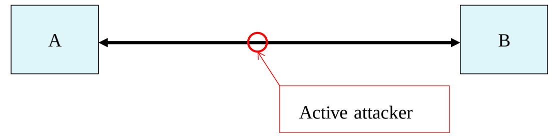
Cryptography

- Introduction to the basic concepts
- Define and see examples of
  - Stream ciphers
  - Block ciphers
  - Public key encryption
  - Hash functions
  - Message authentication codes
  - Digital signatures
  - Digital certificates

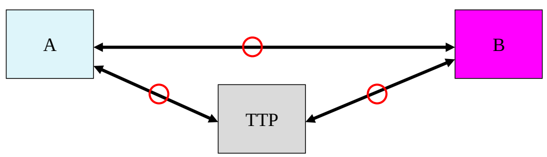
The old paradigm

- Insecure communication links



- A and B trust each other
  - Together they try to avoid attacks from outsiders
- Cryptography can give them
  - data confidentiality
  - data integrity
  - data origin authentication

New paradigm



- The insiders have no reason to trust each other
- *Trusted Third Party* TTP
- *Nonrepudiation* services generate evidence for resolving a dispute

Cryptographic keys

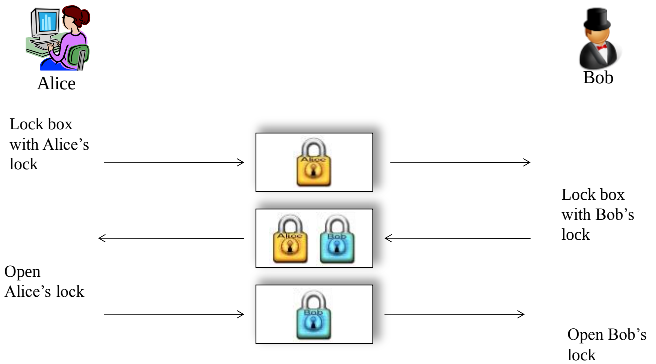
- Cryptographic algorithms use keys to protect data

Key management is the topic of addressing

- Where are keys generated?
- How are keys generated?
- Where are keys stored?
- How do they get there?
- Where are keys used
- How are they revoked and replaced?

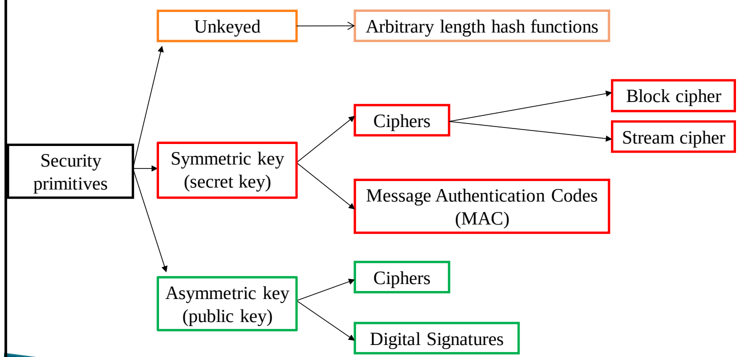
Sending a secure message

- Alice wants to send a box with a message to Bob



Cryptographic primitives

Primitives that we will look at

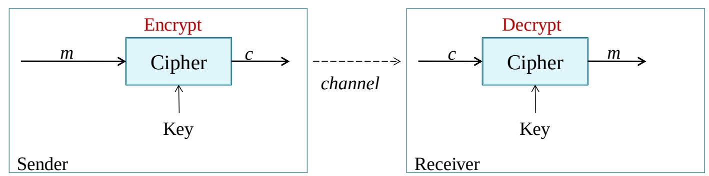


Strength of encryption mechanisms

- Empirically secure – Secure based on the fact that no one has broken it for some time.
  - Most common for practically used symmetric primitives
  - Typically very efficient
- Provably secure – We prove that breaking a scheme is at least as hard as breaking some well known problem like factoring or discrete log.
  - Most common for asymmetric primitives
  - Also possible for symmetric primitives
- Unconditionally secure – The schemes are secure even if the adversary has unlimited computing power
  - Not common but possible

Plaintext and ciphertext

- The plaintext is the message we want to send
  - We denote it by *m*
- The ciphertext is the data that we actually send
  - We denote it by *c*



Simplified model (without source coding, channel coding, modulation etc.)

