UNIVERSITY OF
Southampton

UNIVERSITY OF Southampton

Securing HTTP

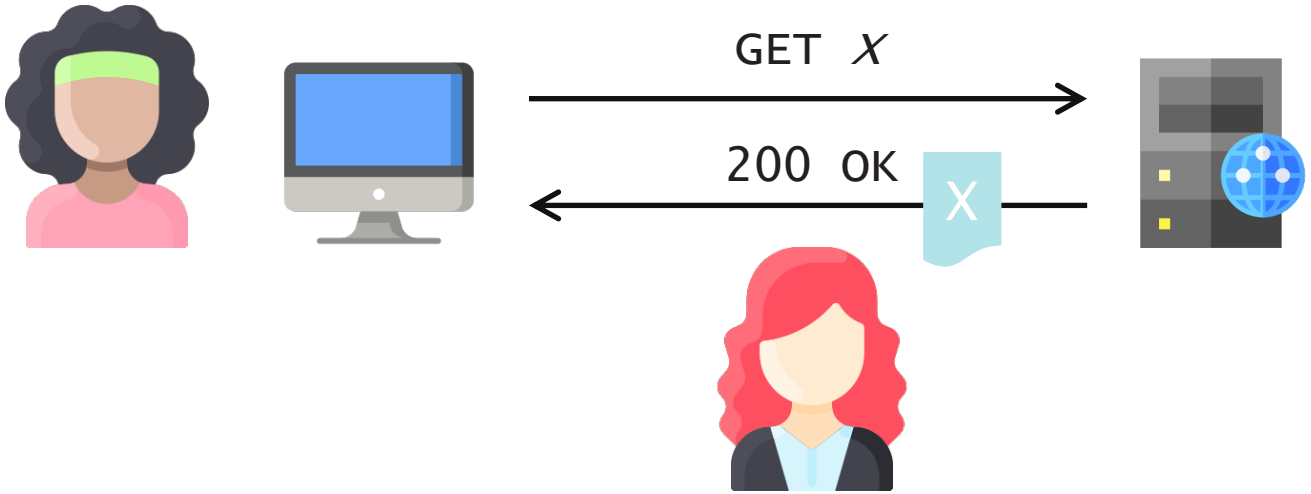
COMP3220 Web Infrastructure

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

Securing HTTP

As originally designed, HTTP sends all data in the clear

Vulnerable to interception by third parties



Transport Layer Security

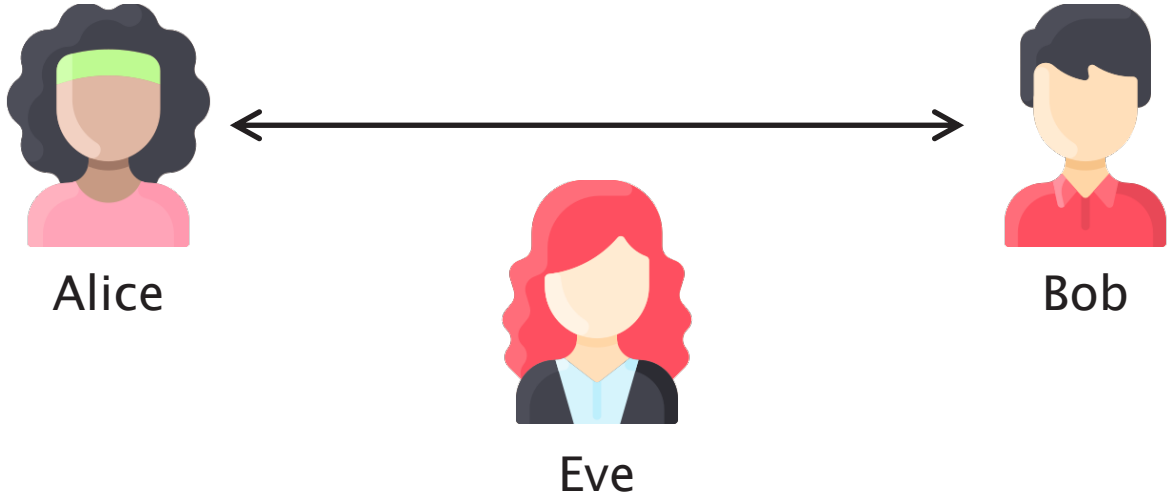
The foundation for Secure HTTP (HTTPS)

- Formerly known as Secure Sockets Layer (SSL) – dates back to 1995
- Protocol for establishing secure communications channels between internet hosts
- Also used for protocols other than HTTP: IMAP, POP3, FTP, SMTP

Despite the name, considered an application layer protocol
(in the OSI 7-layer model, a session layer protocol)

