COMP30023 – Computer Systems
2022– Semester 1 - Week 5 – Lecture 1

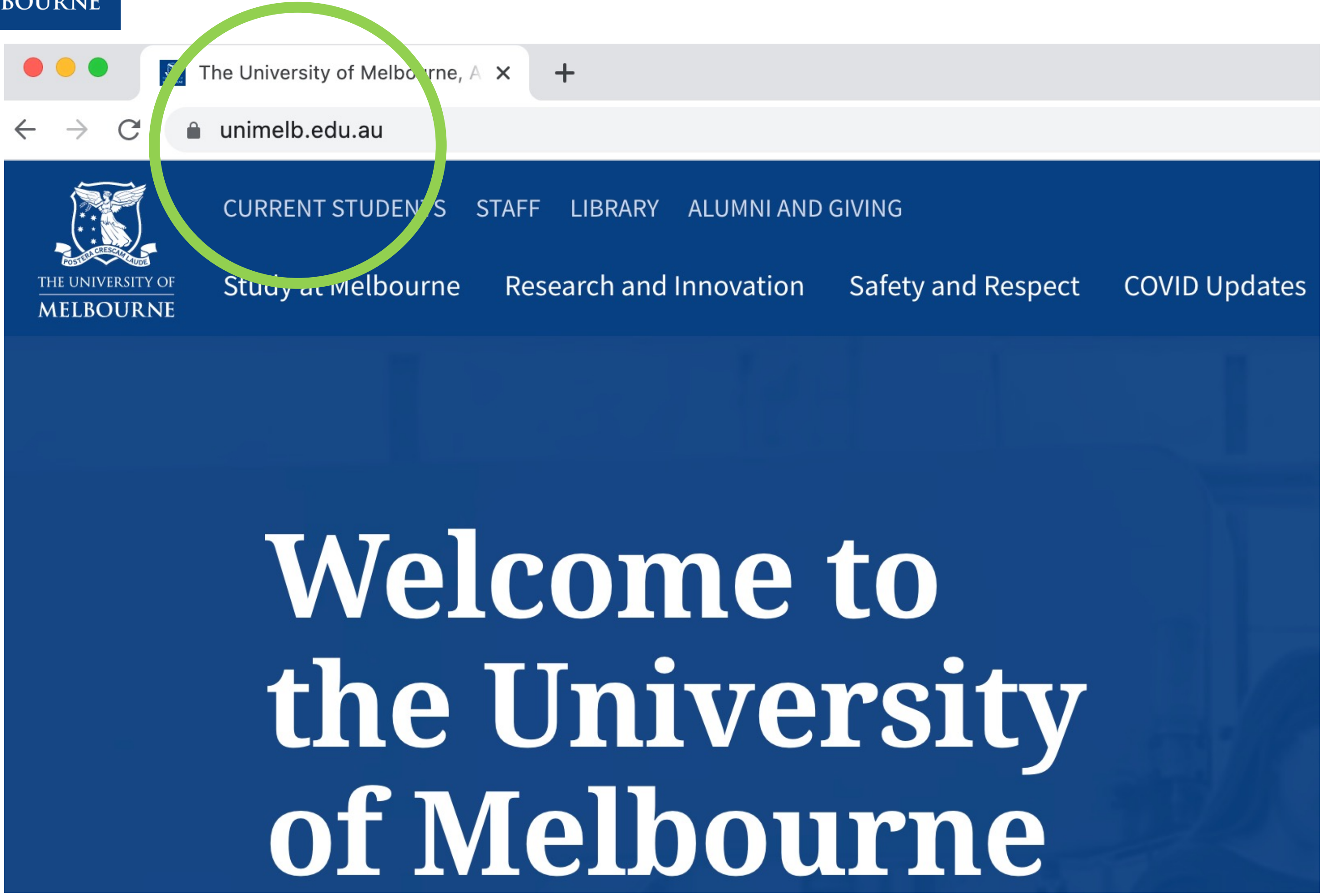
# Certificates, Public Key Infrastructure, TLS

Olya Ohrimenko

## Mid-Semester Test (MST)

- Week 6 Thursday: during lecture time
- Length: 40 mins
- 10% of the total mark
- Canvas quiz
  - 10 multiple choice questions
  - 5 short answer questions
- More details/instructions on LMS

# What does the lock mean?



The University of Melbourne, A

unimelb.edu.au

CURRENT STUDENTS STAFF LIBRARY ALUMNI AND GIVING

Study at Melbourne Research and Innovation Safety and Respect COVID Updates

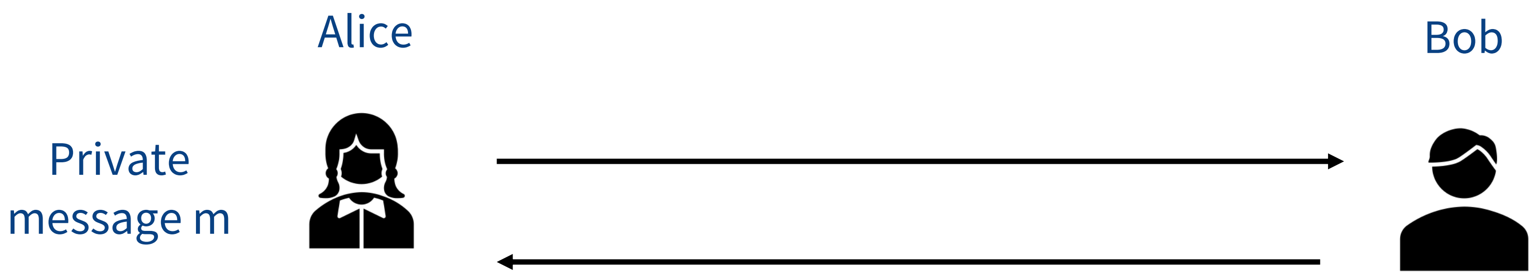
Welcome to the University of Melbourne

## Recap: Secure Communication

Confidentiality – Authentication – Integrity

Objective is to provide **secure private** communication between two end-points, with **integrity checks** to ensure data does not change in transit, and **authentication** to establish identities of one or both of the end-points.

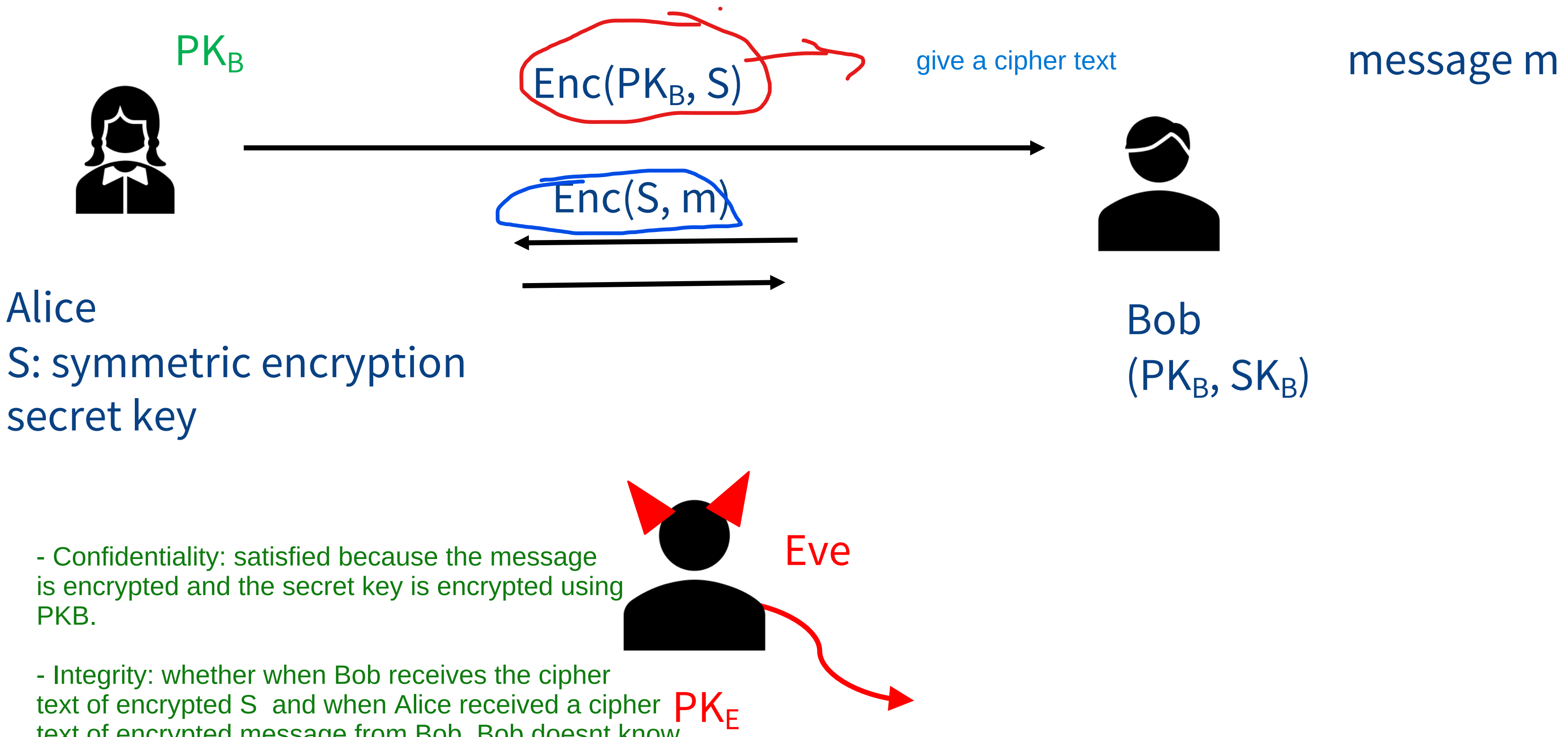
# Goal: Secure Communication



2 problems:
- How does Alice know ciphertext has not been modified?
- How does Alice know $PK_B$ is Bob's public key?