Attack Scenarios

- Kerckhoffs' principle:
  - Only the key should be unknown to an adversary
    - Security should not be based on the fact that the algorithm is secret, WHY?
  - Formulated in the 19th century and is for different reasons still sometimes ignored in the 21th century
- A scheme can be analysed under different scenarios
  - Ciphertext only attack
  - Known plaintext attack
  - Chosen plaintext attack
  - Chosen ciphertext attack
- All scenarios implicitly assume Kerckhoffs' principle
- **Primary attack goal:** Find the secret key
  - However, other goals can be imagined as well

Symmetric Key Cryptography

Some old cryptographic tools

Scytale

Jefferson's disk

Enigma

Very simple symmetric schemes (motivate stream ciphers)

We will assume that all keys are chosen from a uniform distribution!

Shift cipher (Caesar cipher)

Plaintext	A	B	C	D	E	F	...	X	Y	Z
Ciphertext	D	E	F	G	H	I	...	A	B	C

Map letter to number, then

Plaintext	0	1	2	3	4	5	...	23	24	25
Ciphertext	3	4	5	6	7	8	...	0	1	2

$$c_t = m_t + 3$$
$$m_t = c_t - 3$$

Key is "3" (or "D")

Problems:
- ✗ Only 26 keys
- ✗ Redundancy in language is preserved

Substitution cipher

Define a permutation over the alphabet:

Plaintext	A	B	C	D	E	F	...	X	Y	Z
Ciphertext	S	H	D	T	V	B	...	Q	A	O

Table is the key

Problems:
- ✓ Only 26 keys (There are now 26!)
- ✗ Redundancy in language is preserved

Vigenère cipher

Use a shift cipher, but different shifts for n consecutive letters

0					
A	B	C	...	Y	Z
F	G	H	...	D	E

1					
A	B	C	...	Y	Z
T	U	V	...	R	S

....

n - 1					
A	B	C	...	Y	Z
M	N	O	...	K	L

Letter *t* in message of length *N* is encrypted with table *t (mod n)*

Key is sequence of *n* numbers (or letters)

Problems:
- ✓ Only 26 keys (There are now 26^n)
- ✓/✗ Redundancy in language is preserved (n distributions)

The One-Time-Pad (OTP)

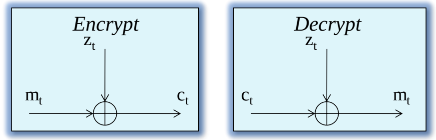
- Substitution cipher and Vigenere cipher can be broken with statistics since the language has redundancy!
  - Note that we are talking about a ciphertext only attack
- But what if n=N in Vigenere cipher? (Length of key is the same as message length)
- Then it is UNBREAKABLE!
- This is called Vernam cipher or One-Time-Pad (OTP)
- Perfect Secrecy (unconditionally secure)

Problems:
- ✓ Only 26 keys (There are now 26^N)
- ✓ Redundancy in language is preserved (No redundancy at all)

- Secure since number of possible keys is same as number of possible messages. New problem!

Stream Ciphers

- **A good idea:** Take a short random key and expand it to a long (pseudo)random sequence of bits
- That is a stream cipher!