Cryptography 101

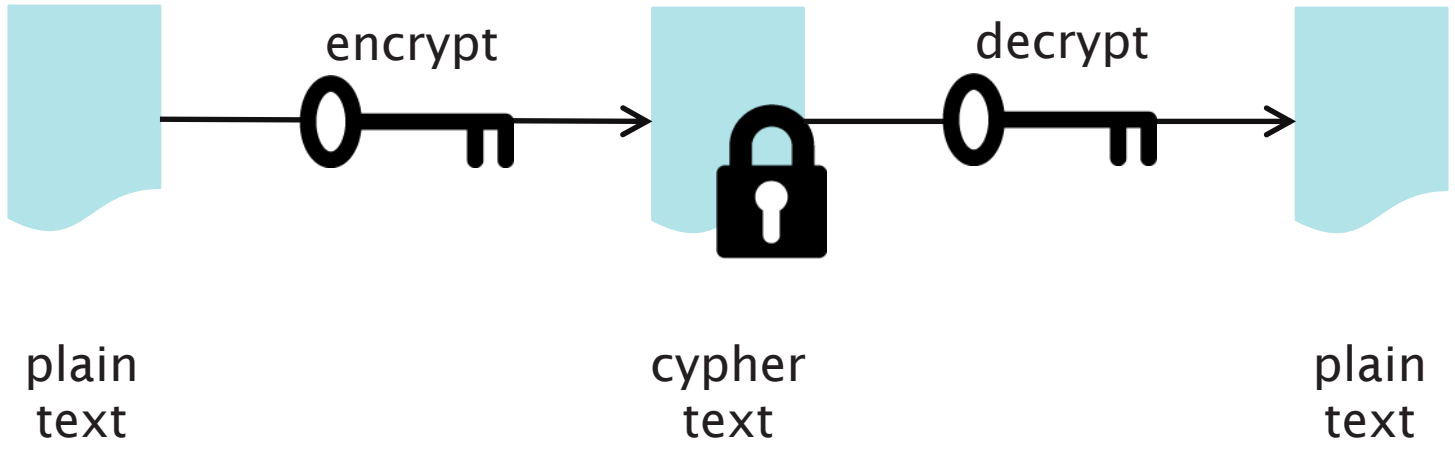
- Confidentiality – data cannot be read by unauthorised parties
- Integrity – data cannot be modified by unauthorised parties without being detected
- Authentication – authorised parties can prove who they are
- Non-Repudiation – the author of data cannot deny that authorship



Symmetric-key Encryption

Uses a single key for both encryption and decryption

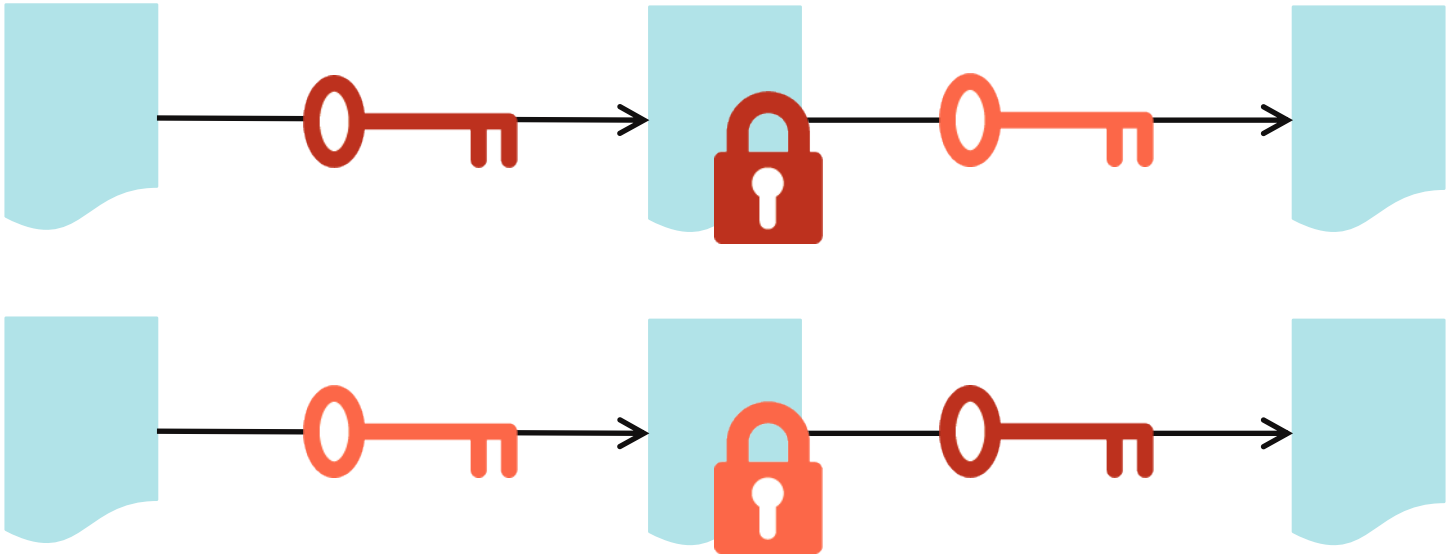
- Generally fast
- Examples: AES, IDEA, 3DES
- Key exchange is an issue