## Why

- Bob created a pair of public key and private key (asymmetric encryption).
- Bob shared the Public Key with Alice.
- Alice encrypted S with the public key Bob shared. (S : symmetric encryption key)
- Only Bob is able to decrypt the secret key because he had a SKB (secret key B).

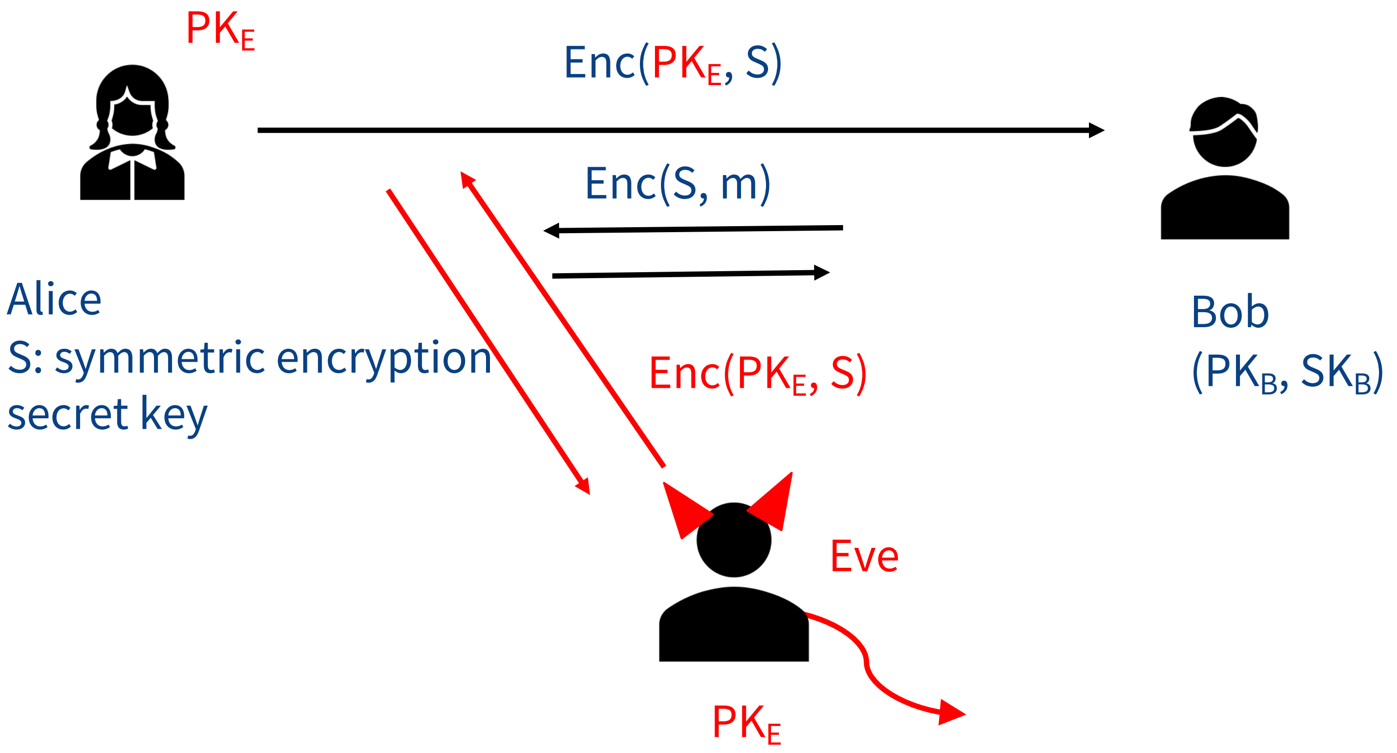


- Confidentiality: satisfied because the message is encrypted and the secret key is encrypted using PKB.

- Integrity: whether when Bob receives the cipher text of encrypted S and when Alice received a cipher text of encrypted message from Bob, Bob doesnt know that S is tampered, Alice doesnt know the message is tampered ?

????

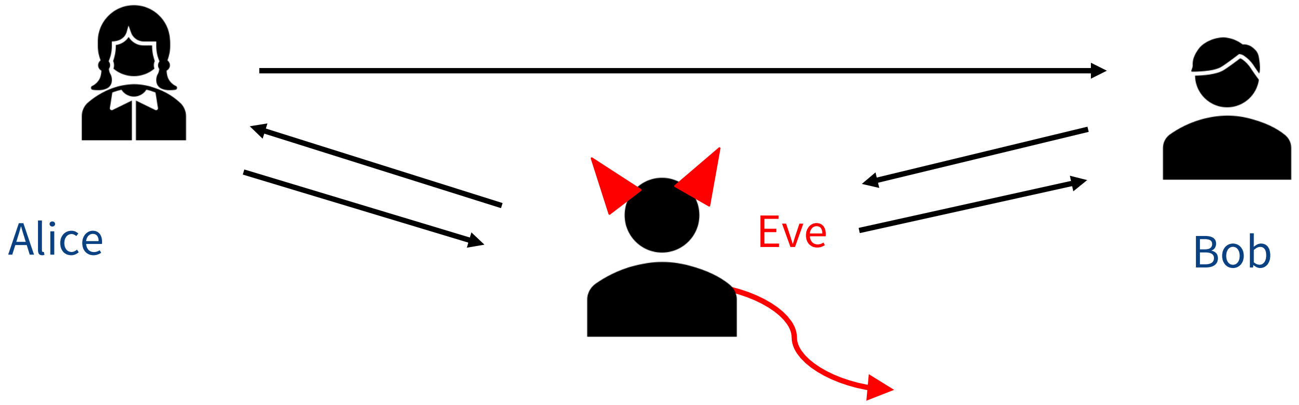
# Why



$PK_E$
Enc($PK_E$, S)
Enc(S, m)
Alice
S: symmetric encryption secret key
Enc($PK_E$, S)
Bob
($PK_B$, $SK_B$)
Eve
$PK_E$

# Human-in-the-middle Attack

## aka Man-in-the-Middle (MITM) attack



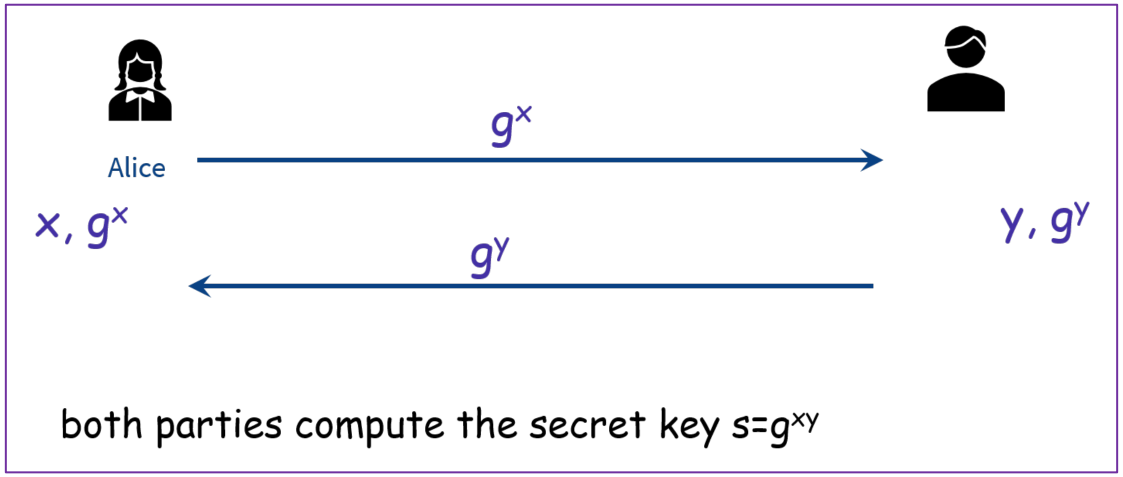
Attack: network traffic between Alice and Bob goes via Eve who:

- Trick Alice into thinking it is communicating with Bob and vice versa by modifying the messages (impersonation)
- Silently snoops on the messages sent between Alice and Bob (eavesdropping)

# Diffie-Hellman Key Exchange and MITM

- Is DH secure against impersonation attack?
- Is DH secure against eavesdropping attack?

g,p publicly known



## Purpose of certificates

- Securely associate identities with cryptographic public keys
  - Name, domain, organisation, etc.
  - e.g., for SSL/TLS and HTTPS for authenticated and encrypted web browsing, code signing, client authentications
- The certificate itself is signed *(Certificate Signature)* – often by a third party

## Certificate Example

- certificate = ($PK_{Alice}$, Alice, …)
- Signing: s = Sign ($SK_{Issuer}$, certificate), certificate contains d and other details such as issuer, algorithms, validity date
- Certificate: Issuer, Signature s, certificate = (Issuer, $PK_{Alice}$, Alice, …)
- Verification: Verify ($PK_{Issuer}$, s, certificate)

## Certificate format

X.509 the most common standard format

Consists of: Version Number, Serial Number, Signature Algorithm ID, Issuer Name, Validity Period (not before and not after), Subject name, Subject Public Key Info (Public Key Algorithm and Subject Public Key) , Issuer Unique Identifier (optional) Subject Unique Identifier (optional), Extensions (optional), Certificate Signing Algorithm, Certificate Signature, subject alternative names