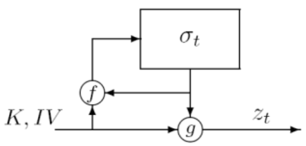
Binary additive stream cipher

xor function

a	b	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

$$m_t \oplus \underbrace{z_t \oplus z_t}_{=0} = m_t$$

Inside the keystream generator



$\sigma_0 = \gamma(K, IV)$ Initialisation function

$\sigma_{t+1} = f(\sigma_t, K, IV)$ State update function

$z_t = g(\sigma_t, K, IV)$ Output function

- IV (Initialisation Vector) allows reuse of key
- State can be: shift register, large table, counter etc
- Well known stream ciphers: RC4, SNOW, A5/1, E0

Block ciphers

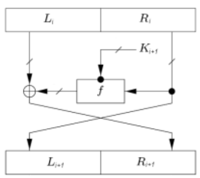
- Return to substitution cipher

Plaintext	A	B	C	D	E	F	...	X	Y	Z
Ciphertext	S	H	D	T	V	B	...	Q	A	O

- Substitution cipher is a block cipher
  - Still, redundancy is a problem
  - Block length too small → complete table easily recovered if some plaintext is known
- Increase block size to e.g., 64, 128, 192 or 256 bits
  - Now table is too large to fit in memory
- Solution: Use mathematic tools to map plaintext symbols to ciphertext symbols (and back)!
  - Still preserved redundancy, but we will solve that soon...

Two variants of block cipher design ideas

Feistel structure

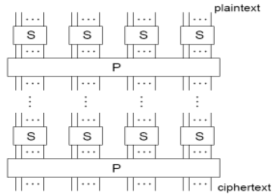


Encrypt $\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus f(K_{i+1}, R_i) \end{cases}$

Decrypt $\begin{cases} R_i = L_{i+1} \\ L_i = R_{i+1} \oplus f(K_{i+1}, L_{i+1}) \end{cases}$

Decryption can be done using the same structure, but with keys in reverse order

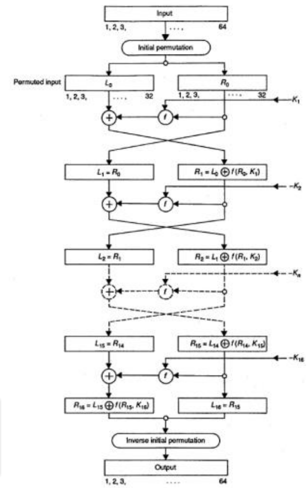
Substitution Permutation Network (SP-network)



- Repeated substitutions and permutations
- Confusion and diffusion
- Go backwards to decrypt

Feistel cipher: DES

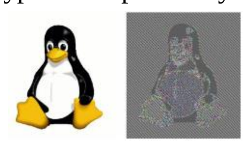
- Block size: 64 bits
- 16 rounds
- Key size: 56 bits
- Can be ”broken” in a day or so
- Standard 1977 – 1998
- 1998 – 2002: 3DES



AES has been standard since 2002 and is an example of a SP-network

Modes of operation – ECB

- Electronic code book mode (ECB)
  - $c_i = eK(m_i)$
  - $m_i = dK(c_i)$
- All blocks encrypted independently of each other



- Redundancy preserved!

Modes of operation – CBC

- Cipher Block Chaining (CBC)
  - $c_i = \mathrm{eK}(m_i \oplus c_{i-1})$
  - $m_i = \mathrm{dK}(c_i) \oplus c_{i-1}$
- Redundancy removed

Original

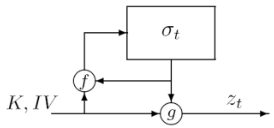
Encrypted with ECB mode

Encrypted with CBC mode

Modes of operation – OFB

- Output feedback mode
  - Turns the block cipher into a stream cipher
  - $z_t = \mathrm{eK}(z_{t-1}),\quad z_{-1} = IV$
  - $c_t = m_t \oplus z_t$
  - $m_t = c_t \oplus z_t$

Model of stream cipher



Advanced state update function *f*, but very simple keystream generation function, *g*. (*Counter mode* has the opposite property, See home exercise 1)

