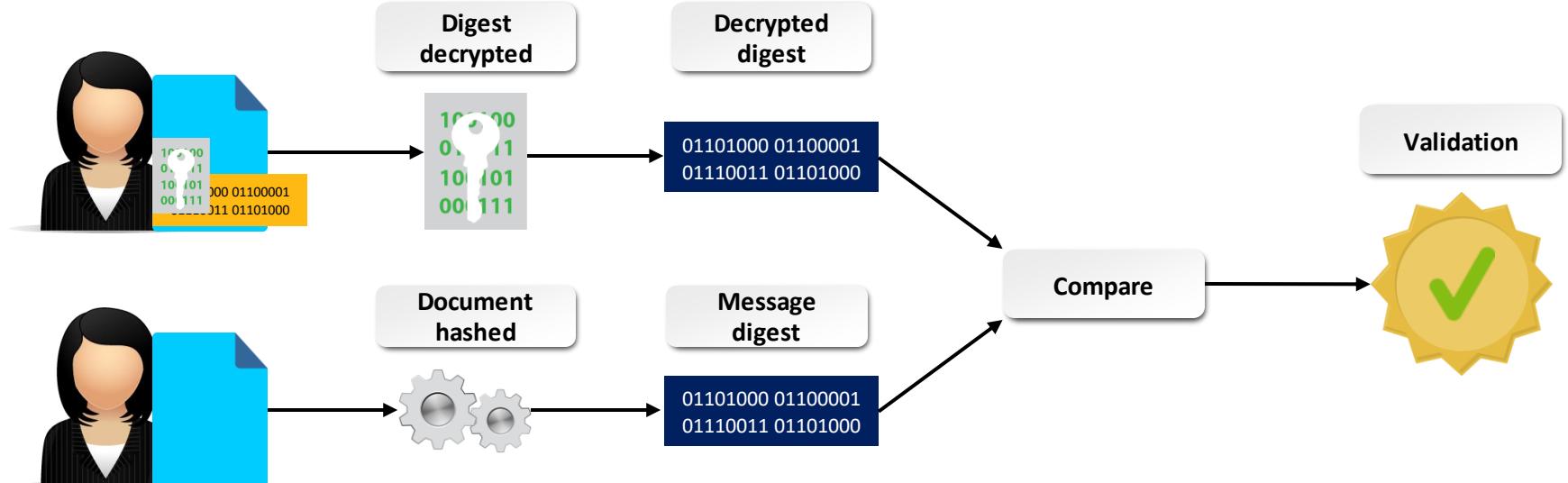
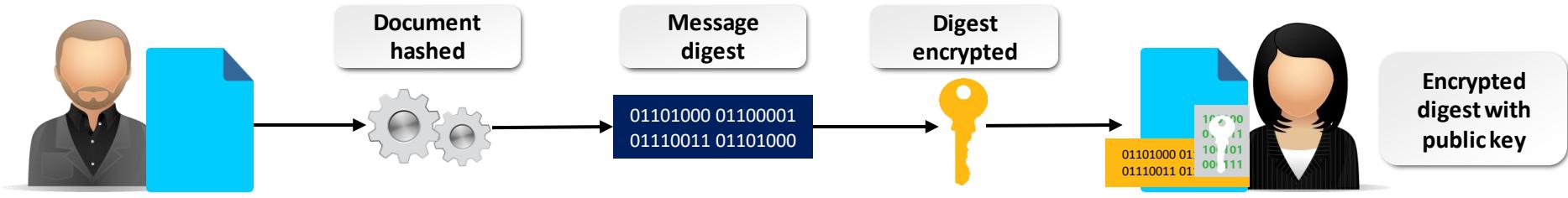
Use two algorithms: a “one-way” algorithm and a “public key” algorithm

1. Create a message
2. Generate a message digest
3. Encrypt the message digest with private key
4. Send both

Receiver runs the plaintext message through the same hash algorithm and compares the outcome with the message digest that was received

- one-way (hash) assures integrity
- public key assures non-reputability

Digital Signatures (Cont.)



Digital Signatures

The digest that is sent with the message serves as a trusted, stored copy, like the password database

The message itself does not need to be encrypted, although it can be.