Asymmetric-key Encryption

Keys are generated in pairs

- Each key can decrypt what the other has encrypted
- Given one key from a pair, cannot work out the other key
- Generally slower than symmetric encryption
- Examples: RSA, ElGamal, Elliptic-Curve Cryptography



Public Key Cryptography

Commonly-used term for asymmetric-key encryption

- Refers to the different roles of the keys in a pair

Public key

- Shared with all by the owner
- Used to encrypt messages sent to the owner

Private key

- Kept a secret by the owner
- Used by the owner to decrypt messages sent to them

Cryptographic hash algorithms

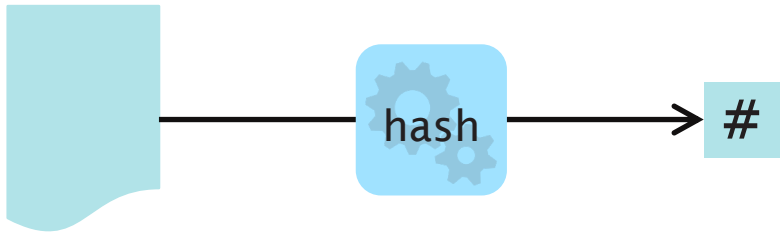
Algorithm that turns an arbitrary-sized message into a fixed size bit string (the digest or hash, typically 256 or 512 bits long)

Lossy transformation

- Given a digest cannot easily calculate the corresponding message
- Brute force attacks, rainbow table attacks
- Used to ensure integrity

Modern hash algorithms: SHA-2, SHA-3

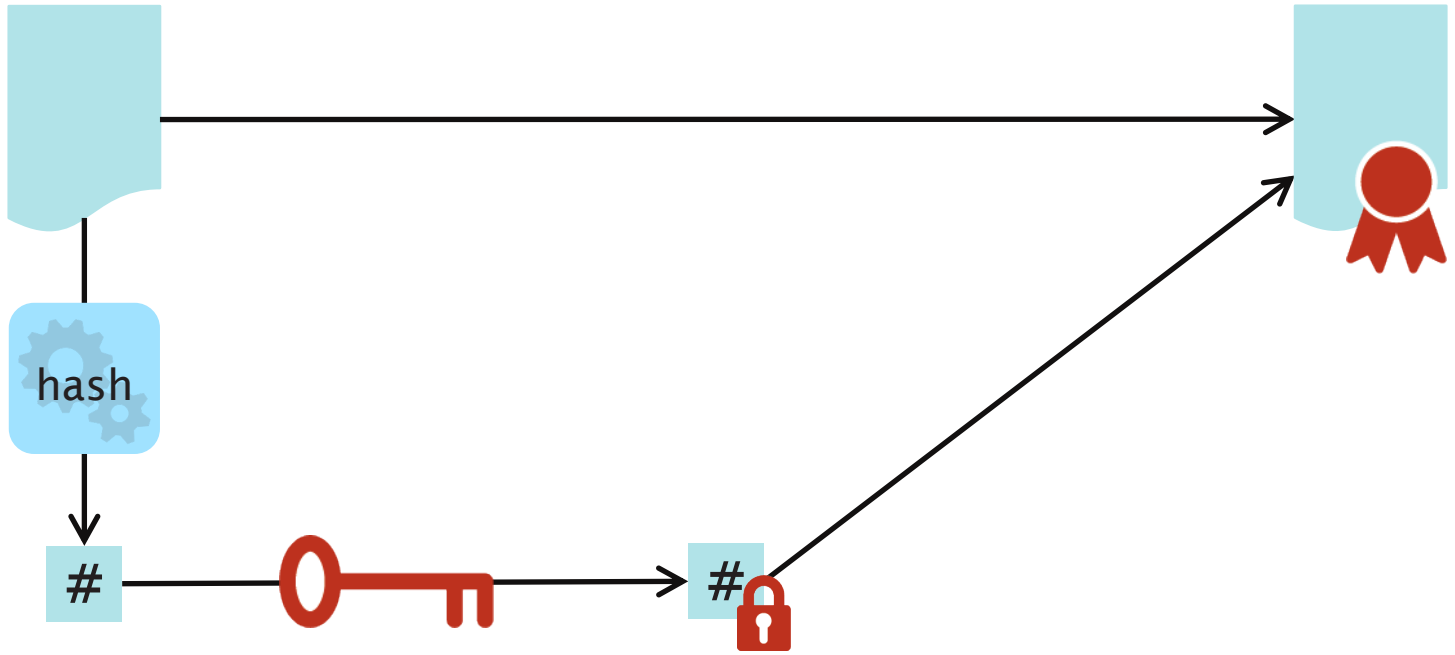
Older insecure hash algorithms: MD5, SHA-1