## Certificate Hierarchies

- The signer is vouching for the identity behind the public/private key pair
- Certificates can be chained: A signs B’s Certificate, B signs C’s certificate and C runs a website or issues digital signatures
- Trust flows down the hierarchy, if you trust A, then you trust all certificates A signs, and in turn, all certificates signed with certificates signed by A, i.e. trusting A implies trusting C
- Limiting purpose is critical (BasicConstraints -  id-ce 19)

# Chained Signing & Verification: Example

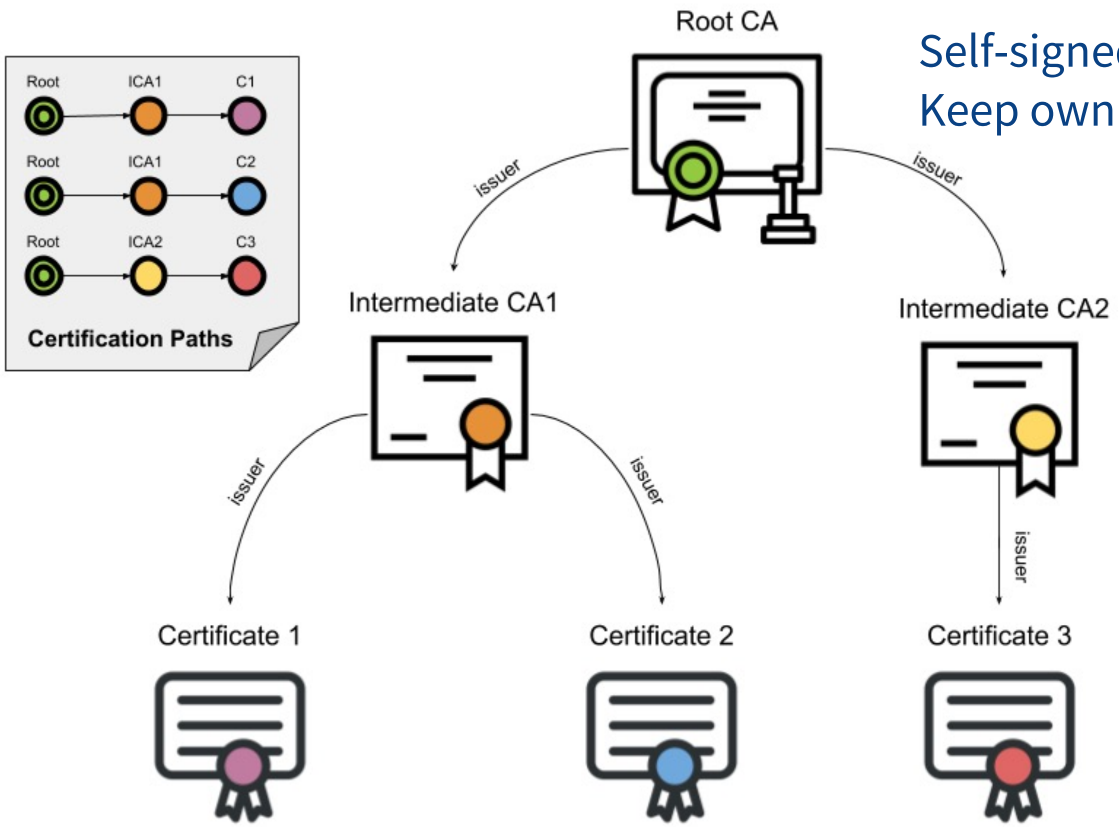
- $PK_A$, $SK_A$, $PK_B$, $SK_B$, verification/signing keys of a digital signature
- A signs B's certificate, B signs C's certificate
  - A: $s_B$ = Sign ($SK_A$, $\text{certificate}_B$), $\text{certificate}_B$ contains $PK_B$
  - B: $s_C$ = Sign ($SK_B$, $\text{certificate}_C$), $\text{certificate}_C$ contains Charlie's domain address www.foobar

- Alice wants to verify www.foobar is Charlie's using $\text{certificate}_C$ :
  - Verify($PK_B$, $s_C$ , $\text{certificate}_C$)
  - Verify($PK_A$, $s_B$ , $\text{certificate}_B$)

## Trust anchors and certificate authorities

- Trust anchors are entities that are explicitly trusted
  - Most commonly found as root certificates
- They are the points from which all other trust is derived
- Certificate Authorities (CA) are the most common trust anchors
  - Sign certificates for others
  - Root certificates are shipped, pre-loaded, with your Operating System/Browser
  - Root certificate is self-signed
  - Sub-CA – intermediate CA that is not a root, but has been signed by a root
  - Cross-signing – Sub-CA signed by multiple root CAs (compatibility/robustness)

# Certificate Hierarchy (Image)



Self-signed certificate
Keep own secret key safe

https://www.ssl.com/article/browsers-and-certificate-validation/

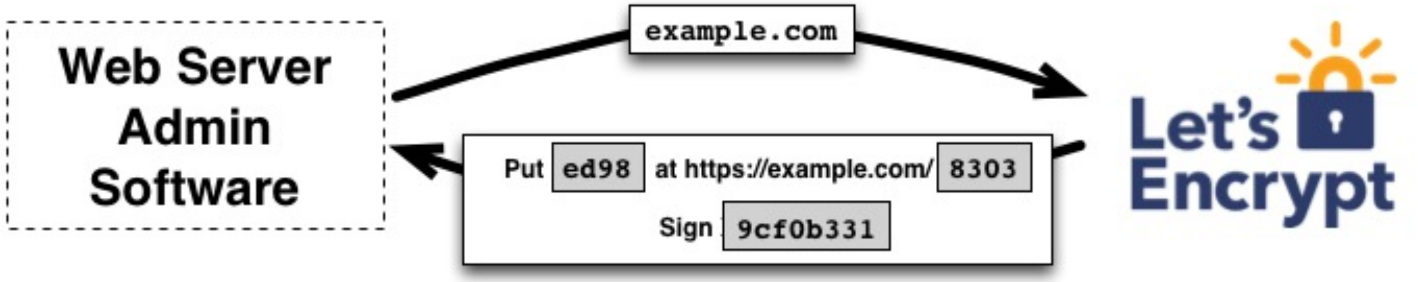
## Trusting root certificates

- Machine will contain 50+ root certificates
  - issued by governments, companies
- Should we really trust all those organisations?

## Certificate Issuance

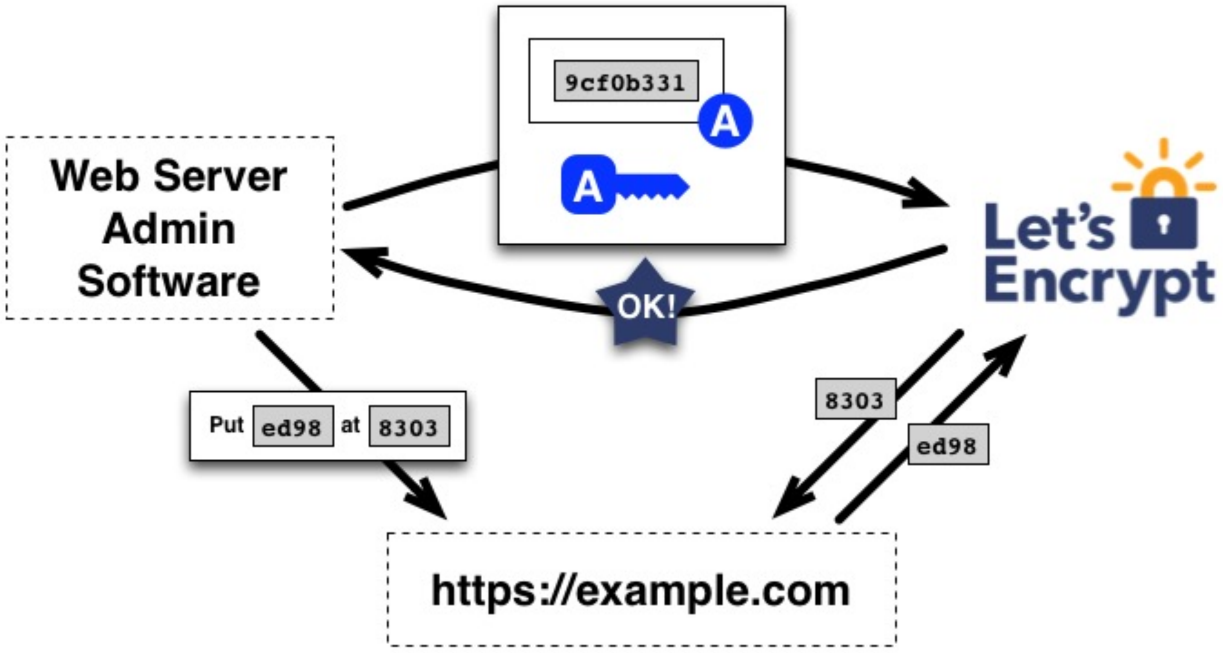
- Domain Validation (DV)
  - Most common
  - Ties a certificate to a domain and checks the requester has some control over the domain
  - Validation via email/DNS/URL – possible weakness
- Organisation Validation (OV)
  - Ties a certificate to a domain and a legal entity (e.g., an operating business)
- Extended Validation (EV)
  - Establishes legal entity, jurisdiction, and presence of authorised officer (e.g., physical address, phone #)
  - Offline process + expensive

# Domain Validation Example


Web Server Admin Software
example.com
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt

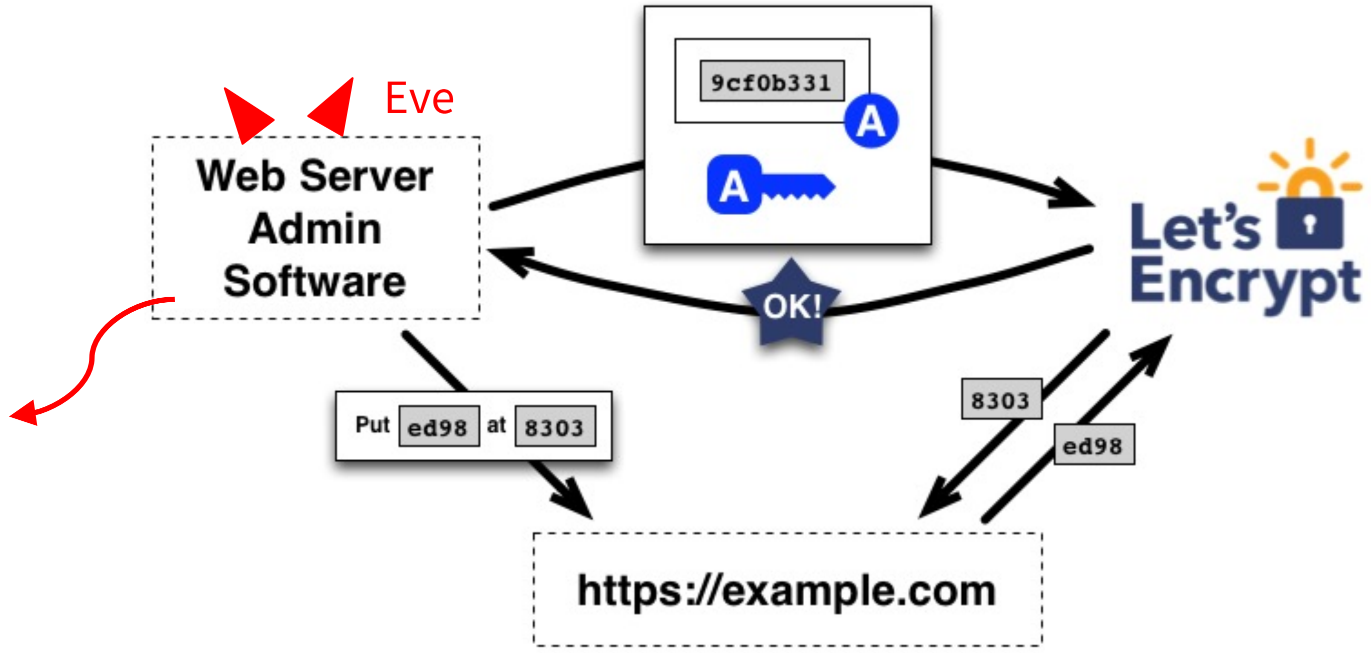
https://letsencrypt.org/how-it-works/

# Domain Validation Example



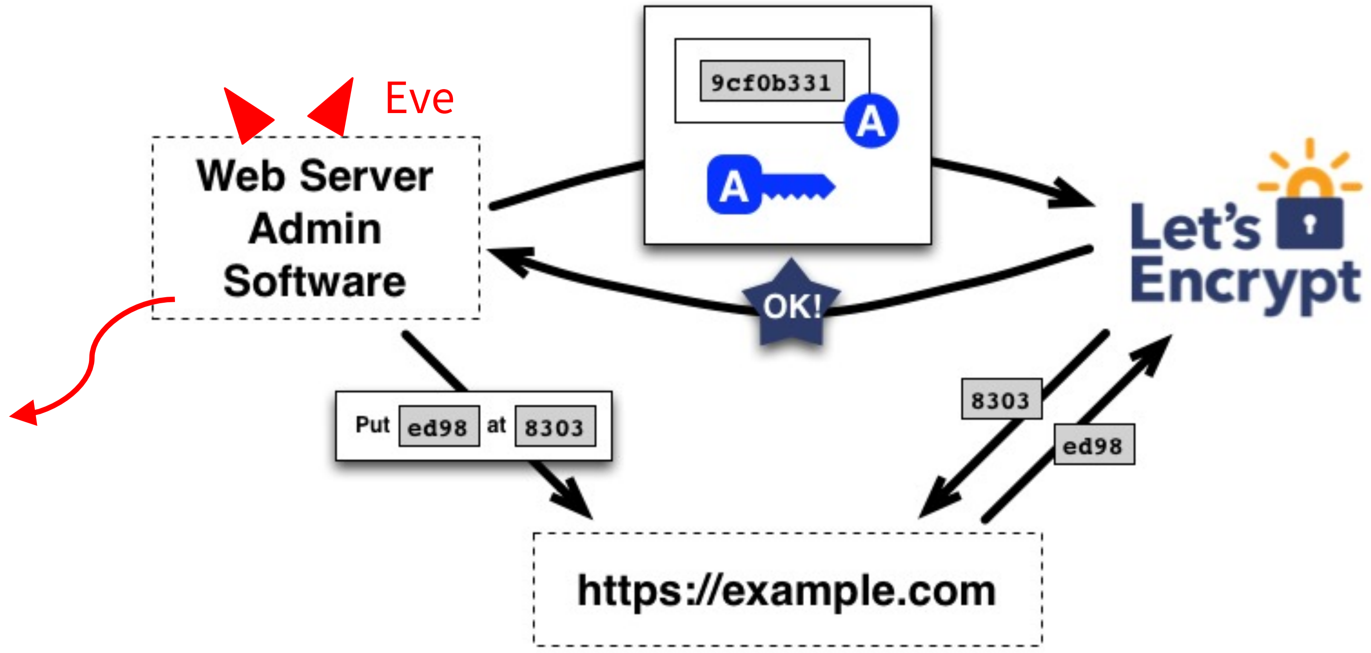
https://letsencrypt.org/how-it-works/

# Domain Validation Example



https://letsencrypt.org/how-it-works/

## Domain Validation Example



Question: why verify ownership of PK **AND** domain address?

https://letsencrypt.org/how-it-works/

## Certificate Issuance - problems

- DV certificates do not establish a link between the domain and a real world entity
  - LetsEncrypt has issued 14,000 Certificates containing the word "paypal"[1]
  - WoSign incorrectly issued certificates for github[2]
- Even EV certificates are not immune
  - Symmantec issued an EV certificate for Google.com[3]

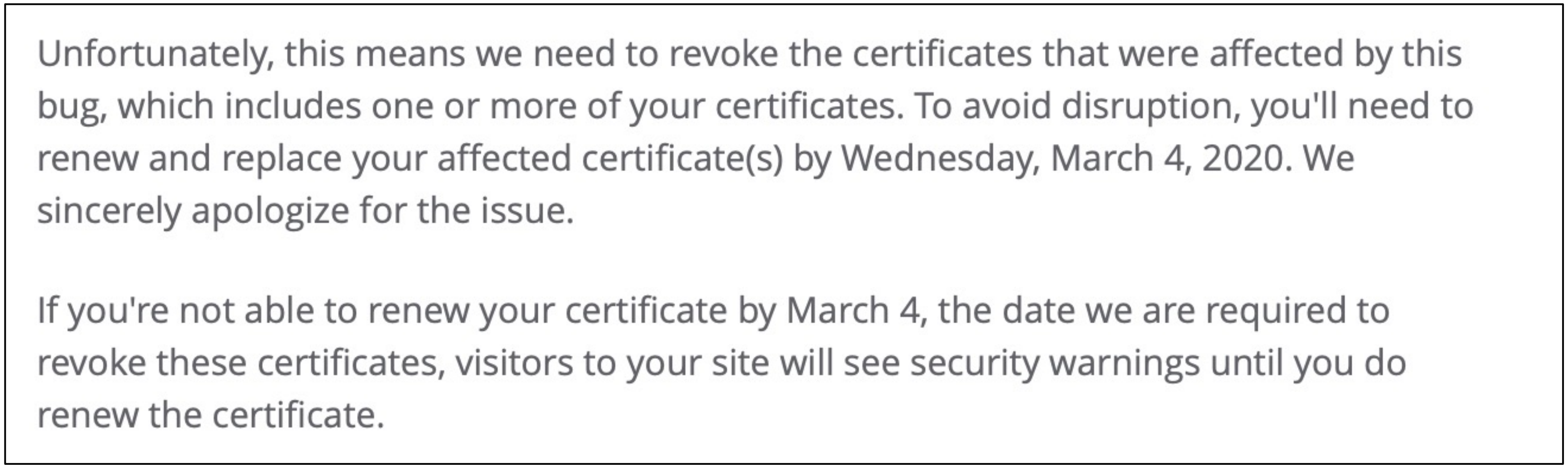
[1] https://www.thesslstore.com/blog/lets-encrypt-phishing/
[2] https://www.schrauger.com/the-story-of-how-wosign-gave-me-an-ssl-certificate-for-github-com
[3] http://www.pcworld.com/article/2999146/encryption/google-threatens-action-against-symantec-issued-certificates-following-botched-investigation.html

## Certificate Revocation

- Revocation occurs when a certificate is:
  - Mistakenly issued (Let’s Encrypt, February 2020)
  - Private key is compromised

## Certificate Revocation

- Revocation occurs when a certificate is:
  - Mistakenly issued (Let’s Encrypt, February 2020)
  - Private key is compromised



Unfortunately, this means we need to revoke the certificates that were affected by this bug, which includes one or more of your certificates. To avoid disruption, you'll need to renew and replace your affected certificate(s) by Wednesday, March 4, 2020. We sincerely apologize for the issue.