The encrypted message digest is the digital signature

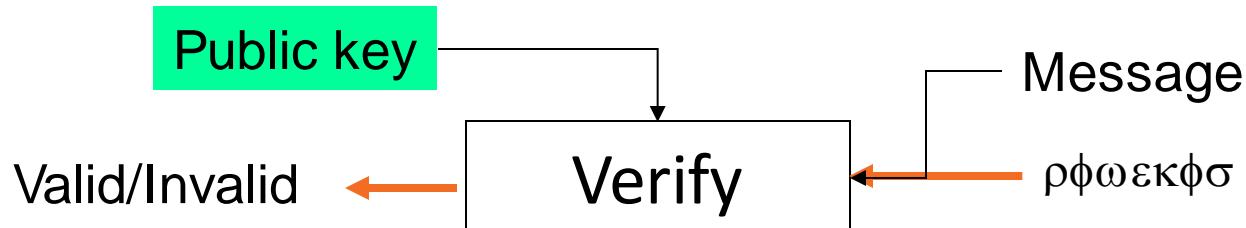
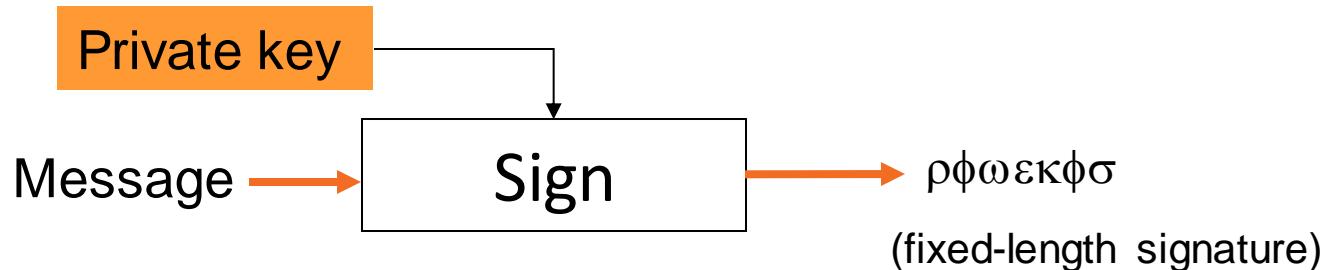
-----BEGIN PGP MESSAGE-----Version:

```
2.6.3ihIwDOds8UFDZJ60BA/9IkThuRPJUtYCoaYFdNusnSGg/GR91uIGH6GI5azCgk6qHVe4
CRODsc0saK1e5ZHW0sTSFpIOsdoFE4Nms7a70xL/WjYBN4ZBedRgue7ox0AC3D18z6baJ+xeS
oGxbn4o0fdh+DoYjo38t1tsbBmMVK4LLhLxscasVUskQDDOoA6YAAAEibz1z3BMGTRtH1kjdb
5pUTOpFvbdLXvvDFlo00QAmW6Leuwb1Q4+Ny5MSI1j3mxkDCjfC2qCp/KUwxhDTqw8hgoFx6
O4Yynmu+OQyomA5A1oo9WEJ1jtURbkGM38dD2AfqmnmkwcAp2SF0IfabaIv7AcfEwWEKQv1Th
pRIxi5uJjgxhXXFyLpmC4SUceDmhYgNESVsZAY2Y+zvtMTv1QP++1HOZP3gw9HdVKAZQX4xES
Xq5bKPD6Wr2IZ2c4+j/psS4ytTzJMa4msL9z5L2vZF1GmsEvt8bxHWVgH1rL14L8mmirCU3jZ
FjWGZY30RYjEjghgQNNgTREqRfwQw6SQwmdN60cEYOFjr8fa1PGQFL42sjbZ1v7+uKbx33PLt
EeR1Q==a+bB
```

-----END PGP MESSAGE-----

Digital signatures

- Only the signer (who has a private key) can generate a valid signature.

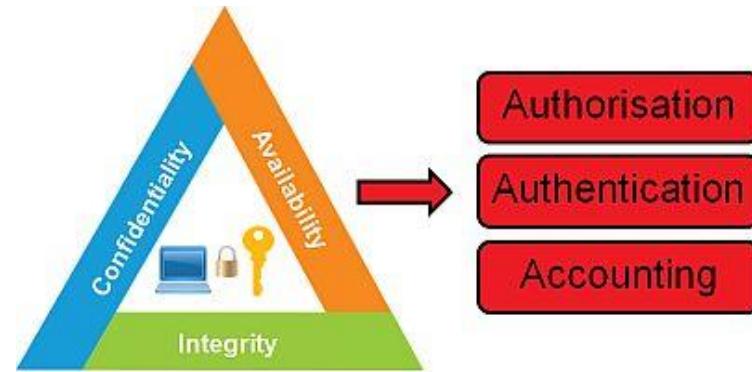


- With the corresponding public key, anyone can verify if a signature is valid with respect to a message.

Digital Signatures

A message digest that has been encrypted again with a user's private key.

- Digital signatures uphold:
 - Authenticity
 - Integrity
 - Non-repudiation



Non-repudiation

Ensuring that the party that sent a transmission or created data remains associated with the data and cannot deny sending or creating the data.





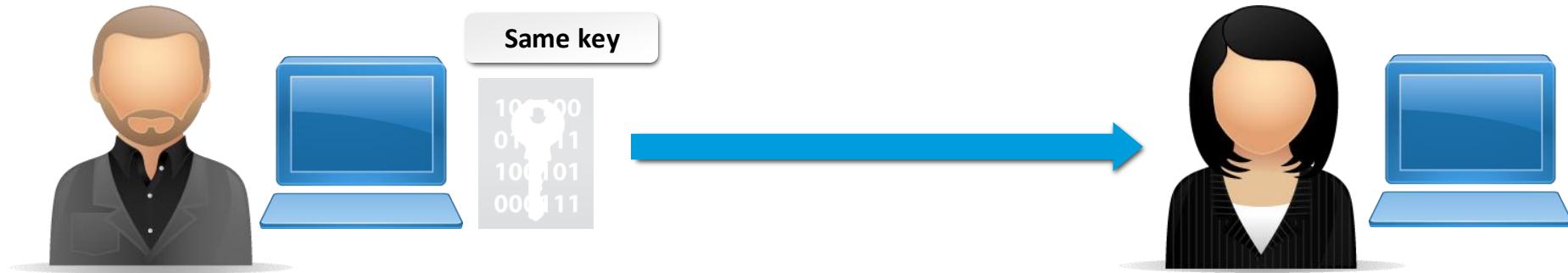
Key Exchange

The method by which cryptographic keys are transferred between entities.