Digital signatures

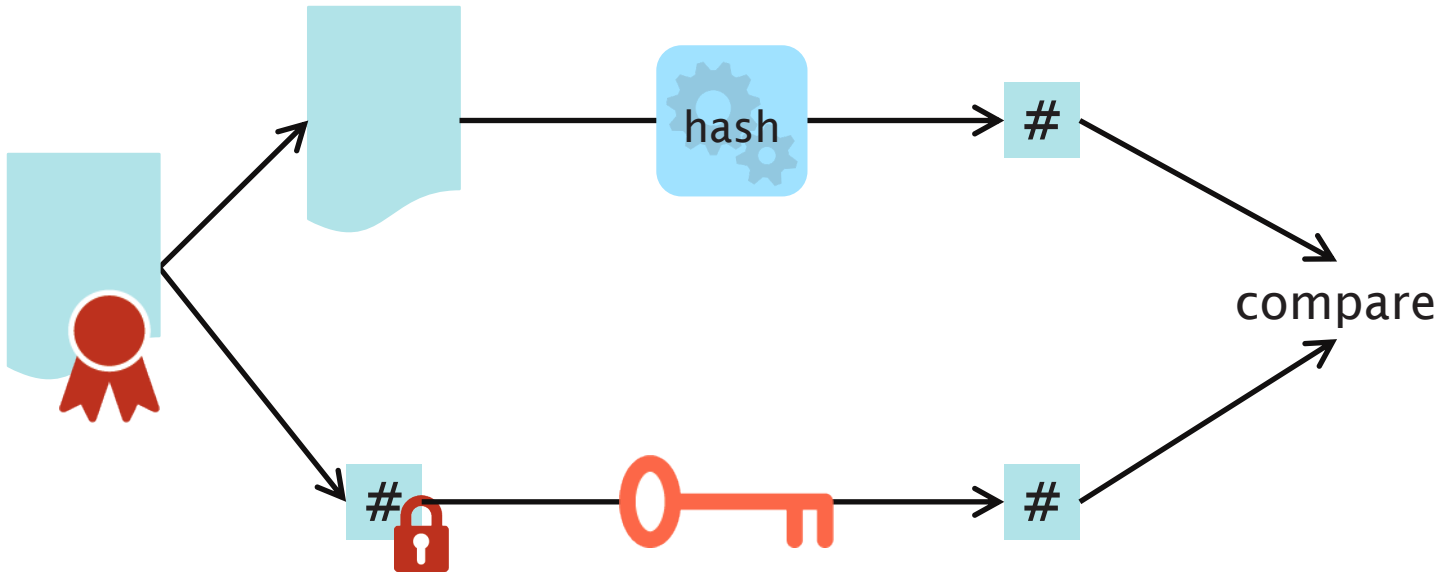
Authentication, non-repudiation, integrity of messages

1. Generate a cryptographic hash of the message
2. Encrypt the hash with your private key
3. Attach the encrypted hash to the message



Signature verification

1. Decrypt the encrypted hash with the public key
2. Generate a cryptographic hash of the message
3. Compare the hashes



Certificates

A public key that has been digitally signed by a trusted third party
(a Certificate Authority or CA)

- Used to make guarantees about ownership of a public key
- CA public keys typically incorporated into browsers or operating systems

Key exchange

Symmetric cryptography needs a shared key

How can two parties agree a shared key over an insecure channel?

Diffie-Hellman key exchange

By Ralph Merkle, based on earlier work by Whitfield Diffie and Martin Hellman

Relies on each participant being able to apply an operation that is secret, commutative, and difficult to reverse

- Difficult to reverse = trapdoor function
- In its original implementation, uses multiplicative groups of a finite field
- $(g^a \bmod p)^b \bmod p = g^{ab} \bmod p = \left(g^b \bmod p\right)^a \bmod p$
- Far easier to calculate exponents than logarithms in a finite field

Can be generalised to other finite cyclic groups

- For example, elliptic curve groups

Diffie, W. and Hellman, M. (1976) New Directions in Cryptography. IEEE Transactions on Information Theory. 22(6). pp. 644-654.
Merkle, R.C. (1978) Secure communications over insecure channels. Communications of the ACM. 21(4). pp.294-299.