Public key cryptography

- Also called asymmetric cryptography
- Encryption
  - Public key used to encrypt
  - Private key used to decrypt
- Digital Signatures
  - Public key used for verification
  - Private key used for signing
- Note the terminology!
  - *Secret key* used in symmetric algorithms
  - *Public key* and *private key* used in asymmetric algorithms
    - Private key is sometimes also called secret key

Some mathematics before we move on

- Modular arithmetic:
- $a \equiv b \bmod n$ if and only if $a - b = k \cdot n$ for some integer k
- Properties:

$(a \bmod n) + (b \bmod n) \equiv (a + b \bmod n)$

$(a \bmod n) \cdot (b \bmod n) \equiv (a \cdot b \bmod n)$

for every $a \neq 0 \bmod p$, p prime, there exists an integer a^{-1} so that $a \cdot a^{-1} \equiv 1 \bmod p$

- gcd(a,b) is the greatest common divisor of a and b
- More generally:

If and only if $\gcd(a, n) = 1$, then ther e exists an integer a^{-1} so that $a \cdot a^{-1} \equiv 1 \bmod n$

Examples

a) $32 \equiv 6 \bmod 13 \text{ since } 32 - 6 = 2 \cdot 13$

b) $60 \bmod 13 \equiv (20 \bmod 13) + (40 \bmod 13) \equiv 7 + 1 \bmod 13 \equiv 8 \bmod 13$

c) $2^{10} \bmod 13 \equiv (2^5 \bmod 13) \cdot (2^5 \bmod 13) \equiv 6 \cdot 6 \bmod 13 \equiv 10 \bmod 13$

d) $8^{-1} \bmod 13 \equiv 5 \bmod 13 \text{ since } 8 \cdot 5 \equiv 1 \bmod 13$

e) $8^{-1} \bmod 12 \text{ does not exist since } \gcd(8,12) = 4 \neq 1$

More mathematics

- Euler phi function: $\phi(n)$ is the number of integers < n that are coprime to n

$$\phi(p^k) = p^k - p^{k-1},\ p \text{ prime}$$
$$\phi(\text{nm}) = \phi(\text{n})\phi(\text{m}) \text{ if } m \text{ and } n \text{ are coprime}$$

- Euler's Theorem: $a^{\phi(n)} \equiv 1 \bmod n$ is valid for all a when gcd(a,n)=1

More examples

a) $\phi(13) = 12$

b) $\phi(17) = 16$

c) $\phi(221) = \phi(13 \cdot 17) = \phi(13) \cdot \phi(17) = 12 \cdot 16 = 192$

d) $\phi(12) = \phi(4) \cdot \phi(3) = (2^2 - 2)(3-1) = 4$

e) $a^{12} \equiv 1 \bmod 13$ for all a that are not multiples of 13

f) $a^{192} \equiv 1 \bmod 221$ for all a such that gcd$(a,221)=1$

More mathematics

- Let p be a prime and a an arbitrary (nonzero) integer. The *multiplicative order of a modulo p* is defined to be the smallest integer n such that $a^n = 1 \bmod p$.
- Fermat's little theorem: For $a \neq 0 \bmod p$ and p prime

$$a^{p-1} \equiv 1 \bmod p$$

- The order of an element divides $p - 1$

Public key cryptography

- Usually based on one of two mathematical problems
  - Factoring – Given an integer n, find the prime factors
  - Discrete Logarithm Problem (DLP) – Given a prime p and integers a and y, find x such that $y = a^x \bmod p$
- Other mathematical problems can be used