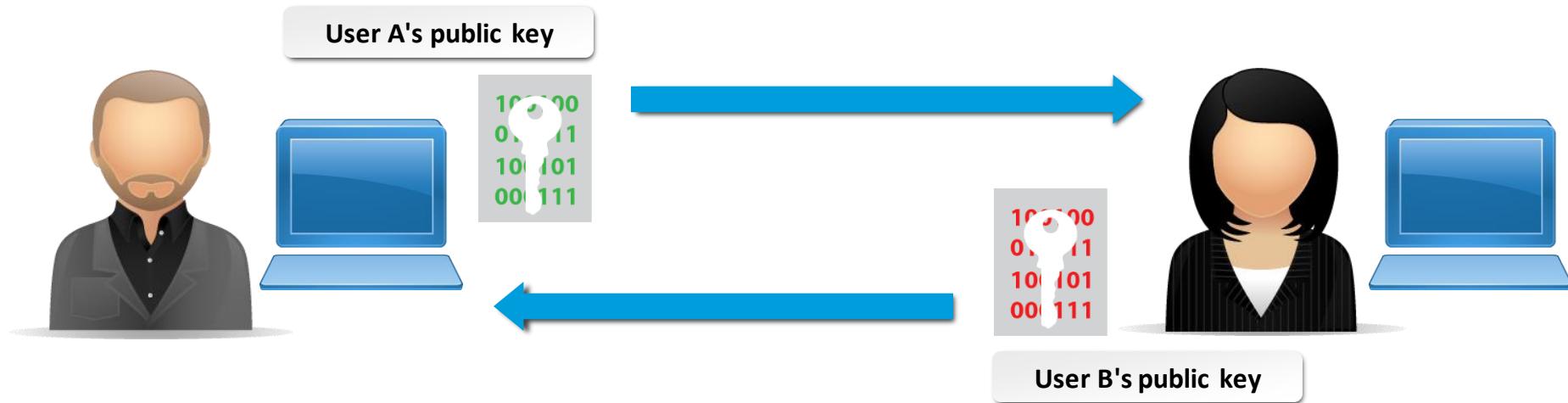
- Both sender and receiver need to be equipped to encrypt and decrypt messages.
- How they are equipped depends on the technique used:
 - In symmetric cryptography, both need a copy of the same key.
 - In asymmetric cryptography, anyone needing to encrypt a message needs the recipient's public key.
- In-band exchange uses the same path as the data being shared.
- Out-of-band exchange uses a different path, like a phone call or physical meeting.
 - Symmetric key cryptography requires out-of-band.

Key Exchange (Cont.)

Symmetric Cipher



Asymmetric Cipher

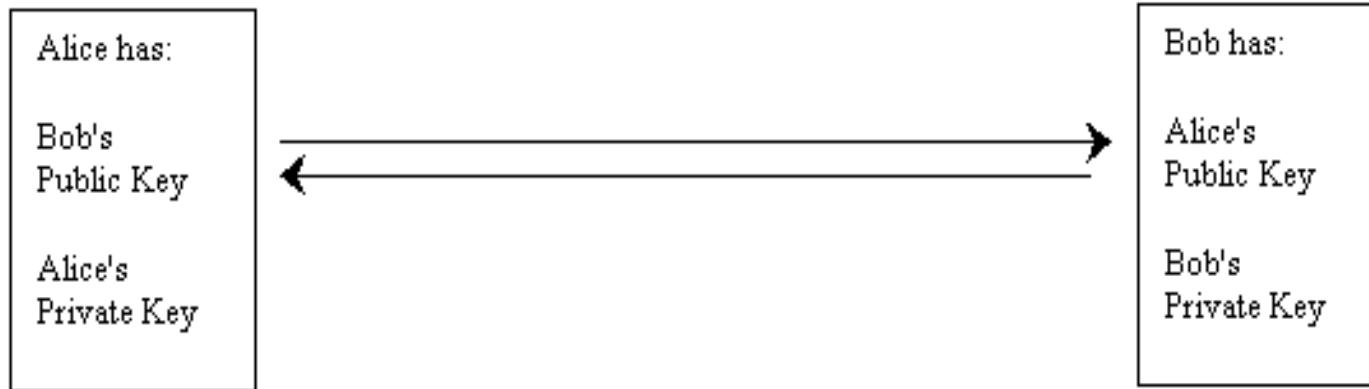


Public key cryptography
as described so far still has a missing link

- How do you know that the person on the other end of the communication channel is who they say they are?
- ***They can present their public key, but that proves nothing about their identity.***