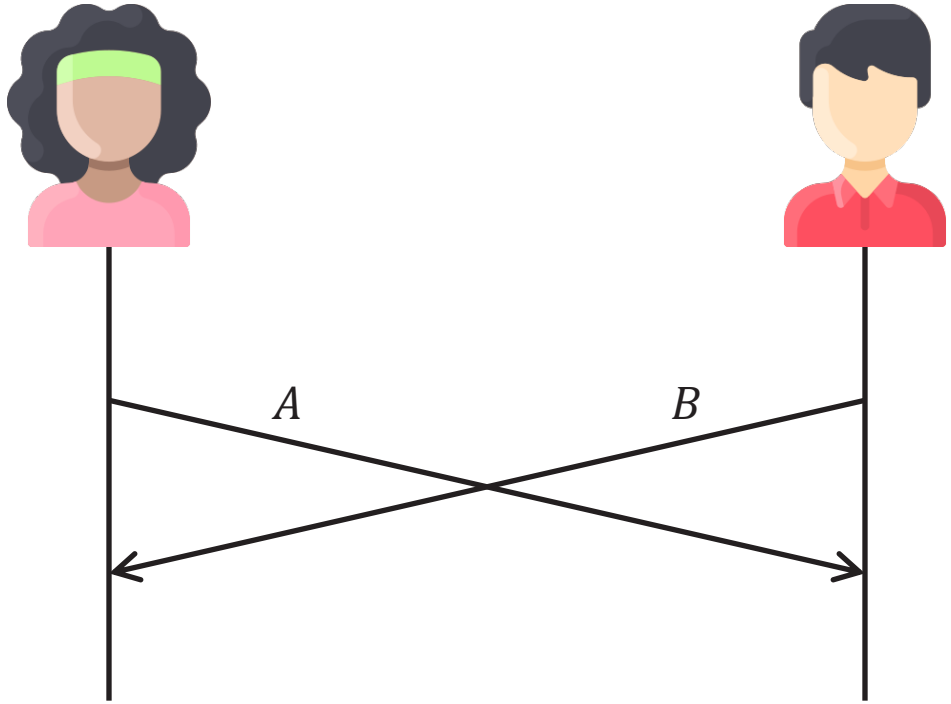
Diffie-Hellman key exchange

p is a large prime number

g is a primitive root modulo p
(i.e. for every integer a coprime to p, there is an integer k such that $g^k = a \bmod p$

Pick random large integer a

$A = g^a \bmod p$



Pick random large integer b

$B = g^b \bmod p$

$Z = B^a \bmod p$

$Z = A^b \bmod p$

Transport Layer Security

Symmetric-key encryption is much faster than asymmetric-key encryption

TLS is a hybrid cryptosystem

- Uses asymmetric-key encryption to agree on a shared symmetric key
- Uses symmetric-key encryption for subsequent application data, using the shared key

TLS 1.2 defined in Aug 2008, TLS 1.3 defined in Aug 2018

- TLS 1.3 offers significant security and efficiency improvements over TLS 1.2
- TLS 1.3 is widely supported by most modern browsers (notable exception: Microsoft)
- Adoption by websites is much lower (<40% as of September 2020)

Dierks, T. and Rescorla, E. (2008) *The Transport Layer Security Protocol Version 1.2*. RFC5246. Available at: https://tools.ietf.org/html/rfc5246
Rescorla, E. (2018) *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC8446. Available at: https://tools.ietf.org/html/rfc8446

