Security on the Internet

Outline

- Internet security threats
- HTTP authentication
- Hash functions and encryption
- Public key encryption
- Pretty Good Privacy
- Transport Layer Security and HTTPS
- Digital certificates

Pervasive Computing

- social media
- phones
- cars
- medical devices
- cash

    - The Internet of Things (IoT)

Computer Security

- The engineering of systems that exhibit the following properties:
  - Confidentiality
  - Integrity
  - Availability

Confidentiality

- Non-disclosure of information except to another authorised agent

Integrity

Every piece of data is as the last authorised modifier left it.

- Data integrity:

  ensuring that the data has not been deleted or altered without permission.

- Software integrity:

  ensuring that the software programs have not been altered, whether by an error, a malicious user, or a virus.

Availability

- being accessible and usable upon demand by an authorized entity.

Computer Security and the Internet

The internet can make attacks easier because of:

- Action at a Distance
- Technique Propagation
- Automation, e.g. through harnessing distributed computation

Vulnerabilities and exploits

- Most software systems do **not** exhibit these properties.
- When a software system is not secure, it has one or more *vulnerabilities*.
- Malicious users attack vulnerable systems through *exploits*.
- There is a one-to-many mapping between vulnerabilities and exploits.
- An exploit is a piece of code, or a replicable procedure, which is able to exploit the vulnerability.
- A given vulnerability may have zero or more known exploits.

Vulnerability Announcements

- All software has vulnerabilities.
- Some of these vulnerabilities are known, others are not.
- Of the known vulnerabilities, different parties may have different knowledge.
- When a vulnerability is detected by the security community, e.g. CERT, software vendors are informed.
- There is a typically grace period before the vulnerability is announced to the general public.
  - https://www.us-cert.gov/ncas/alerts.xml
  - https://web.nvd.nist.gov/view/vuln/search

Zero-Day Vulnerabilities

- Vulnerabilities that have not been publicly unannounced are called "zero-day".
- Attackers users who have knowledge of a zero-day vulnerability and a corresponding exploit are in a very powerful position.
- Government agencies bid for zero-day vulnerabilities and exploits on the black market

Authentication

- In order to defend our system from unauthorised users, we must distinguish between authorised and unauthorised users.
- The process of determining the true identity of a user is called authentication.
- The simplest technique for authentication is to use a secret password or pass-phrase.

Access control

- Preventing illegitimate access can be split into two issues
  - **Authentication**: Determining who is trying to gain access to your host
  - **Access Control**: Determining whether that individual is allowed to access a resource on your host

Web Authentication

- Authentication mechanisms of web servers prevent illegitimate access to resources
- Resources are often grouped into named **realms**, which users can be allowed access to
- They require clients to demonstrate who they are, from extra data sent with their messages

Authentication HTTP status

- If a client tries to access a secured web server with no authentication, it receives a response with status code:

  401 Authentication Required

- This tells the client to supply identification

- The response will contain a field:

  WWW-Authenticate: ...

- specifying the authentication scheme required
- The client provides authentication in new request

Proof of identity

- Common forms of proof of identity are:
  - Username plus password
  - Public-Key Cryptography and Digital certificates

- Software applications (agents) must also authenticate themselves, and may have identities different from but based on that of their owner

HTTP basic authentication

- Basic authentication is indicated by the Basic scheme being passed in the 401 response:

  WWW-Authenticate: Basic realm="somerealm"

- Basic secure request gives credentials as base64 encoding of text

  username:password

- Encoding put into Authorization request field:

  Authorization: Basic QWxhZGR=

Hash functions

- A **hash** of some data is a transformation of that data into a fixed length string, from which the original cannot be deduced:
  - I.e. it is a one-way function
- Use a hash wherever data across places or times need to be compared to check they are the same: if hash values match this is **almost** proof
- Value of algorithm depends on:
  - Difficulty of deducing original from digest
  - Lack of **collisions**: 2 messages with same digests

Password Hashing

- When a server authenticates a user using a password, it compares the supplied password against a field stored in the database.
- If we store the original password, then if an attacker breaks into the server, they can steal all the users' passwords.
- Passwords are often reused across different servers.
- To prevent this, we store a hash of the password instead.
- We also include random salt (also called a nonce) to prevent rainbow attacks.
- On Unix, the file `/etc/shadow` contains hashed passwords.