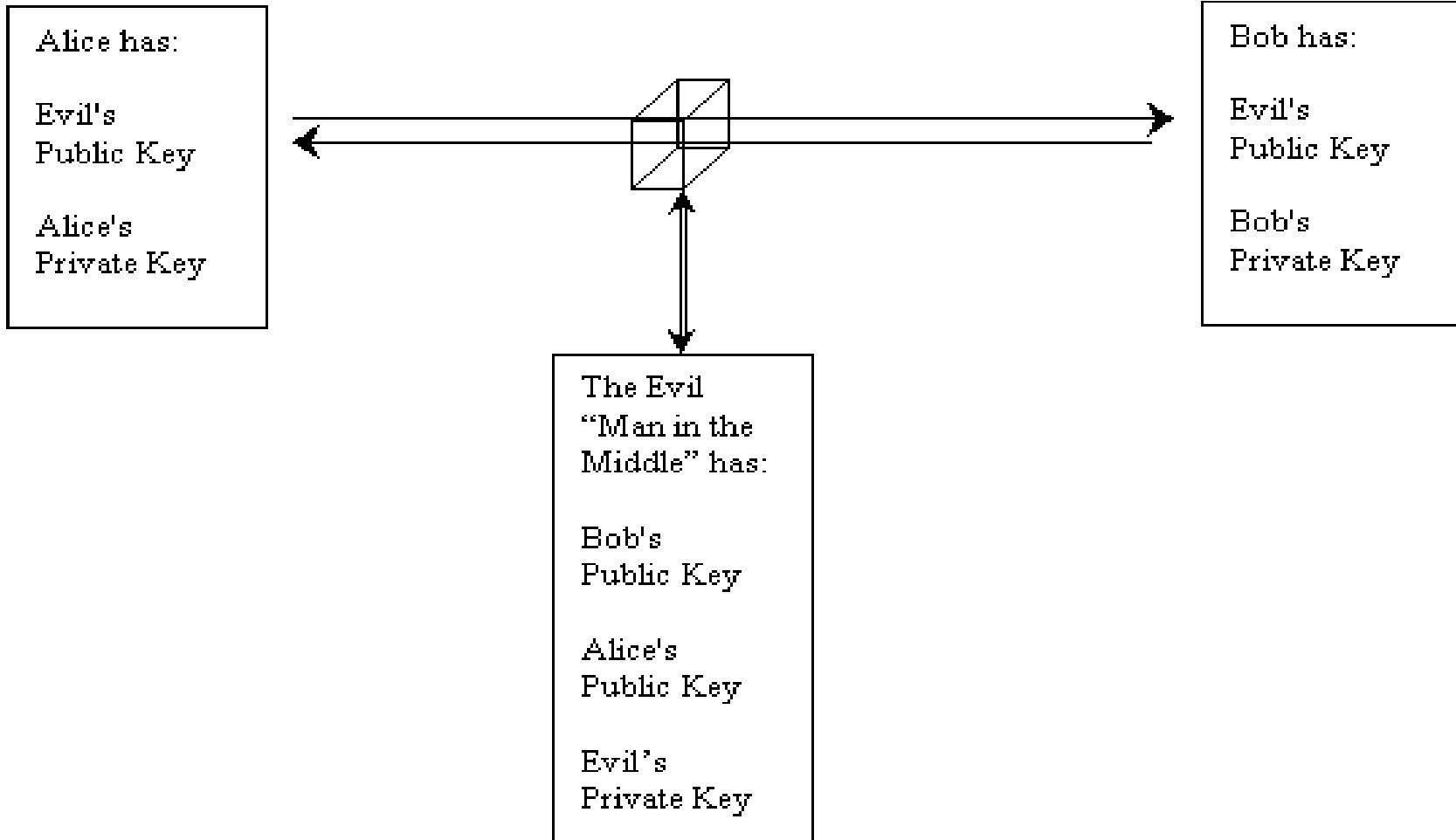
Security Protocols

Example: Mutual authentication and session key agreement via public keys



Alice and Bob exchange public keys; Alice selects a session key, encrypts it with Bob's public key, and sends it to Bob. It is a private message, because it can only be decrypted using Bob's private key. Bob confirms the session key by decrypting it, and sending it back to Alice after re-encrypting it with Alice's public key. After this, they exchange messages encrypted with the (symmetric) session key.

Both Alice and Bob can be confident that the session key is a secret shared only by them, *provided that they have the correct public keys*- otherwise this method is vulnerable to attack by a third party impersonating Alice to Bob and Bob to Alice.

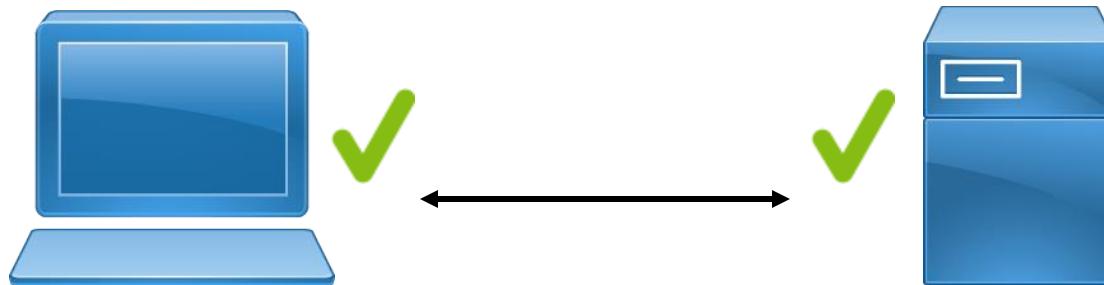


An evil third party intercepts all communications between Alice and Bob. They believe the system is working correctly, because “the man in the middle” can send private messages to both parties. This attack works when all public keys look the same, and Alice and Bob (naively) trust each other and the system. If Alice and Bob used digital certificates, Evil could not substitute his public key for theirs.