If you're not able to renew your certificate by March 4, the date we are required to revoke these certificates, visitors to your site will see security warnings until you do renew the certificate.

https://arstechnica.com/information-technology/2020/03/lets-encrypt-revoking-https-certs-due-to-certificate-authority-bug/

## Certificate Revocation

- Revocation occurs when a certificate is:
  - Mistakenly issued (Let’s Encrypt, February 2020)
  - Private key is compromised
- Performed via Certificate Revocation Lists and OCSP (Online Certificate Status Protocol) [CRLite, Mozilla]
  - Performance vs privacy
- When root certificates are revoked all certificates below that become untrusted
  - Example: DigiNotar, CNNIC, WoSign
  - Hence cross-signing root certificates
- Only useful if aware of incorrectly issued certificate

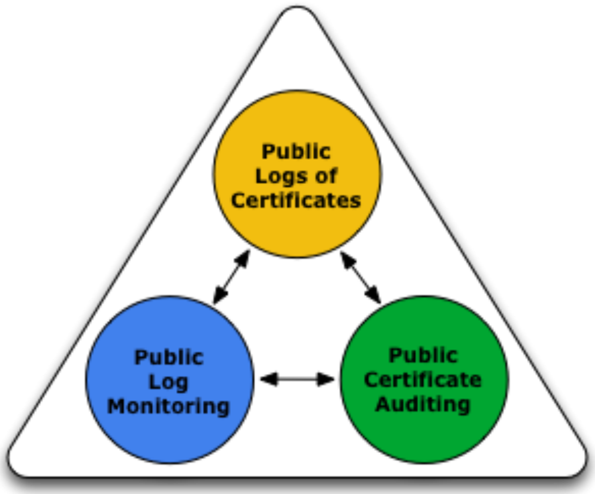
## Certificate Transparency

- Open framework for monitoring and auditing certificates
- Intended to provide a way of monitoring certificates that have been issued
- Detect:
  - misissued certificates
  - maliciously acquired certificates
  - rogue CAs

## Certificate Transparency (CT)

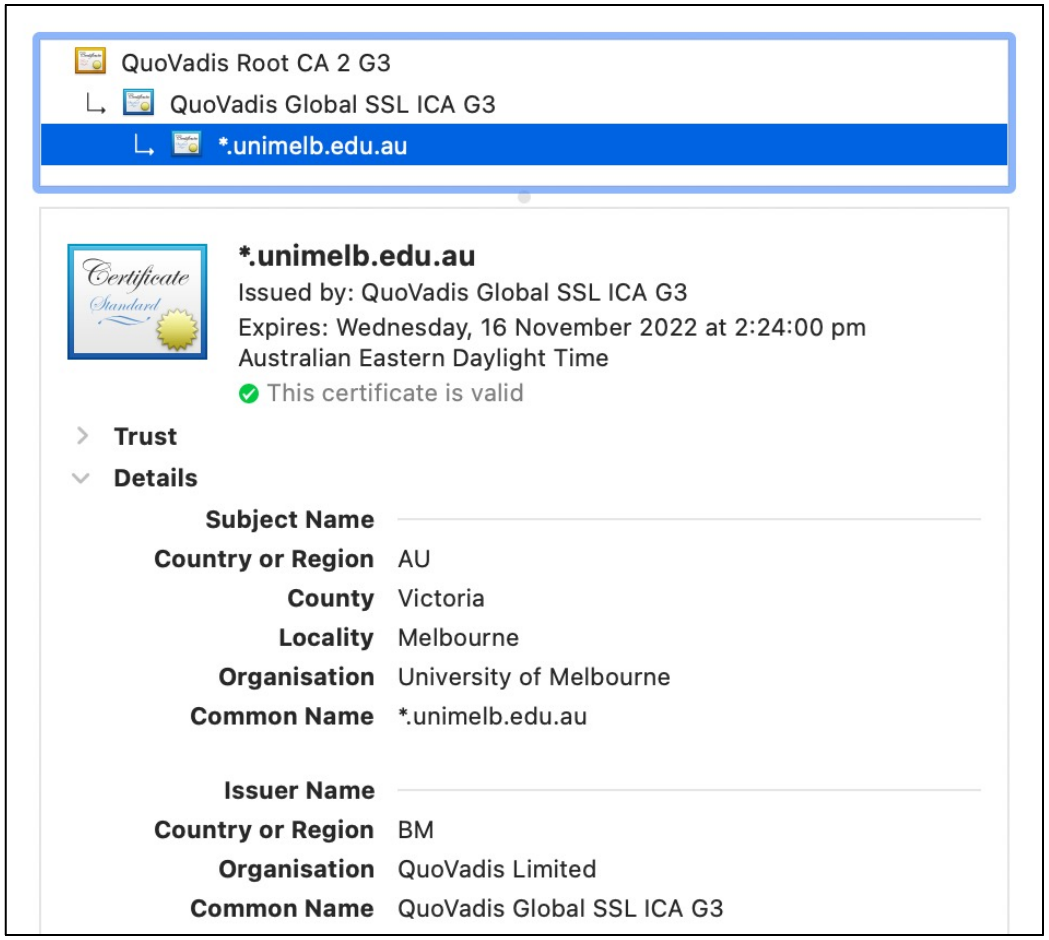
- Uses a cryptographic append-only log to record the issuance of certificates
  - Merkle Trees: consistency and membership verification
- Server should send SCT (signed certificate timestamp)
  - SCT is issued by CT log to prove inclusion
- Browser may choose to decline certificates not on CT log
- SCT is an extension in X.509

As of Mar 24, 2022, there have been 17,910,600,871 entries made to the set of Certificate Transparency logs that Google monitors.



https://www.certificate-transparency.org/what-is-ct

# Certificates and CAs examples



QuoVadis Root CA 2 G3
↳ QuoVadis Global SSL ICA G3
↳ *.unimelb.edu.au

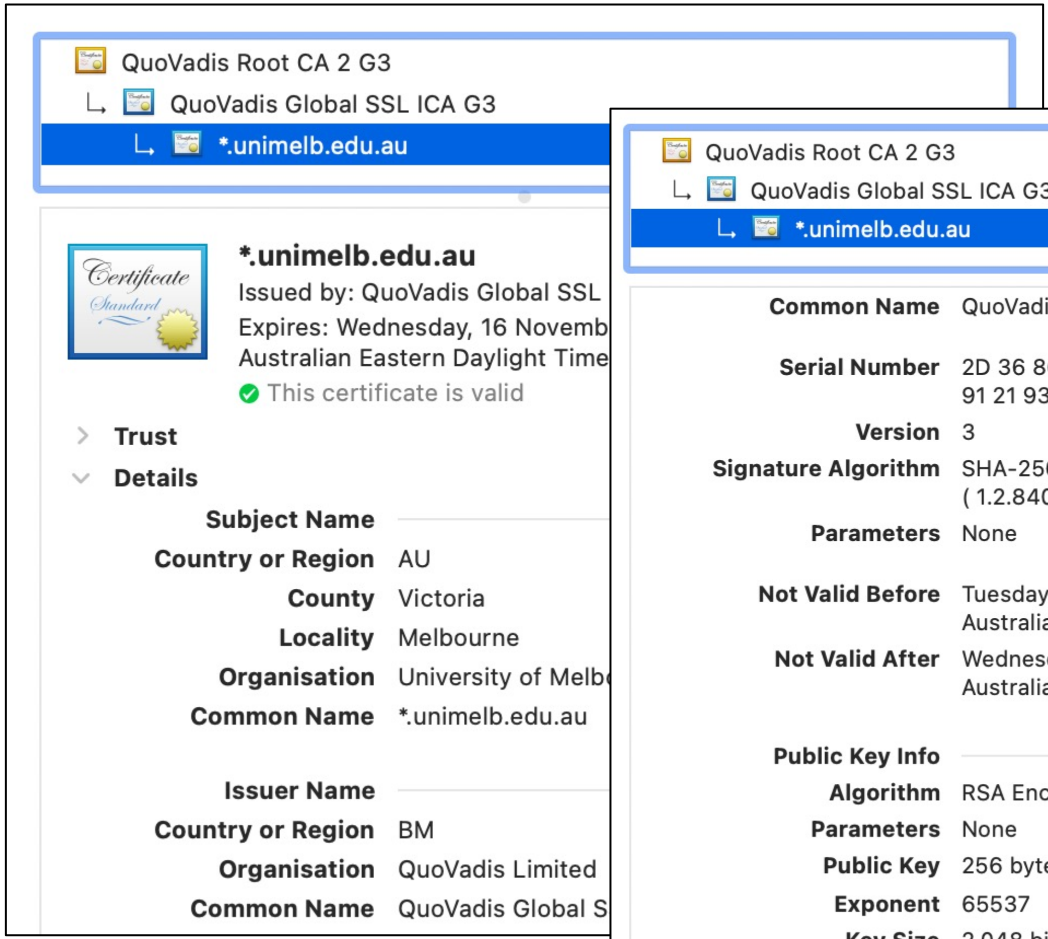
***.unimelb.edu.au**
Issued by: QuoVadis Global SSL ICA G3
Expires: Wednesday, 16 November 2022 at 2:24:00 pm Australian Eastern Daylight Time
This certificate is valid

**Trust**

**Details**

| | |
|---|---|
| **Subject Name** | |
| **Country or Region** | AU |
| **County** | Victoria |
| **Locality** | Melbourne |
| **Organisation** | University of Melbourne |
| **Common Name** | *.unimelb.edu.au |
| **Issuer Name** | |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Global SSL ICA G3 |