Transport Layer Security handshake

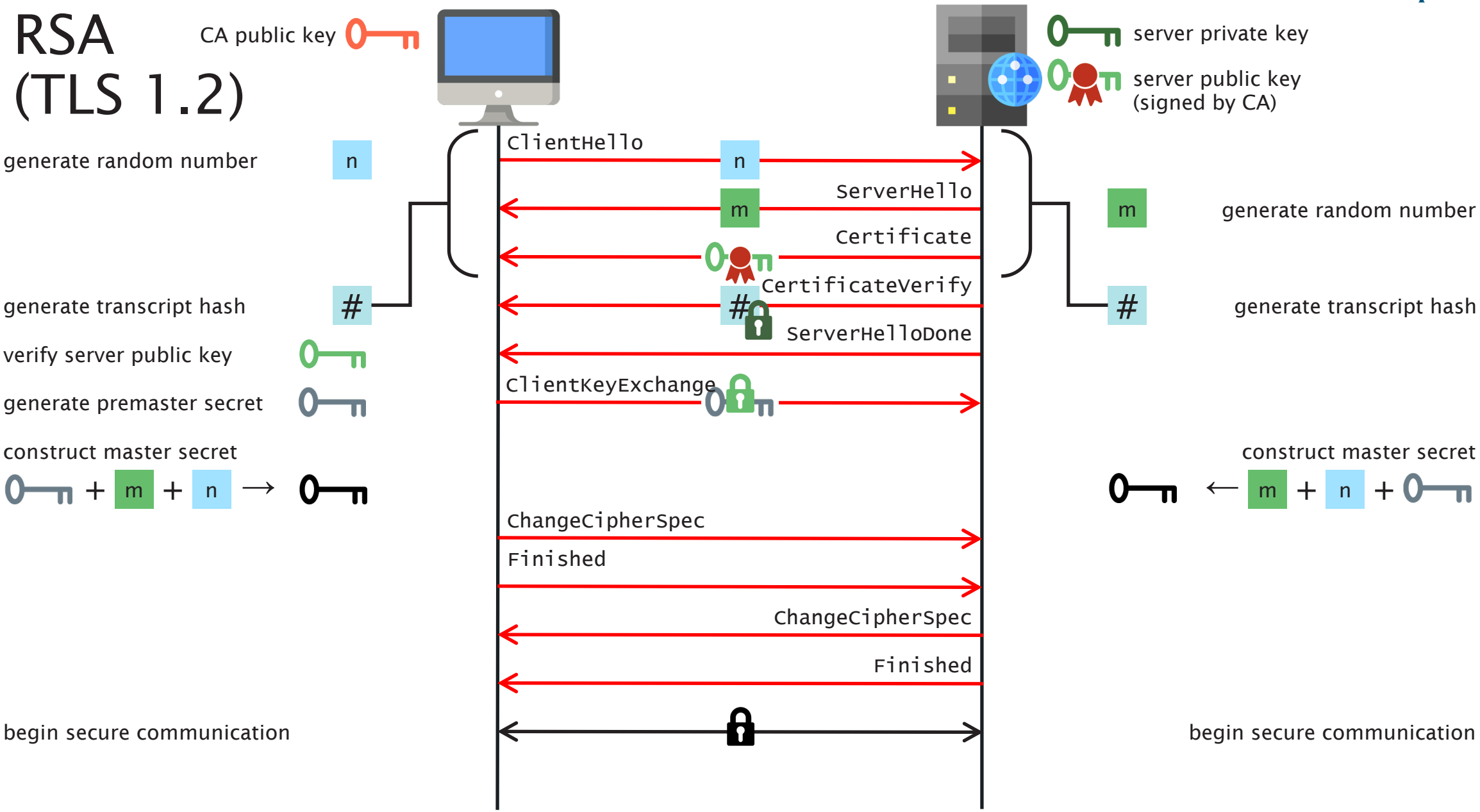
TLS handshake is the protocol used to agree on the shared symmetric key

Several key exchange algorithms in use – these are the most common cases:

- RSA (TLS 1.2 and earlier only)
- Ephemeral Diffie-Hellman (TLS 1.3 onwards)

Because TLS 1.2 is still in wide use, we'll look at both TLS 1.2 and TLS 1.3

RSA (TLS 1.2)


CA public key
server private key
server public key (signed by CA)
generate random number
n
ClientHello
n
ServerHello
m
m
generate random number
Certificate
CertificateVerify
generate transcript hash
#
#
generate transcript hash
verify server public key
ServerHelloDone
generate premaster secret
ClientKeyExchange
construct master secret
+ m + n →
construct master secret
← m + n +
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
begin secure communication
begin secure communication

Perfect forward secrecy

As shown, protocol ensures the confidentiality of the subsequent communications

What happens if the server's private key is leaked at some point in the future?

- With a copy of the messages from ClientHello to ClientKeyExchange, we can reconstruct the session key (master secret) and decrypt the rest of the session
- We need to protect past communications against the loss of a long-term secret

Ephemeral Diffie-Hellman is used by TLS 1.3+ to ensure perfect forward secrecy

- TLS 1.3 lets the client identify their preferred p and g from a small registry of "good" values

Gillmor, D. (2016) *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security*. RFC7919. Available at: https://tools.ietf.org/html/rfc7919

Latency

The handshake from TLS 1.2 and earlier requires two round trips before secure communications can begin

- Minimum of 4 * 0.067 = 0.27s

This is on top of the 3-way TCP handshake (minimum 0.2s) and before any additional overheads from HTTP