Mutual Authentication

A security mechanism that requires that each party in a communication verifies the identity of every other party in the communication.

- The server verifies the client's credentials, while the client verifies the credentials of the server
- Prevents clients from sending confidential information to insecure servers.
- Helps to avoid man-in-the-middle attacks.



Security Protocols

The Man-in-the-Middle attack works when all public keys look the same, and people exchange keys directly.

- **Digital Certificates** are used to assure *authenticity* of the sender.
- Issued by third parties: *certificate authorities* (CA).
- Individuals and companies apply by sending CA their public key and identifying info.
- CA verifies this info and creates a certificate containing public key and identifying info and encrypts this using its private key.
- Usually have an expiration date and a confidence limit (“valid for transactions \leq this amount”)

Public Key Infrastructure (PKI)

- In PKI a Certification Authority (CA) validates keys.
 - Checks real-world credentials,
 - Gets Public key from user in person,
 - Signs Certificate (“cert”) validating the Public key.
- If the end point trusts the CA, then it will trust that entity and who it claims to be.
- **The CA solves the problem of authentication by trusted referral.**





CA Hierarchies

CA hierarchies: A single CA or group of CAs that work together to issue digital certificates.

Chain of trust: The process by which an entity validates a certificate from the bottom of the hierarchy to the top.

- Certificate may be signed by another CA, which is signed by another CA above it, etc.
- Entity must trust all links (CAs) in the chain.

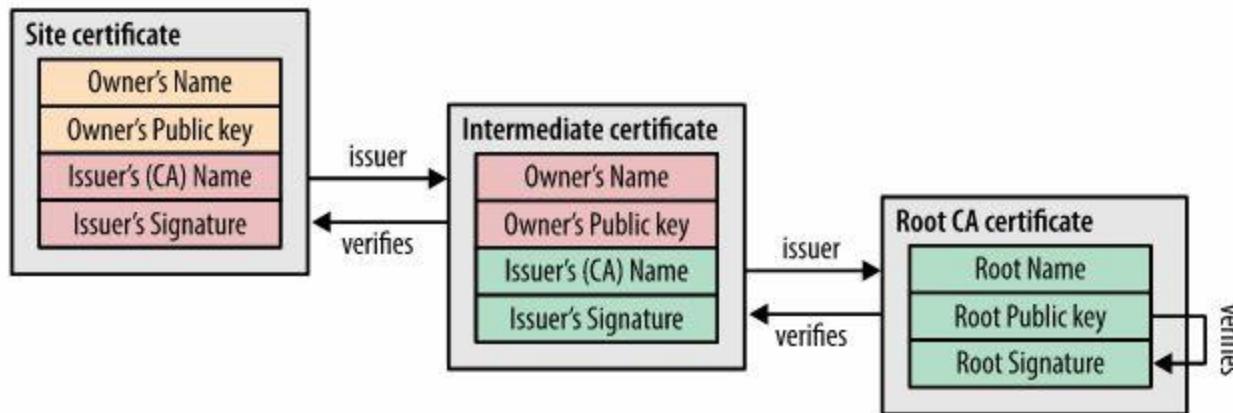


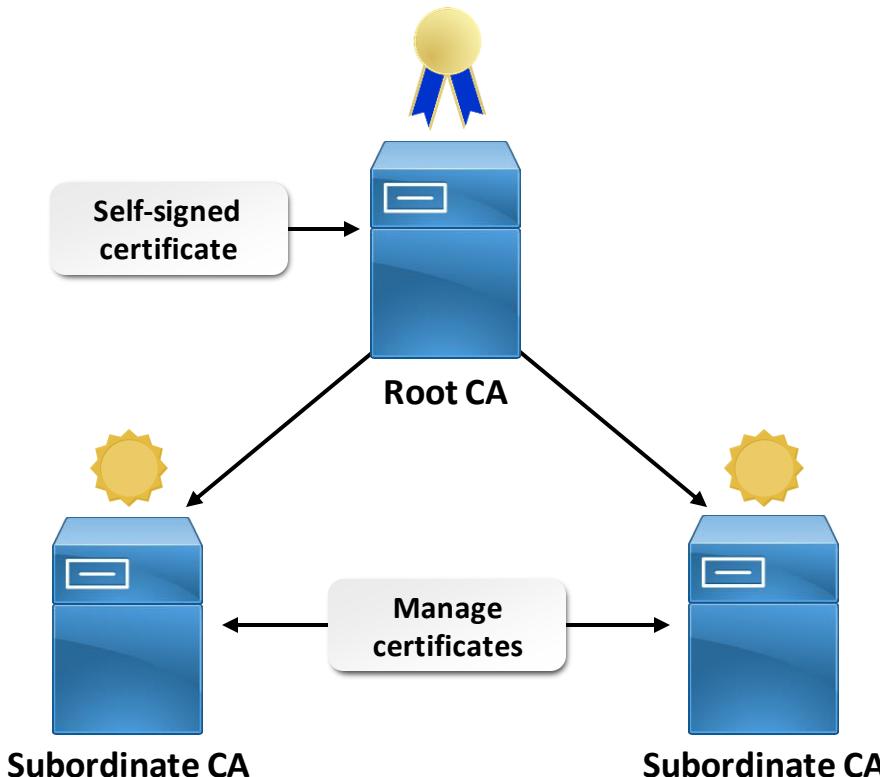
Figure 4-5. CA signing of digital certificates