# Certificates and CAs examples

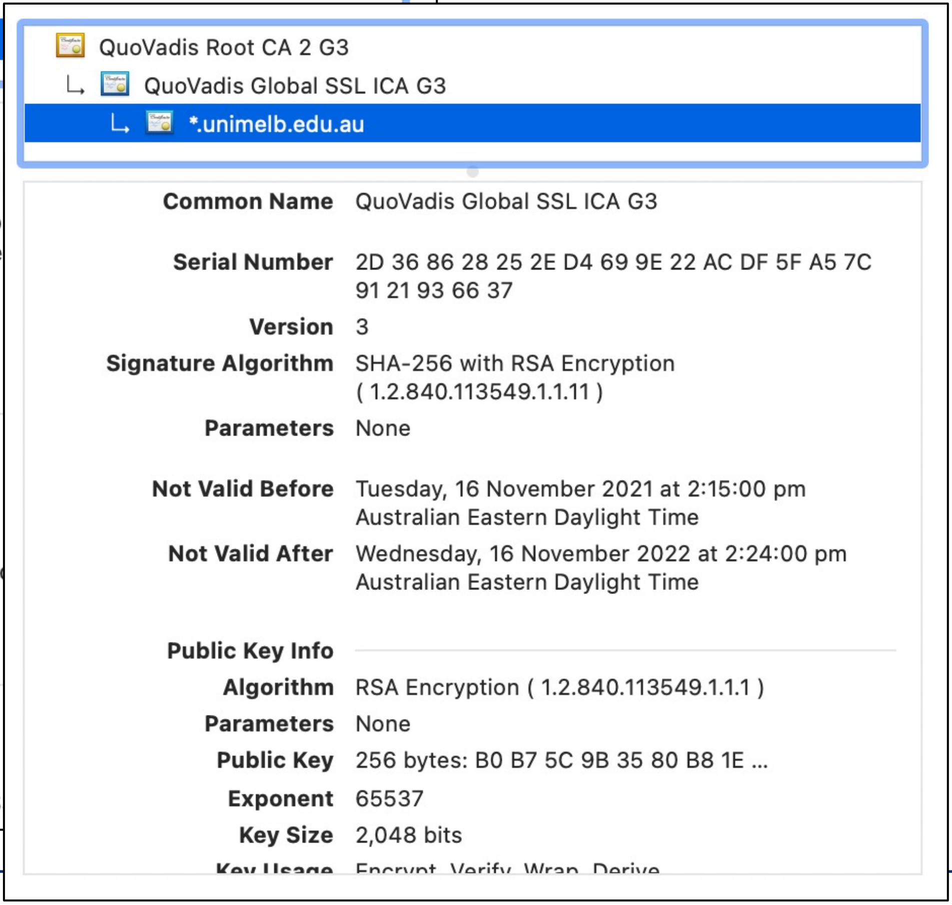


QuoVadis Root CA 2 G3
└ QuoVadis Global SSL ICA G3
  └ *.unimelb.edu.au

*.unimelb.edu.au
Issued by: QuoVadis Global SSL
Expires: Wednesday, 16 Novemb
Australian Eastern Daylight Time
✓ This certificate is valid

› Trust
⌄ Details

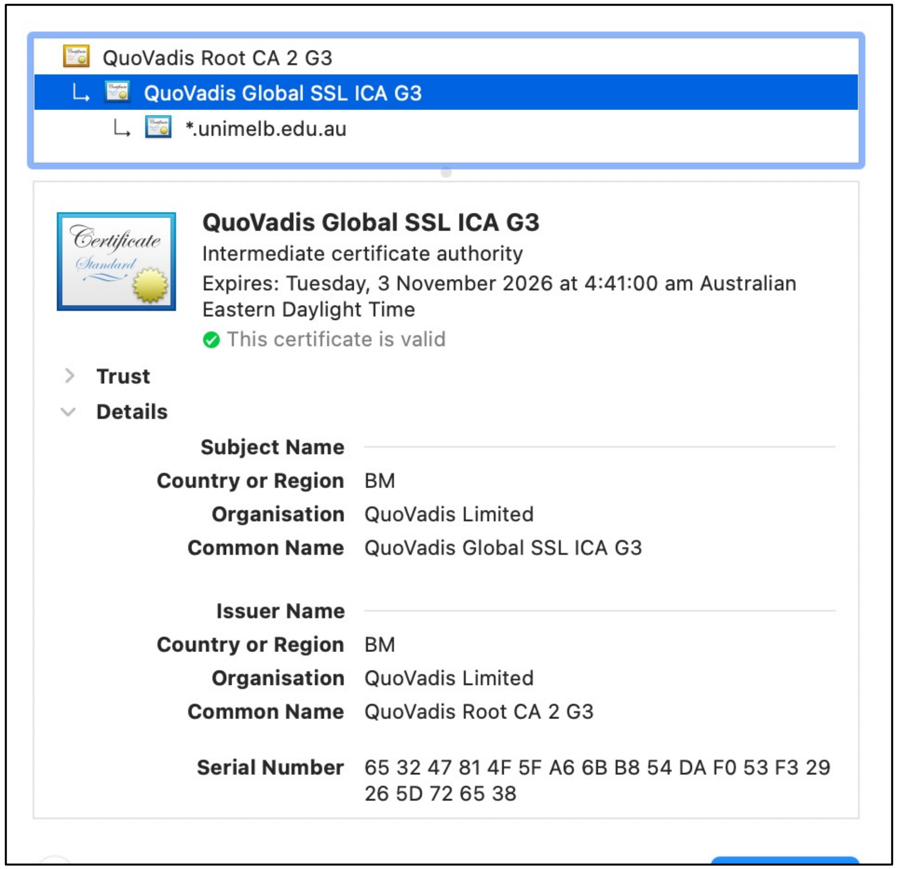
| | |
|---|---|
| **Subject Name** | |
| **Country or Region** | AU |
| **County** | Victoria |
| **Locality** | Melbourne |
| **Organisation** | University of Melb |
| **Common Name** | *.unimelb.edu.au |
| **Issuer Name** | |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Global S |



QuoVadis Root CA 2 G3
└ QuoVadis Global SSL ICA G3
  └ *.unimelb.edu.au

| | |
|---|---|
| **Common Name** | QuoVadis Global SSL ICA G3 |
| **Serial Number** | 2D 36 86 28 25 2E D4 69 9E 22 AC DF 5F A5 7C 91 21 93 66 37 |
| **Version** | 3 |
| **Signature Algorithm** | SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 ) |
| **Parameters** | None |
| **Not Valid Before** | Tuesday, 16 November 2021 at 2:15:00 pm Australian Eastern Daylight Time |
| **Not Valid After** | Wednesday, 16 November 2022 at 2:24:00 pm Australian Eastern Daylight Time |
| **Public Key Info** | |
| **Algorithm** | RSA Encryption ( 1.2.840.113549.1.1.1 ) |
| **Parameters** | None |
| **Public Key** | 256 bytes: B0 B7 5C 9B 35 80 B8 1E ... |
| **Exponent** | 65537 |
| **Key Size** | 2,048 bits |
| **Key Usage** | Encrypt, Verify, Wrap, Derive |

## Certificates and CAs examples



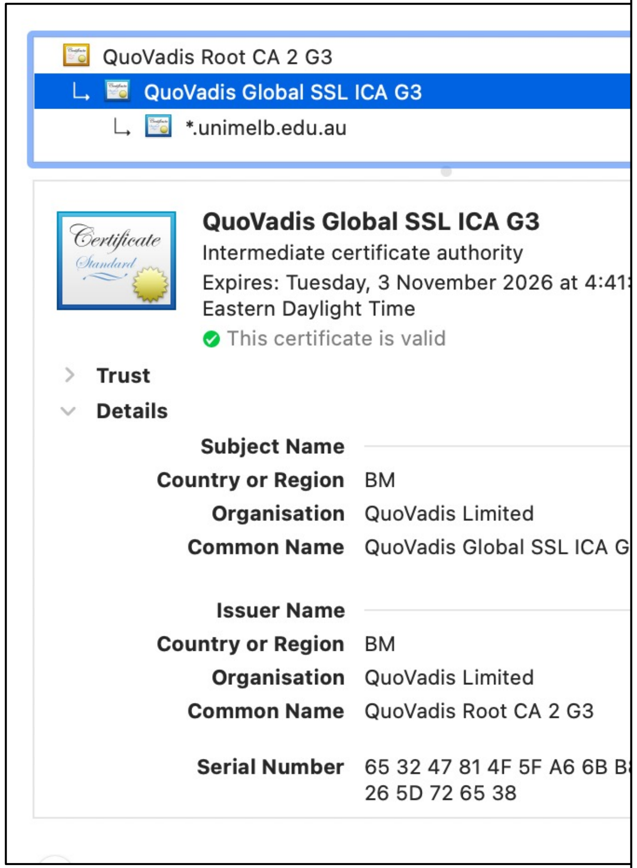
QuoVadis Root CA 2 G3
└ QuoVadis Global SSL ICA G3
  └ *.unimelb.edu.au

**QuoVadis Global SSL ICA G3**
Intermediate certificate authority
Expires: Tuesday, 3 November 2026 at 4:41:00 am Australian Eastern Daylight Time
✓ This certificate is valid

› **Trust**
⌄ **Details**

| | |
|---|---|
| **Subject Name** | |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Global SSL ICA G3 |
| **Issuer Name** | |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Root CA 2 G3 |
| **Serial Number** | 65 32 47 81 4F 5F A6 6B B8 54 DA F0 53 F3 29 26 5D 72 65 38 |