TLS 1.3 reduces the handshake to a single round trip

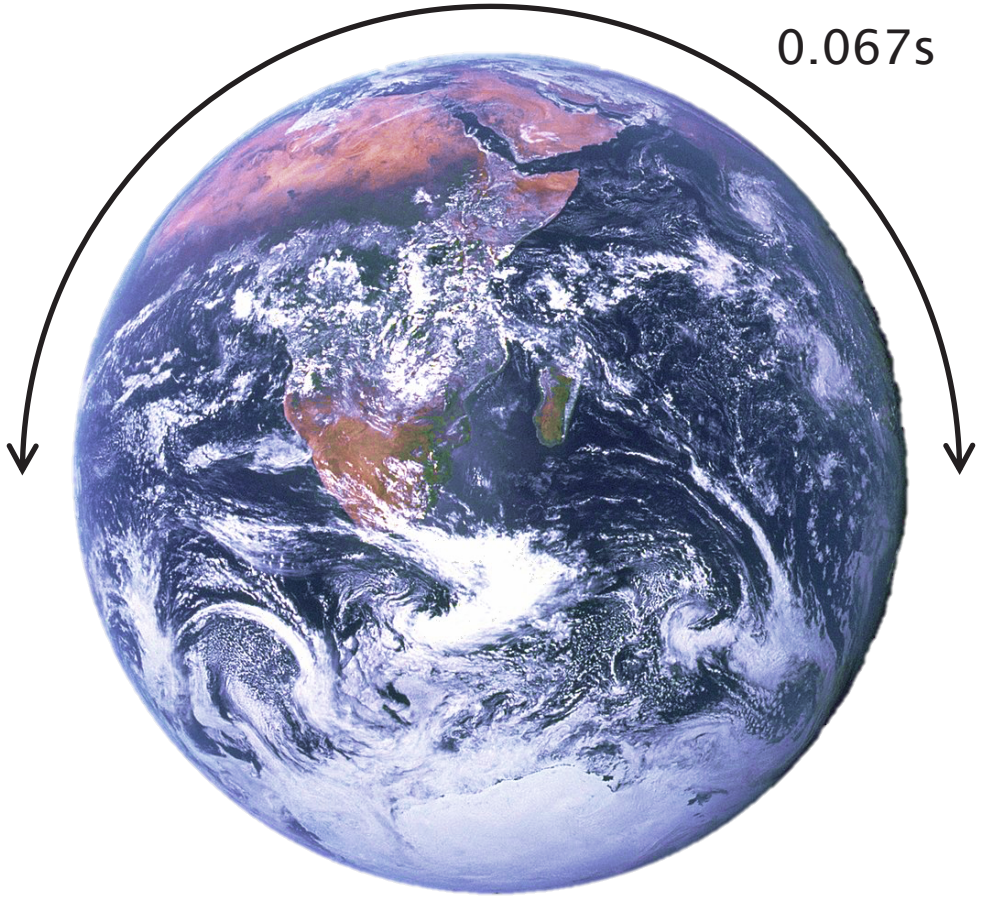
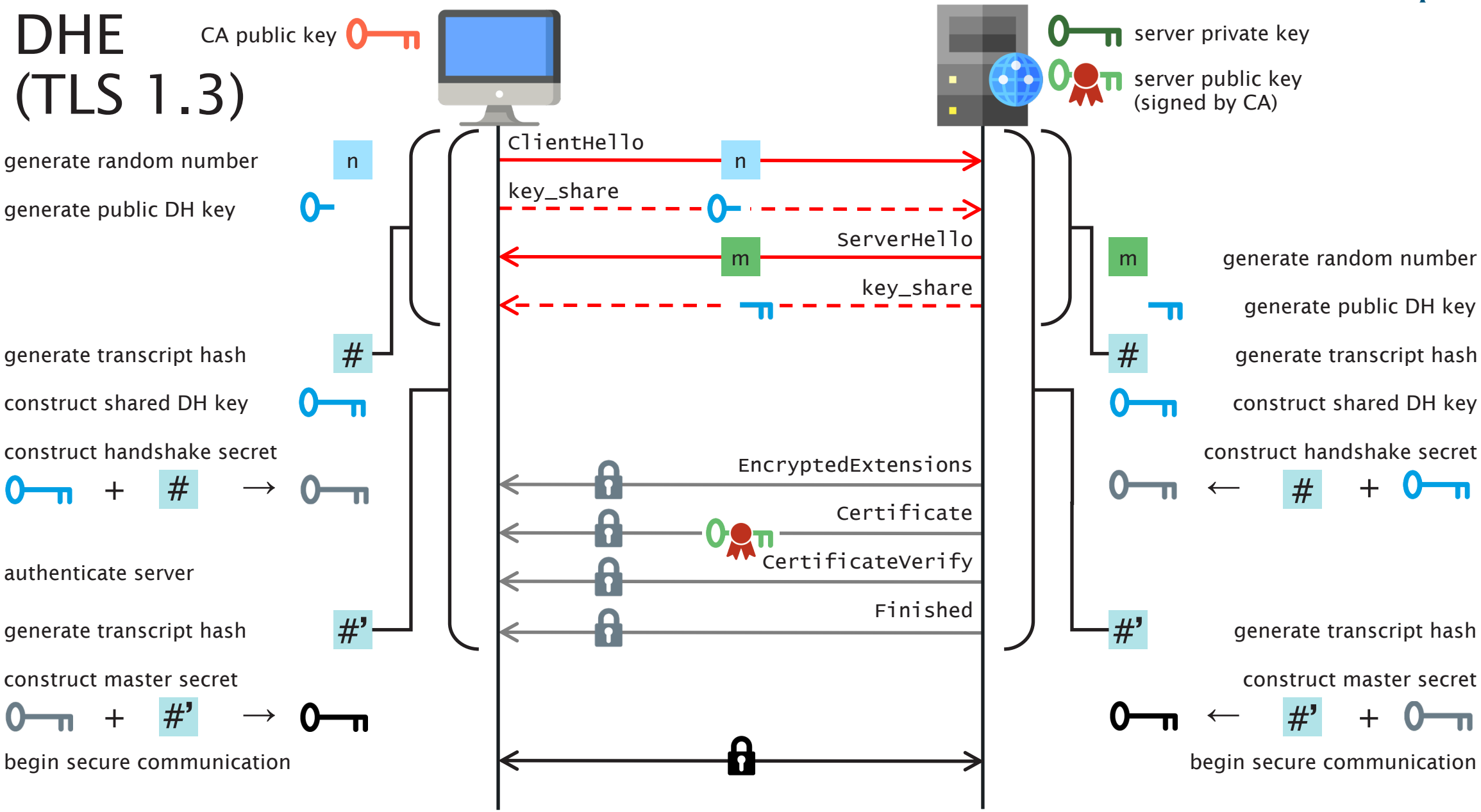