Password authentication

- The client sends the username and password (in some form) with every request, and the web server matches against its list to see whether access is allowed.
- Eavesdropping is a problem with this if the authentication occurs over an unencrypted channel.

MD5

- MD5 is a hash function devised by Ronald L. Rivest
- MD stands for Message Digest (another word for hash).
- It takes an arbitrarily long string and produces a string of fixed length, the hash value or *digest*.
- Collisions are possible in MD5.
- Can try it out with UNIX command md5sum.

```
"digest"  abfd2c0ecb4e9dec4a6b1159d5fea334
"digest"  5a20c77381e982467465dd18facf0807
"digest " e21681785dc42cfc30867e4fcf78edaf
```

SHA-2

- MD5 should *no longer used* for cryptographic applications, as it was shown to be insecure.
- The current standard is SHA 2
- MD5 is still considered to be secure for password hashing in the case of legacy applications
  - New applications should make use of SHA-2

HTTP digest authentication

1. Client tries to access a realm on the server
2. Server responds asking for authentication (HTTP 401) using digests and providing a unique identifier, a number used once (nonce), for the request
3. Client sends a digest of the concatenation of:
   - username, realm, password, URL, request method and nonce id

HTTP digest authentication

- Digest authentication is indicated by the Digest scheme passed in the 401 response:

  WWW-Authenticate: Digest

  realm="somerealm" algorithm=”MD5”

  nonce=”564dsd” ...

- Create digest of password plus other data
- Digest put into Authorization request field:

  Authorization: Digest Username=“Simon”,

  response="6629fae49393a053",

  realm=“somerealm”, nonce=”564dsd” ...

Eavesdropping

- Eavesdropping is a problem on the internet:
  - Happens at any point in route between hosts
  - Whole physical network cannot (realistically) be secured
- Promiscuous Mode: Ethernet allows hosts to request that all data sent through a network be passed to it, even if not a router or destination.
- A sniffer is a device/program to monitor all data across a network, and so can be used to extract passwords etc.
- **Encryption** is the transformation of data to a form unreadable by anyone but the intended recipient.
- The algorithm for performing the transformation is called a **cryptographic cipher**.

Encryption types

- Encryption on the internet takes several forms:
  - **Link encryption**: Encrypts all communication across a physical link, but it is expensive and unrealistic over large scale
  - **Document encryption**: Documents encrypted, sent, then decrypted by document-handling applications
  - **Transport Layer Security**: TLS encrypts all messages at the TCP layer

Hashes and encryption

- Both a hash and an encryption of a message are a transformation of that message into some new data that gives no clue to the original.
- The original message can be computed from the encrypted data (decrypted) given the right information, but this is not true of a hash.

Ciphers

- A cryptographic cipher provides an encryption

$$C = E(M, K)$$

- And a decryption function:

$$M = D(C, K)$$

- The original message is called the plain-text.
- The encrypted message is called the cipher-text.
- The decryption function can only be computed if you have access to the correct secret key K.

Single-key encryption

- A **key** is a secret piece of data that sender and receiver use to encrypt and decrypt messages
- As no-one else knows the key, no-one else can decrypt the message
- In single key encryption, the same key is used by both parties to both encrypt and decrypt.
- Another name for single-key encryption is symmetric cryptography.
- An example of a symmetric cipher is the Advanced Encryption Standard (AES).

Key Sizes

- The space of possible keys must be large.
- This is in order to prevent an attacker simply trying every possible key
  - This is called a brute-force attack.
- For a symmetric cipher such as AES:
  - Maximum key size is 256 bits,

$2^{256} =$
115792089237316195423570985008687907853269984665640564056
4039457584007913129639936 possible keys.