RSA encryption, parameters

- Based on the problem of factoring
- Pick primes p, q. Let $n=p\cdot q$ and compute

$$\phi(n)=(p-1)(q-1)$$

- Pick an integer e such that

$$\gcd(e,\phi(n))=1$$

- Find d such that

$$e\cdot d\equiv 1 \bmod \phi(n)$$

- Public key: e, n
- Private key: $d, \phi(n), p, q$

RSA encryption

- Encrypt:
  - $c=m^e \bmod n$
- Decrypt:
  - $m=c^d \bmod n$
- Proof that it works:

$$c^d=m^{ed}=m^{k\phi(n)+1}=\left(m^{\phi(n)}\right)^k m=1^k m=m \bmod n$$

Note that only d and n is needed in decryption. However, in practice p and q are used to speed up decryption using the chinese remainder theorem. (Not included in course)

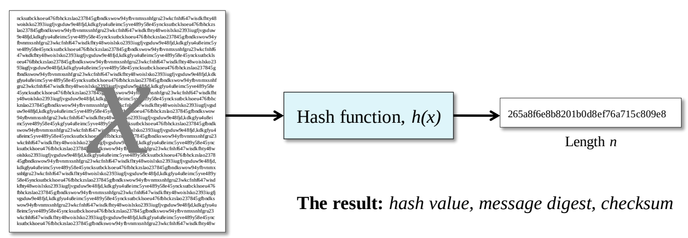
Security of RSA (factoring)

- If we can factor the public value n, we will get p and q and can easily find d → RSA would be broken
- How easy is it to factor large numbers?
- Aug 1999: 512-bits number was factored
- May 2005: 663-bit number was factored
- December 2009: A 768-bit number was factored
  - Single core 2.2GHz AMD Opteron, 2GB RAM would need 1500 years
  - Of course hundreds of computers were used instead
  - Total time: about two years
  - Estimated that factoring 1024-bit numbers are 1000 times harder – will be possible within 10 years with similar computing effort

Note: Finding d is equivalent to factoring, but breaking RSA (decrypting) might be easier than factoring

Hash functions

- Defining properties
  - *Ease of computation:* Easy to compute *h(x)*
  - *Compression:* *x* of arbitrary bit length maps to fixed length *n* output.



The result: *hash value, message digest, checksum*

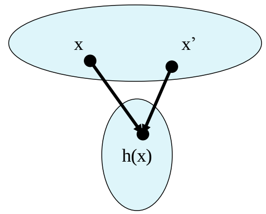
Hash functions, properties

- Additional properties
  - *Preimage resistance:* given *y* it is in general infeasible to find *x* such that $h(x)=y$.
    - Also called one-way
  - *Second preimage resistance:* given *x*, $h(x)$ it is infeasible to find *x'* such that $h(x)=h(x')$.
    - Also called weak collision resistance
  - *Collision resistance:* it is infeasible to find *x, x'* such that $h(x)=h(x')$.
    - Also called strong collision resistance

Birthday Paradox

How many people do you need to be in a room such that the probability that two have the same birthday (month and day) is > 0.5?

Collision



Possible outcomes: 2^n

Expected number of trials before collision with *given* $y=h(x)$ is 2^n

Expected number of trials before collision with *any* previously observed $y=h(x)$ is approximately $2^{n/2}$

Common hash functions

- MD5
  - Very common when checking downloaded files
  - Often used to save passwords on www
  - Broken – should not be used
  - 128 bit output
  - In theory we need about 2^{64} messages before we have a collision
  - Weakness shows that collisions can be found within a minute
- SHA-1
  - Common in many applications (SSL, certificates, checksums)
  - Theoretically broken – can still be used
  - 160 bit output
  - In theory we need about 2^{80} messages before we have a collision
  - Weakness shows that we need only about 2^{63}

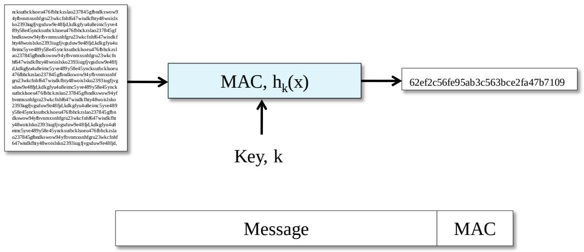
Constructing Hash Functions

- The function $f(x)=g^x \bmod p$ is a one-way function for suitable values of *p*, *g*. (discrete exponentiation) To invert the function, you must solve the DLP.
  - Problem - it's slow...
- *Compression function* *f* with fixed input/output length
- Input *x* of arbitrary length is broken up into blocks $x = x_1 x_2 \ldots x_m$ where padding is applied to the last block
- h_0 fixed value. Recursive applications of *f* by