Image: NASA

DHE (TLS 1.3)


CA public key
server private key
server public key (signed by CA)
generate random number
n
ClientHello
n
generate public DH key
key_share
ServerHello
m
m
generate random number
key_share
generate public DH key
generate transcript hash
#
#
generate transcript hash
construct shared DH key
construct shared DH key
construct handshake secret
EncryptedExtensions
construct handshake secret
+
#
→
←
#
+
Certificate
CertificateVerify
authenticate server
Finished
generate transcript hash
#'
#'
generate transcript hash
construct master secret
construct master secret
+
#'
→
←
#'
+
begin secure communication
begin secure communication

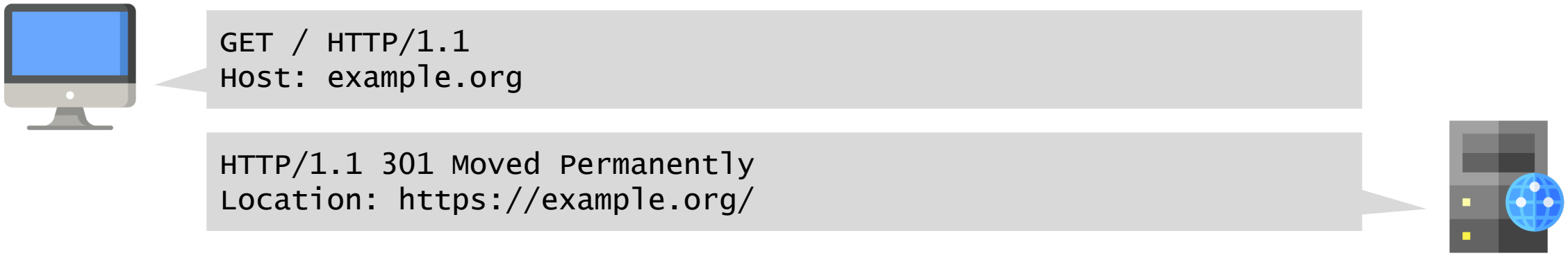
HTTPS

HTTP over a TLS connection

- Standard port is 443 (as opposed to port 80 for HTTP)

Now common for HTTP URIs to redirect to an equivalent HTTPS URI

- `301 Moved Permanently` or `308 Permanent Redirect`



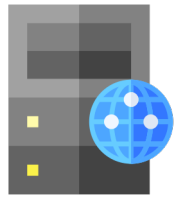
HTTP Strict Transport Security

`301 Moved Permanently` requires a redirect for every URI on a website

- `Strict-Transport-Security:` header allows a website to declare that it is only accessible via HTTPS
- Avoids future redirections for all URIs on that website (and possibly also sub-domains)

```
GET / HTTP/1.1
Host: example.org
```

```
HTTP/1.1 301 Moved Permanently
Location: https://example.org/
Strict-Transport-Security: max-age=31536000; includeSubDomains
```



Hodges, J. et al (2012) *HTTP Strict Transport Security*. RFC6797. Available at: https://tools.ietf.org/html/rfc6797

Further Reading

Dierks, T. and Rescorla, E. (2008) *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC5246.

https://tools.ietf.org/html/rfc5246

Rescorla, E. (2018) *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC8446.

https://tools.ietf.org/html/rfc8446

Diffie, W. and Hellman, M. (1976) *New Directions in Cryptography*. IEEE Transactions on Information Theory. 22(6). pp. 644-654.

Merkle, R.C. (1978) *Secure communications over insecure channels*. Communications of the ACM. 21(4). pp.294-299.

Hodges, J. et al (2012) *HTTP Strict Transport Security*. RFC6797.

https://tools.ietf.org/html/rfc6797

UNIVERSITY OF Southampton

Next Lecture: Proxies and Caching