# Certificates and CAs examples

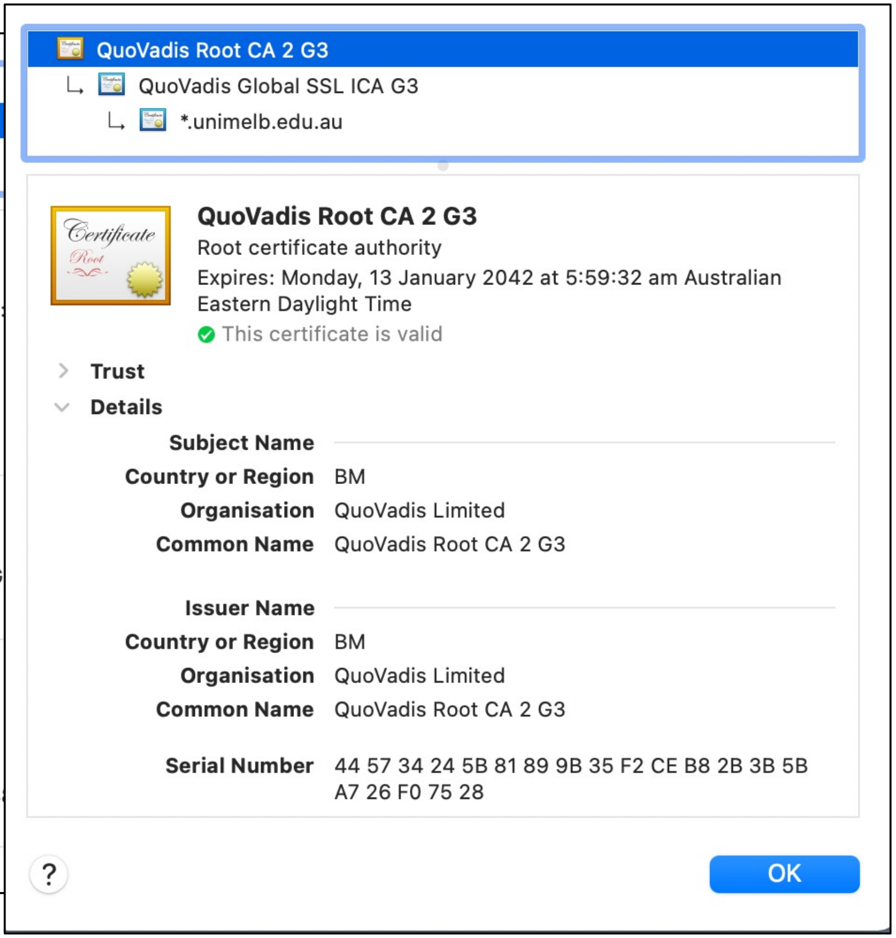


QuoVadis Root CA 2 G3
QuoVadis Global SSL ICA G3
*.unimelb.edu.au

**QuoVadis Global SSL ICA G3**
Intermediate certificate authority
Expires: Tuesday, 3 November 2026 at 4:41
Eastern Daylight Time
This certificate is valid

Trust
Details

|  |  |
|---|---|
| **Subject Name** |  |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Global SSL ICA G |
| **Issuer Name** |  |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Root CA 2 G3 |
| **Serial Number** | 65 32 47 81 4F 5F A6 6B B 26 5D 72 65 38 |



QuoVadis Root CA 2 G3
QuoVadis Global SSL ICA G3
*.unimelb.edu.au

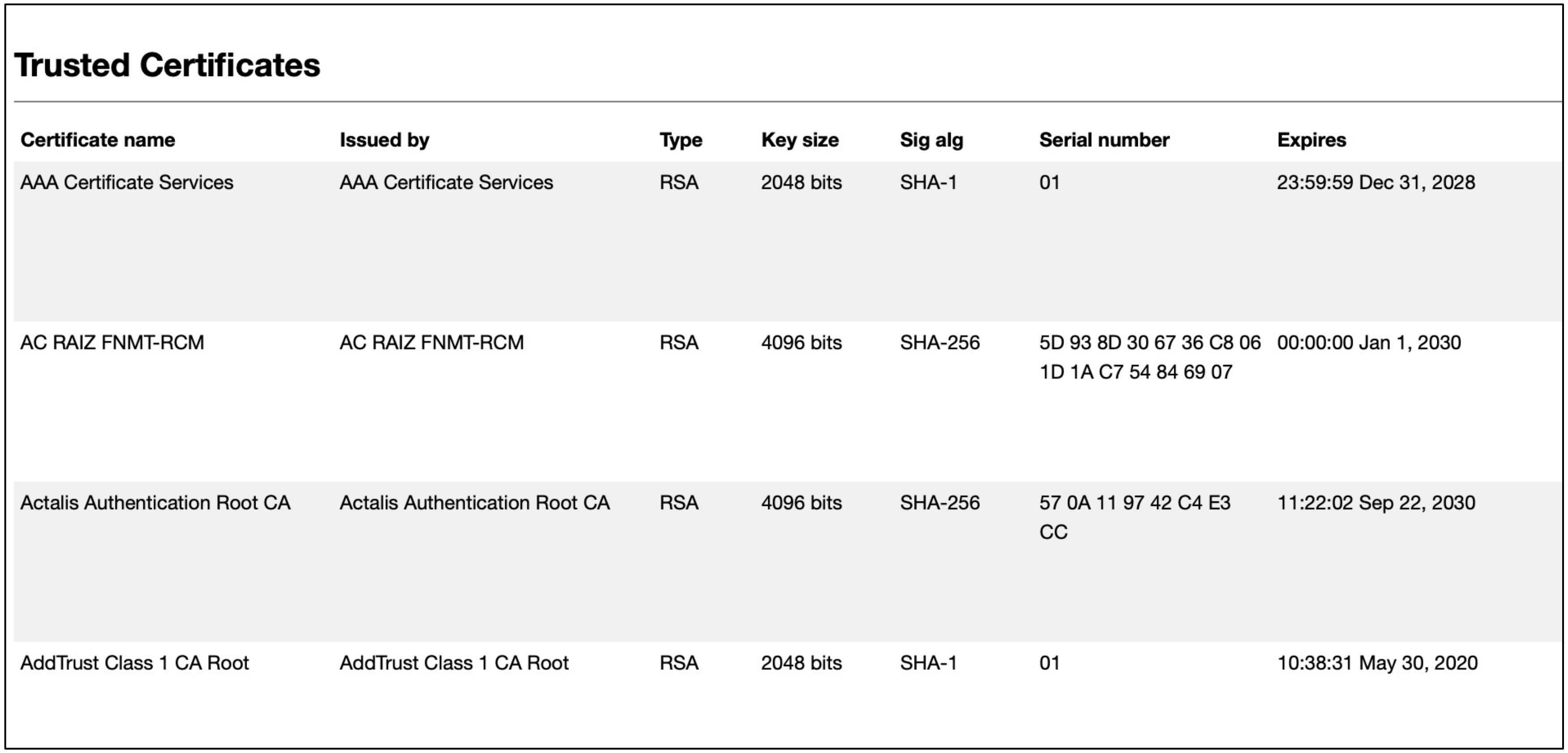
**QuoVadis Root CA 2 G3**
Root certificate authority
Expires: Monday, 13 January 2042 at 5:59:32 am Australian Eastern Daylight Time
This certificate is valid

Trust
Details

|  |  |
|---|---|
| **Subject Name** |  |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Root CA 2 G3 |
| **Issuer Name** |  |
| **Country or Region** | BM |
| **Organisation** | QuoVadis Limited |
| **Common Name** | QuoVadis Root CA 2 G3 |
| **Serial Number** | 44 57 34 24 5B 81 89 9B 35 F2 CE B8 2B 3B 5B A7 26 F0 75 28 |