Key Generation

- Keys must be hard to guess.
- Therefore they should be randomly chosen.
- It is very hard to generate genuinely random data using a deterministic digital computer.
- Usually the best we can do is to use a Pseudo-Random Number Generator (PRNG).
- In cryptography we make use of external sources of randomness (entropy) such as heat in order to create Secure-Random Number Generators.
- On Unix, see `/dev/urandom`

Key Management

Problems with symmetric ciphers:

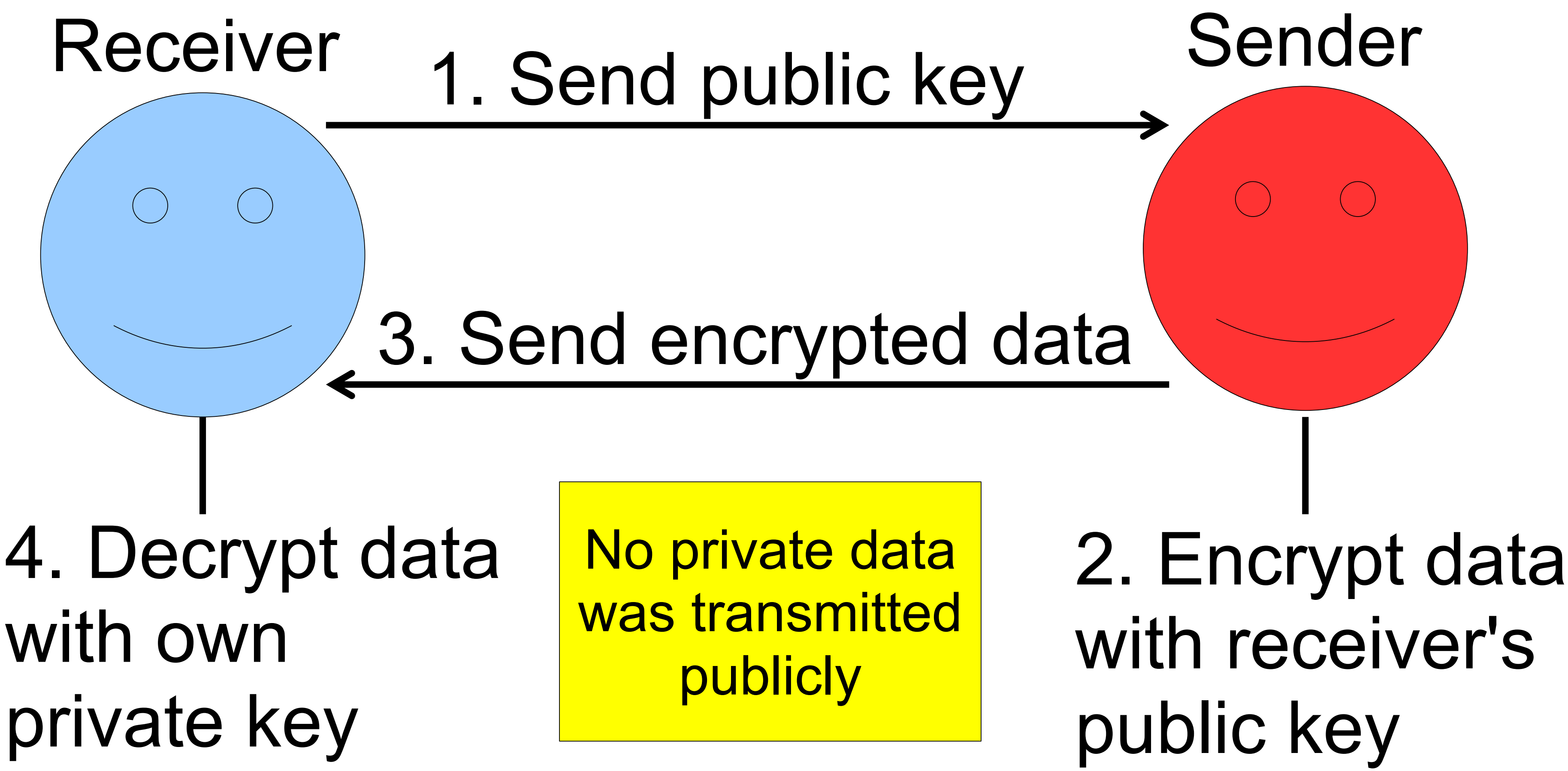
1. We need a separate key for every possible pair of users.
2. We need to agree on a key with the other party.