<https://hpbn.co/transport-layer-security-tls/>

The Root CA

The topmost CA in the hierarchy and the most trusted authority.

- **Self-signs** first certificate in the chain.
- Must be secure; if compromised, all other certificates are invalid.

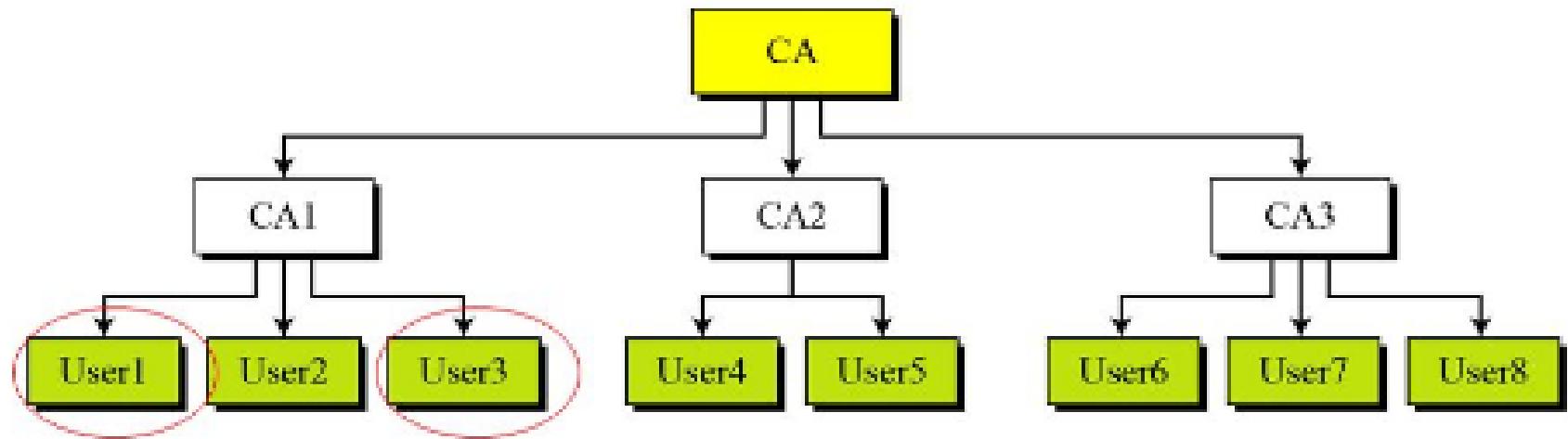


Subordinate CAs

- Issue, renew, suspend, and revoke certificates as needed.

PKI: CA Hierarchy

- The root CA has a self-signed, self-issued certificate;
- it needs to be trusted by other CAs and users for the system to work.



The CA (root) has signed certificates for CA1, CA2, and CA3.
 CA1 has signed certificates for User1, User2 and User3;
and so on.

Types of Certificates

Certificate Type	Description
Self-signed	<ul style="list-style-type: none"> • Certificate owned by same entity that signs it. • Does not recognize higher authority in the chain. • Requires client to trust entity directly.
Root	<ul style="list-style-type: none"> • Issued by CA. • Certifies all other certificates below it. • Must be self-signed
User / Computer	<ul style="list-style-type: none"> • Use certificates for authentication rather than passwords. • Example: certificate stored on smartcard. • Example: browser certificate for https:// authentication.
Email	<ul style="list-style-type: none"> • Used to encrypt and authenticate email messages.
Code signing	<ul style="list-style-type: none"> • Developers digitally sign source code so customers can validate apps. • Code is often self-signed, but can be signed by CA-issued certificates.

- Every browser ships with a list of well-known certificate authorities
- Every browser allows you to inspect the chain of trust of your secure connection
 - Usually by clicking on the lock icon beside the URL
- Every operating system and most browsers provide a way for you to manually import any certificate you trust.
 - How you obtain the certificate and verify its integrity is completely up to you.

PKI in Practice

- The browser specifies which CAs to trust (root CAs), and the burden is then on the CAs to verify each site they sign, and to audit and verify that these certificates are not misused or compromised.
- If the security of any site with the CA's certificate is breached, it is the responsibility of that CA to revoke the compromised certificate.