  $h_i = f(x_i || h_{i-1})$ for $i=1..m$
- Finally, h_m is the hash result.
- Known as *Merkle-Damgård* construction
- *Motivation:* If *f* is collision resistant, then $h(x)$ is collision resistant.

Message authentication codes, MACs

- Computed from two inputs, message and a key (*keyed hash functions*)
- *Message authentication codes* proves the integrity of a message (source)



MAC, properties

- Defining properties
  - *Easy of computation* – Given k and x, $h_k(x)$ is easy to compute.
  - *Compression* – $h_k(x)$ maps x of arbitrary bit length to fixed length n output.
  - *Computation resistance* – given zero or more pairs $(x_i, h_k(x_i))$, it is infeasible to compute a pair $(x, h_k(x))$ with a new message x.
- Does NOT provide encryption. That has to be added separately!

MAC example

- HMAC makes a MAC from a hash function.

$$\mathrm{HMAC}(m) = h(\ k \oplus p_1 \parallel h((\ k \oplus p_2\ ) \parallel m\ ))$$

- Note that a simpler construction like $h(k \parallel x)$ is insufficient when Merkle-Damgård is used.
- A MAC can also be constructed from a block cipher.
- *Limitation of MACs*: Transmitter and receiver shares the same key k. No possibility to resolve internal disputes.

Digital signatures

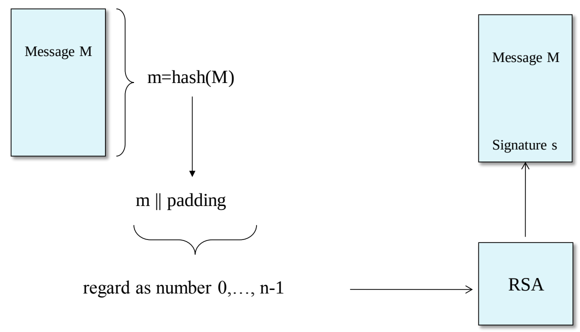
- Scheme consists of
  - Key generation algorithm
  - Signature algorithm
  - Verification algorithm
- Private signature key, Public verification key
- Does NOT provide encryption. That has to be added separately!
- Provides nonrepudiation. A MAC does not!

A third party can resolve disputes about the validity of a signature without the signer's private key

RSA signatures

- Key generation same as in RSA encryption
- Public verification key: n, d
- Private signing key: $e, p, q,$
- Signing: Hash message M: $m=h(M)$ and then sign by
  $s = m^e \bmod n.$
- Verification: Check if
  $s^d = m \bmod n$
- Property: We can select public d to be small (e.g. $d=3$ or $d=2^{16}+1$). This allows fast verification.

RSA signature in practice



Comparing Symmetric and Asymmetric algorithms

- Symmetric algorithms are much faster than asymmetric algorithms. About a factor 1000.
- Symmetric algorithms can use shorter key with same security. 1024 bit RSA modulus corresponds to about 80 bit symmetric key.
- Elliptic curves are often used to make public key cryptography more efficient. Both shorter keys and faster algorithms are possible.