- We cannot simply send the key, because of eavesdroppers.

Public key encryption

- In public key cryptography, each user has a pair of keys: a **public** one and a **private** one.
- These are called *asymmetric* ciphers.
- The public key can be made available to anyone wanting to send an encrypted message to the user
- Sender encrypts their messages with the public key: the algorithm ensures the message can only be decrypted with the private key
- On receiving an encrypted message, the user uses the private key to decrypt the messages

Scenario 1: Secret message



RSA

- One of the best known public key encryption algorithms is RSA, named after the inventors: Rivest, Shamir and Adleman
- The idea of RSA is to use two very large prime numbers for the keys

RSA key generation

1.	Generate two large primes:	p, q
2.	Calculate the product:	$n = pq$
3.	Calculate the totient:	$m = (p - 1)(q - 1)$
4.	Find a co-prime to totient m:	e
5.	Choose integers d, i so that:	$d = (1 + im) / e$
6.	The public key is	(n, e)
7.	The private key is	(n, d)

RSA generation example

1. Generate two large primes: *7, 11* (p, q)

RSA generation example

1. Generate two large primes:	*7, 11*	(p, q)
2. Calculate the product:	*77*	(n = pq)

RSA generation example

1. Generate two large primes:	*7, 11*	(p, q)
2. Calculate the product:	*77*	(n = pq)
3. Calculate the totient:	*60*	m = (p - 1)(q - 1)

RSA generation example

1. Generate two large primes:	*7, 11*	(p, q)
2. Calculate the product:	*77*	(n = pq)
3. Calculate the totient:	*60*	m = (p - 1)(q - 1)
4. Find a co-prime to totient:	*7*	e

To find co-prime, count up each prime, and find first not divisible into totient:

Is 60 divisible by 2? Yes

Is 60 divisible by 3? Yes

Is 60 divisible by 5? Yes

Is 60 divisible by 7? No

RSA generation example

1. Generate two large primes:	*7, 11*	(p, q)
2. Calculate the product:	*77*	($n = pq$)
3. Calculate the totient:	*60*	$m = (p - 1)(q - 1)$
4. Find a co-prime to totient:	*7*	e
5. Choose integers *i, d:*	*5, 43*	
	(1 + (5 x 60)) / 7 = 43	$d = (1 + im) / e$

To find i and d, count up i from 1 until find first
(1 + (i x 60)) divisible by 7:

(1 + (1 x 60)) divisible by 7? No

(1 + (2 x 60)) divisible by 7? No

...

(1 + (5 x 60)) divisible by 7? Yes, d = 43

RSA generation example

1. Generate two large primes:	*7, 11*	(p, q)
2. Calculate the product:	*77*	(n = pq)
3. Calculate the totient:	*60*	m = (p - 1)(q - 1)
4. Find a co-prime to totient:	*7*	e
5. Choose integers *i, d:*	*5, 43*	
	(1 + (5 x 60)) / 7 = 43	d = (1 + im) / e
6. The public key is (n, e):	*(77, 7)*	(n, e)
7. The private key is (n, d):	*(77, 43)*	(n, d)

RSA encryption algorithm

1. Obtain receiver's public key *(n, e)*
2. Convert message to an array of bits representing a large integer
   $$M < 2^n$$
3. Compute encrypted values $C = M^e \bmod n$

---

1. Obtain receiver's public key *(77, 7)*
2. Represent data as +ve integer *6*
3. Compute encrypted value $6^7 \bmod 77 = 41$

RSA decryption algorithm

1. Use private key (n, d)
2. Receive encrypted message C
3. Calculate original value $M = C^d \mod n$

---

1. Use private key $(77, 43)$
2. Receive encrypted message 41
3. Calculate original value $41^{43} \mod 77 = 6$

RSA signing

- RSA can also be used to digitally sign a document.
- The signer uses their private key (n, d) to produce a signature.
- The verifier can check this using the signer's public key.
- The signature S of a message M is:

$$S = M^d mod\ n$$

- To verify the signature, the verifier checks that:

$$S^e = M\ mod\ n$$

- We typically hash the message, and then sign the hash.
- MD5 is not suitable for digital signatures- use SHA-2 or SHA-3.