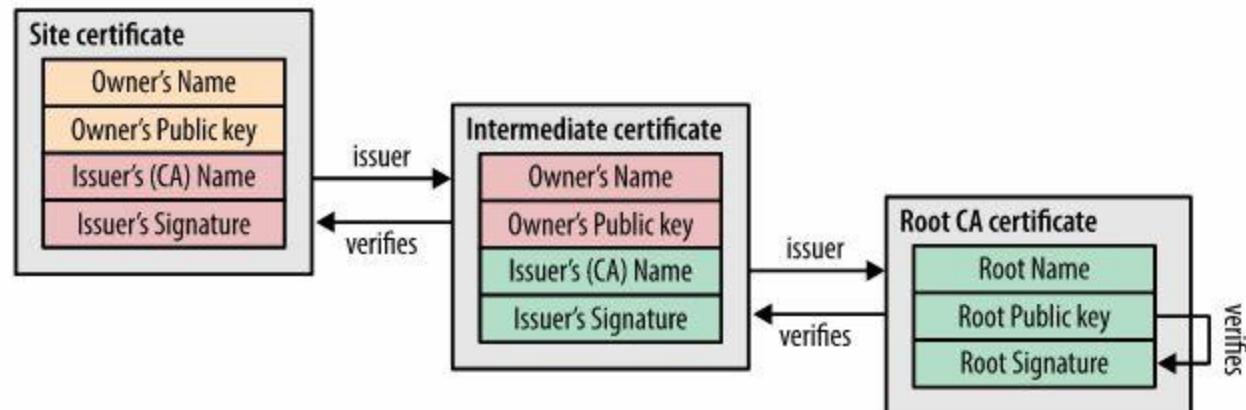


Figure 4-5. CA signing of digital certificates

<https://hpbn.co/transport-layer-security-tls/>

X.509

A standard for formatting digital certificates used in a PKI

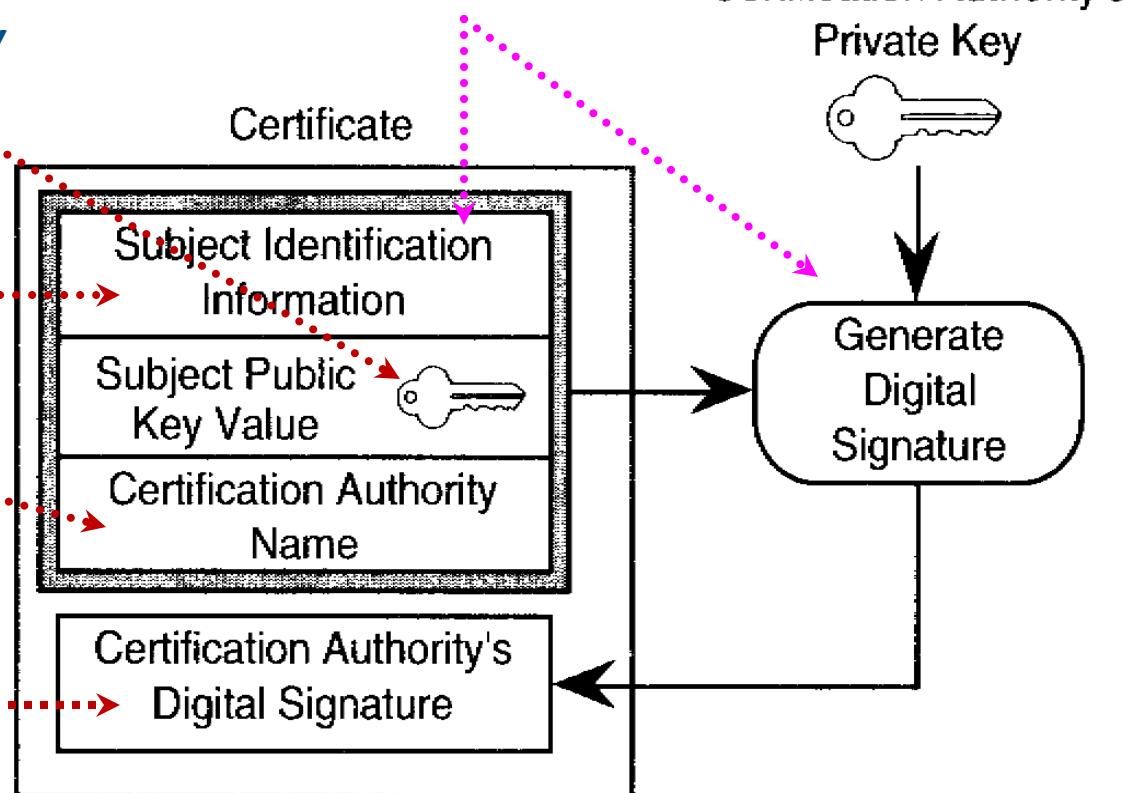
- Defines the structure of a certificate with information provided in CSR.
- Includes:
 - The public key.
 - The X.509 version.
 - The certificate's serial number.
 - The algorithm used to sign the certificate.
 - The name of the issuing entity.
 - The validity period.
 - The name of the subject being certified.
 - Optional organizational/regional attributes.