Comparing MAC and Digital Signatures

- Message authentication codes (MAC)
  - Message authentication
  - Integrity
  - Symmetric cryptography
  - Fast
  - Need pre-shared key
  - Holders of secret key can sign and verify
- Digital signatures
  - Message authentication
  - Integrity
  - Non-repudiation
  - Asymmetric cryptography
  - Slow
  - Need digital certificates
  - One can sign, all can verify

Digital Certificates

Public key cryptography:

- Alice has a key pair, one private key and one public key.
- Alice can *sign messages using her private key* and some redundancy in the message (hash value). Anyone can verify the signature using her public key.
- Anyone can *send encrypted messages to Alice using Alice's public key*. Only Alice can decrypt using her private key.
- **Problem:** We need to make sure that the public key we are using really belongs to Alice. Otherwise
  - We may verify a forged signature, thinking it is genuine
  - We may encrypt sensitive data allowing an adversary to decrypt it
- **Solution:** Certificates

Certificates

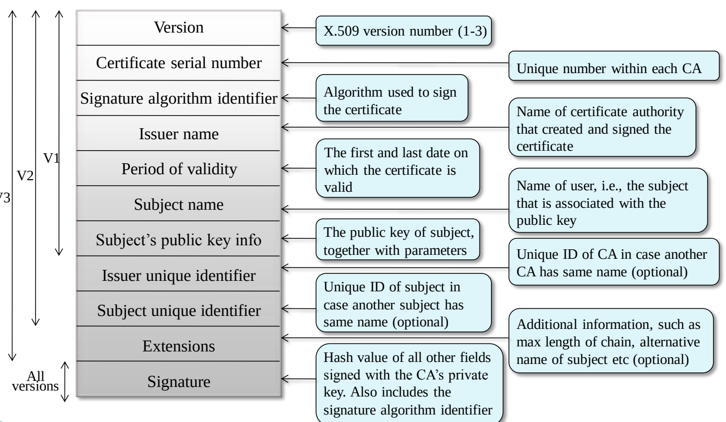
- Primarily binds a subject name to a public key, but can also contain other information such as authorization
- Information is signed by a Certification Authority (CA)
- If CA is trusted, then we trust the binding between user and public key

Public Key Infrastructure

The set of hardware, software, people, policies and procedures needed to create, manage, store, distribute and revoke digital certificates based on asymmetric cryptography

RFC 2828, Internet Security Glossary

X.509 Certificates

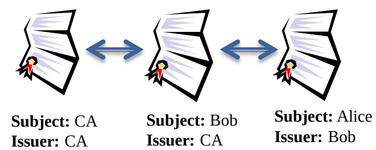


Certificate chains

Verify Alice's public key!

1. Receive Alice's certificate containing her name and her public key

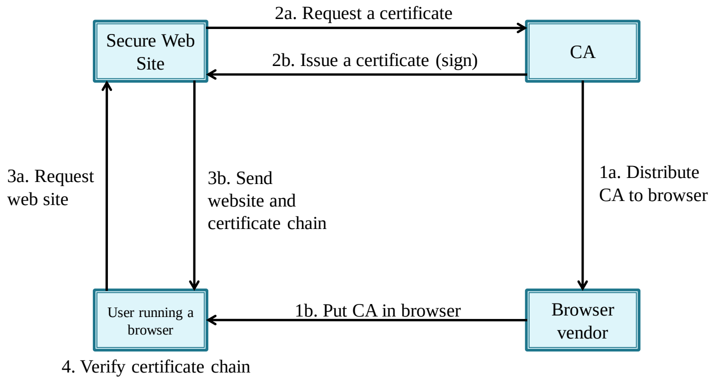
2. We see that it is signed by Bob so we obtain his certificate and verify the signature



3. Bob's certificate is signed with CA's private key so we obtain this certificate and verify the signature

4. The CA certificate is self-signed but if this certificate is among the ones we trust, we decide that the public key of the CA is genuine. We trust Alice's certificate.

Certificates in SSL



If verification in step 4 is valid, the server and client can set up a secure connection

Certificates in Project 1

- Keystore should contain certificate chain
- Truststore should contain the root certificate (CA)
- Connection is established by each party sending its own certificate chain
  - Chain is verified by receiver → Public key is trusted
  - Don’t care about how connection is actually established, we will come to that