Hybrid crypto-systems

- For large messages RSA is typically used in combination with a symmetric cipher such as AES.
- RSA can be used encrypt messages containing the symmetric keys, which can then be distributed over a public channel.
- The symmetric key is typically a temporary key that it is only valid for a particular session.

Transport Layer Security

- Secure Socket Layer (SSL) was developed by Netscape Communications
- It operates between host-to-host protocols (TCP) and the application layer protocols (e.g. HTTP)
- For each communication, SSL uses the most recent secure communication protocol that both hosts can support
- Transport Layer Security (TLS) is a more recent variation on SSL, standardised by IETF

TLS negotiating protocol

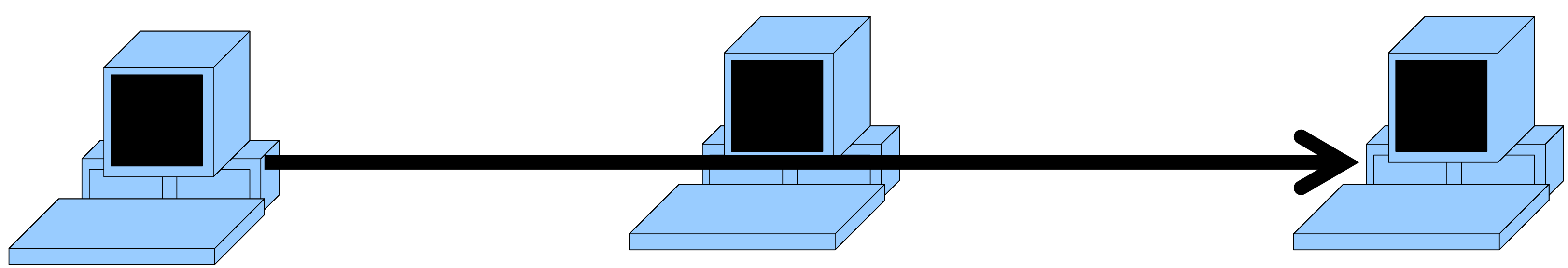
- TLS initiates a cryptographic protocol between hosts with a Hello message
- Both parties declare what they can support and the strongest encryption available is chosen
- This allows for the change and development of encryption methods
- The client will choose the stronger of the two protocols

TLS sharing certificates

- The hosts then exchange **certificates**
- Digital certificates provide verifiable host data for authentication
- They also provide public keys for encrypting the communication
- We'll say more on certificates later
- The public keys in the certificates are used to encrypt communication over TLS

Man-in-the-middle attacks

- TLS tackles the problem of man-in-the-middle attacks
- A malicious host routes communication through itself, without being apparent to either sender or receiver
- Data can then be copied and, if not encrypted, read



TLS and the internet architecture

- TLS operates over TCP, but under HTTP or other application protocols
- Inserts a new layer into the four-layer internet layering model
- This layer deals with the issue of secure communication of application data

HTTPS

- HTTPS is HTTP over SSL/TLS
- Uses its own URI scheme
- https:...
- Has a different default TCP port (443)
- Otherwise the same as HTTP over TCP
- An HTTPS web server must have a digital certificate that it can use to authenticate itself with a client

Digital certificates

- A digital certificate is a block of data about a communicating host that is signed
- Signing a certificate means adding an encrypted hash of the host data, so that other hosts can check that:
  - You are who you say you are
  - The host data has not been tampered with