Digital Certificate

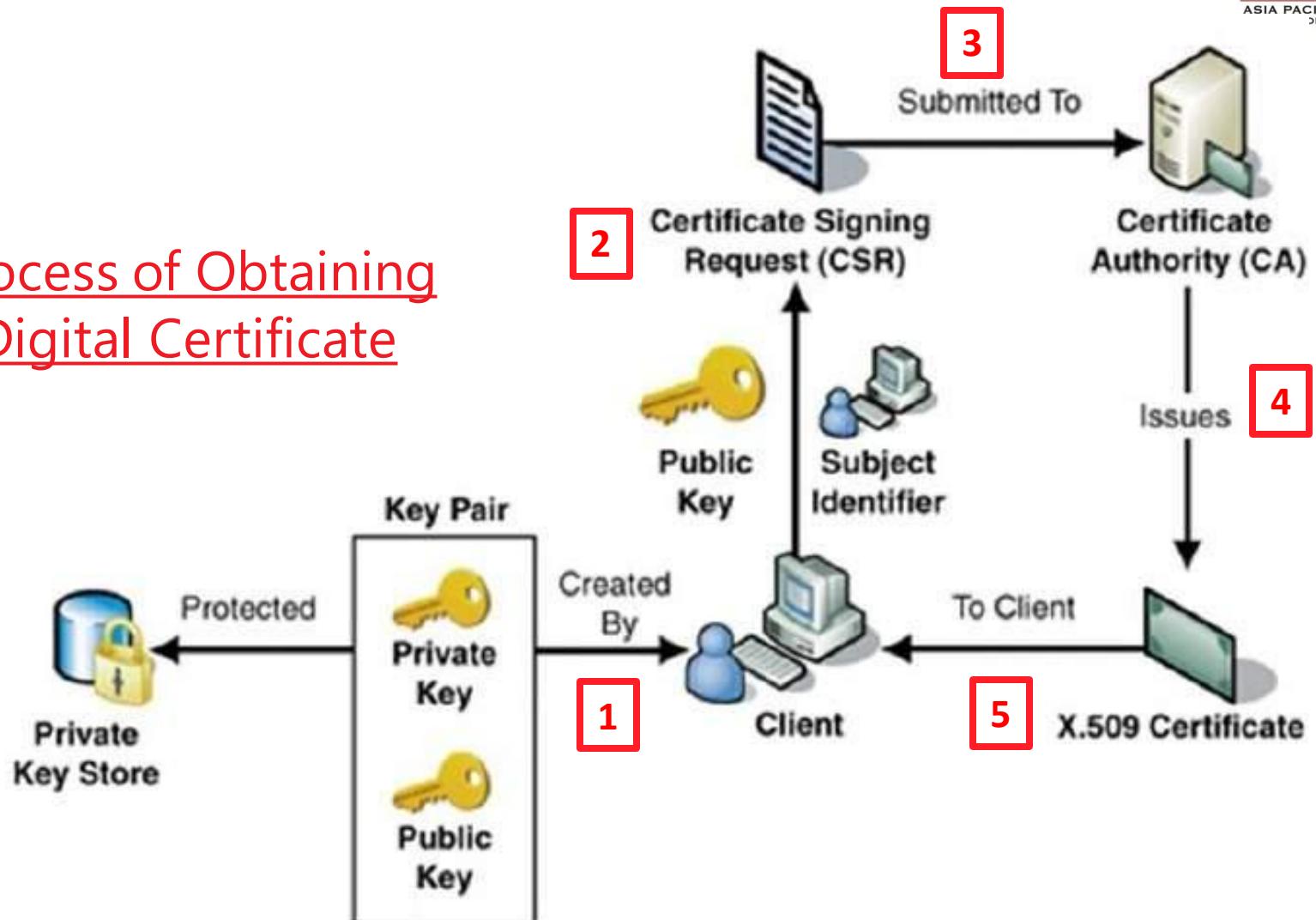
4. DECRYPT SENDER'S MESSAGE USING SENDER'S PUBLIC KEY
3. VERIFY IDENTITY OF SENDER.
2. VERIFY THAT CA NAME MATCHES.
1. USE CA'S PUBLIC KEY TO VERIFY CERTIFICATE IS GENUINE AND HAS NOT BEEN ALTERED.

0. CA AUTHENTICATES SENDER, DIGITALLY SIGNS CERTIFICATE



From another certificate!

Process of Obtaining Digital Certificate



Self-signed certificate: Encryption vs. Trust



- **xca** from <http://xca.hohnstaedt.de/xca/index.php/download> has a nice YouTube tutorial to get you started with creating self-signed certificates.
- While any self-signed certificate is fine for **encryption**, we need a different certificate to make a **trusted connection** with browsers - there is a good discussion of trust vs. security at

<https://security.stackexchange.com/questions/112768/why-are-self-signed-certificates-not-trusted-and-is-there-a-way-to-make-them-tru>

- Note that you must use **https://** in the URL to get a browser to use SSL for communication with a web server.
- Since the release of Google Chrome 58 (2017-04-19) browsers have been tightening their criteria for certificates, to the point where there is (and may not ever be) any way to trust self-signed certificates in Microsoft Edge ([see this post from Microsoft - 2017-05-09](#))
- Also, "**Subject Alternate Name**" is now a required field for Chrome to trust a certificate.