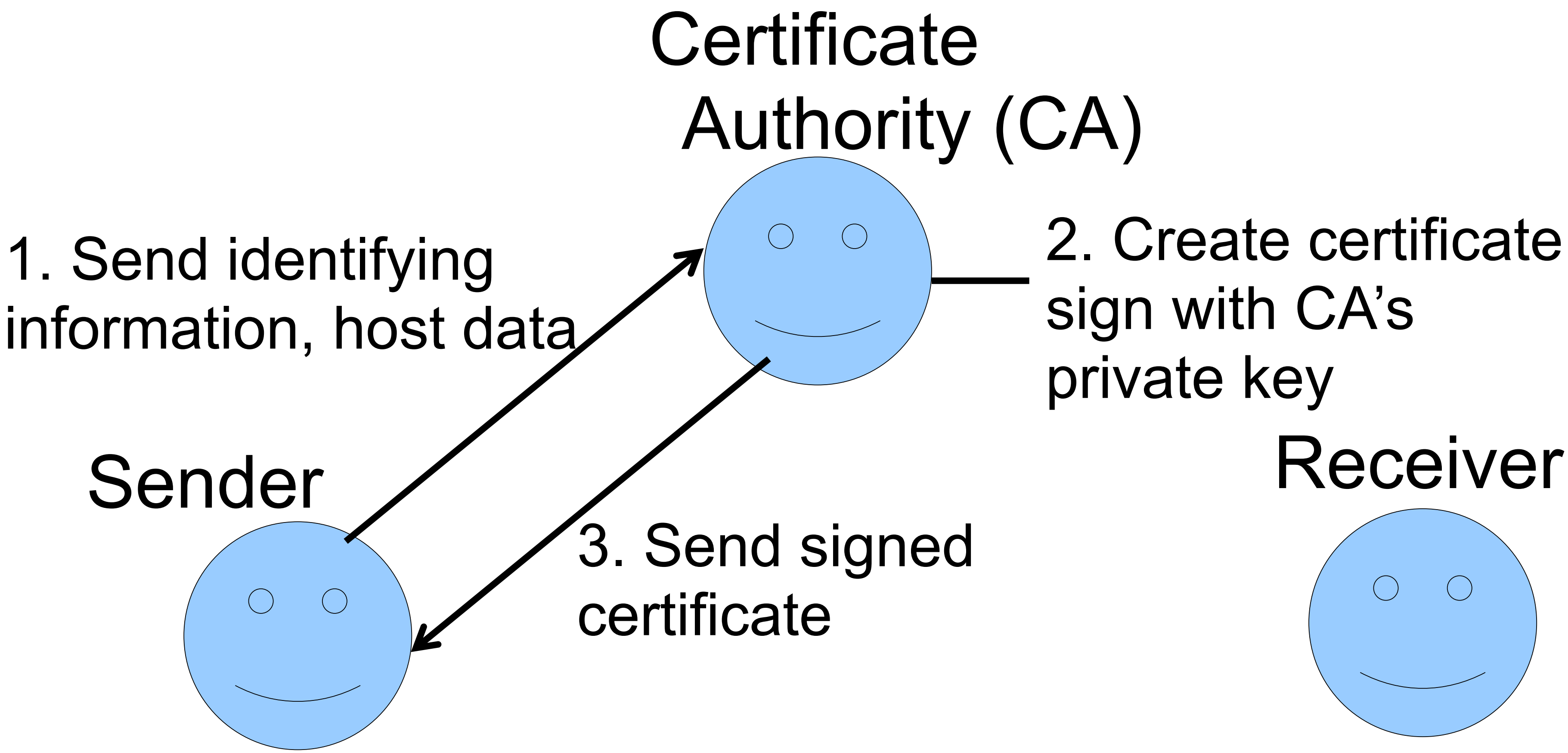
Host data

- Host data can include:
  - Public key
  - Validity period of certificate
  - URL of revocation centre
  - Name, institution, email address of owning user
  - Public key is used to secure communication to the host
  - Certificates are revoked if they are suspected of being compromised (like credit cards)
  - Revocation centres provide lists of revoked certificates to check against

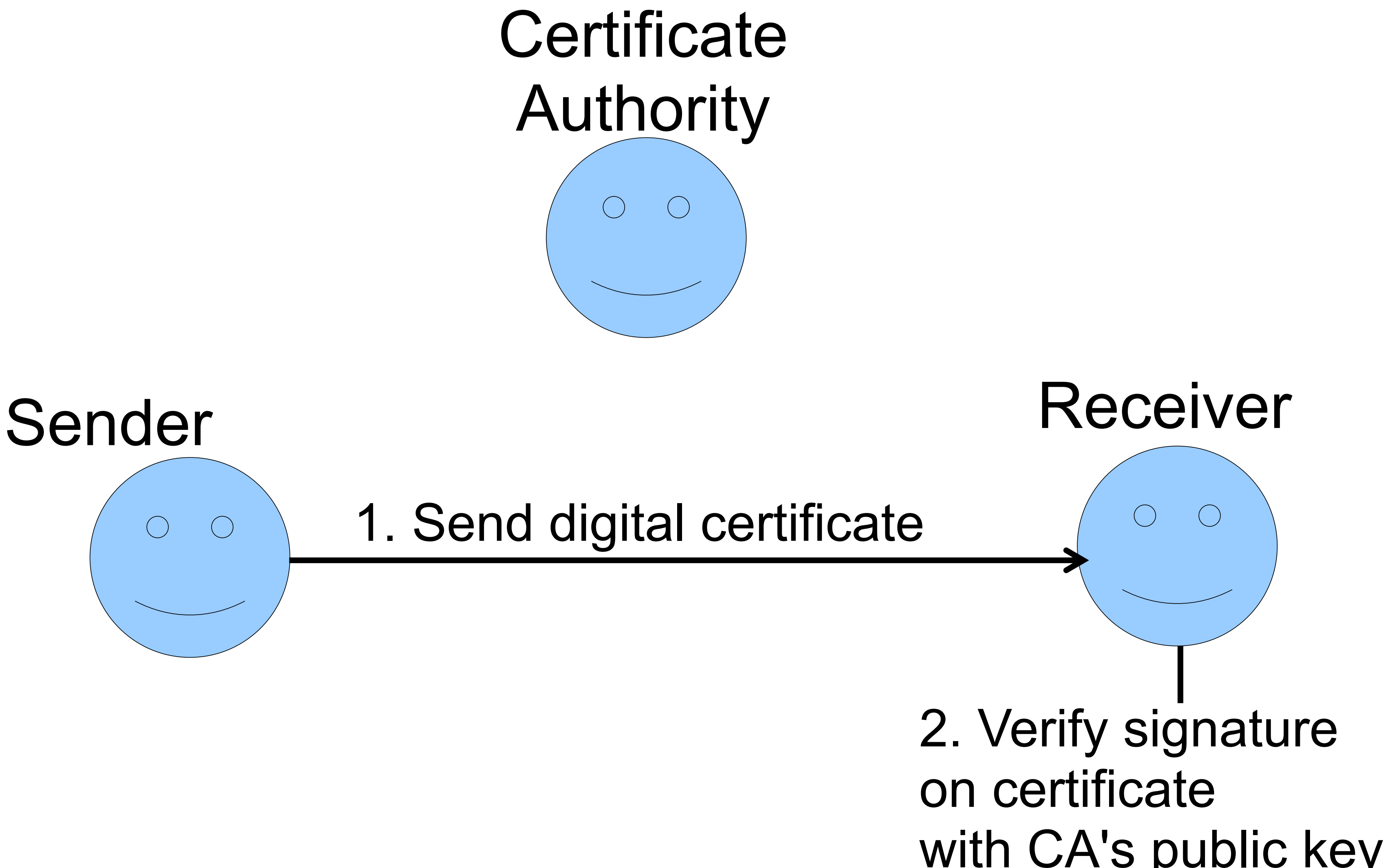
Certification authorities

- A certification authority is an organisation responsible for issuing and verifying the correctness of certificates
- If a host's certificate is signed by a CA, then any other host trusting the CA may reliably know that the certificate's public key belongs to the host as stated
- Publicly trusted CAs exist, such as VeriSign and CertCA

Applying for a certificate



Applying for a certificate



X.509

- X.509 is a popular form of certificate
- An X.509 certificate consists of three parts:
  - The certificate details
  - The signature of the certificate
  - The algorithm used to sign the certificate
- The certificate details then include:
  - A unique serial number for the certificate
  - The period (from X to Y) that the certificate is valid
  - The name of the certificate's issuer
  - A unique identifier for the issuer
  - The name of the certificate's owner
  - The public key of the owner

What we've covered

- Internet security threats
- HTTP authentication
- Digests and encryption
- Public key encryption
- Transport Layer Security and HTTPS
- Digital certificates