## Certificates and CAs examples

https://support.apple.com/en-au/HT212140

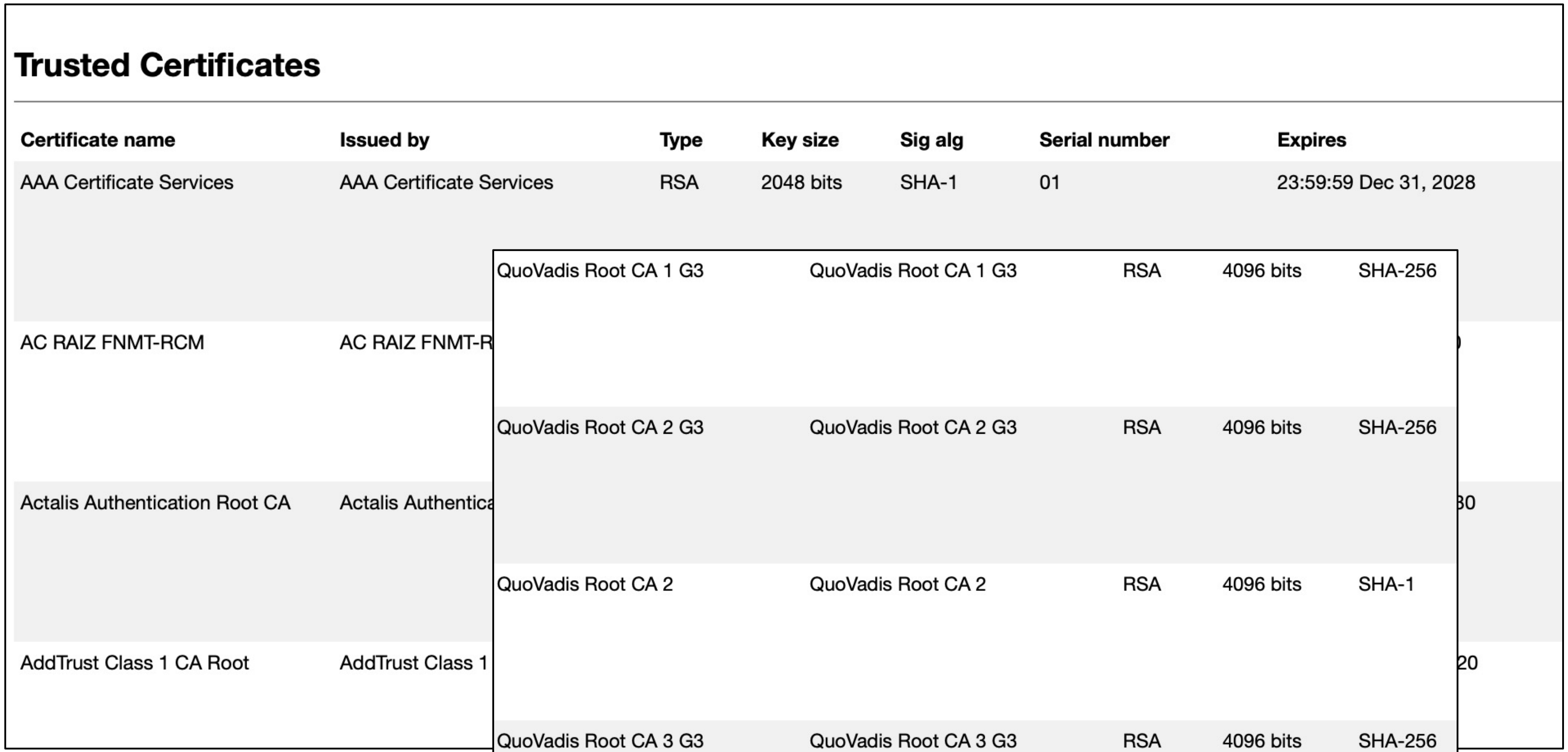


**Trusted Certificates**

| Certificate name | Issued by | Type | Key size | Sig alg | Serial number | Expires |
|---|---|---|---|---|---|---|
| AAA Certificate Services | AAA Certificate Services | RSA | 2048 bits | SHA-1 | 01 | 23:59:59 Dec 31, 2028 |
| AC RAIZ FNMT-RCM | AC RAIZ FNMT-RCM | RSA | 4096 bits | SHA-256 | 5D 93 8D 30 67 36 C8 06 1D 1A C7 54 84 69 07 | 00:00:00 Jan 1, 2030 |
| Actalis Authentication Root CA | Actalis Authentication Root CA | RSA | 4096 bits | SHA-256 | 57 0A 11 97 42 C4 E3 CC | 11:22:02 Sep 22, 2030 |
| AddTrust Class 1 CA Root | AddTrust Class 1 CA Root | RSA | 2048 bits | SHA-1 | 01 | 10:38:31 May 30, 2020 |

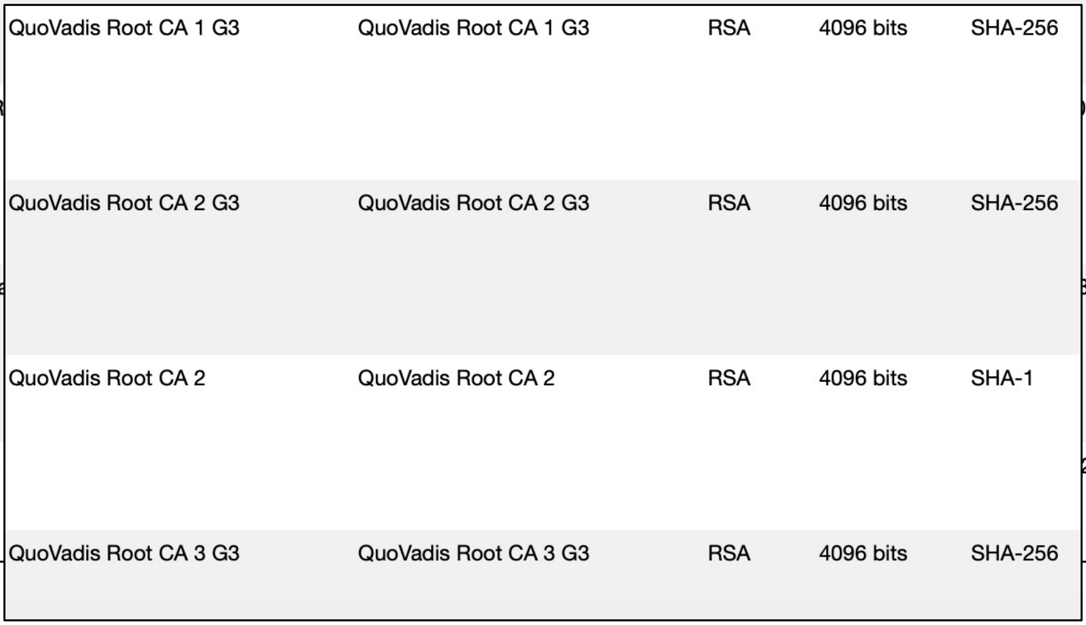
# Certificates and CAs examples

https://support.apple.com/en-au/HT212140



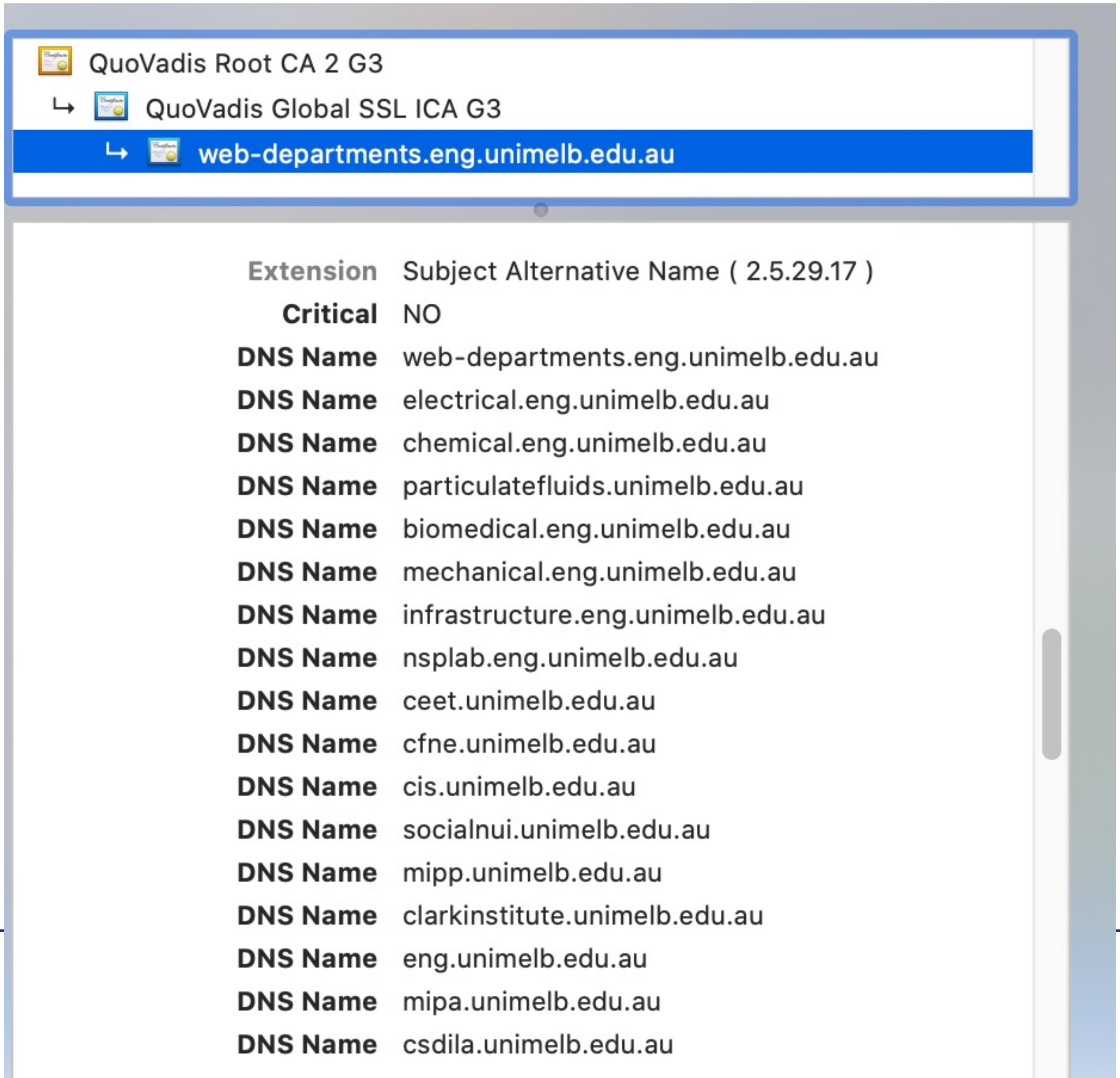
## Trusted Certificates

| Certificate name | Issued by | Type | Key size | Sig alg | Serial number | Expires |
|---|---|---|---|---|---|---|
| AAA Certificate Services | AAA Certificate Services | RSA | 2048 bits | SHA-1 | 01 | 23:59:59 Dec 31, 2028 |
| AC RAIZ FNMT-RCM | AC RAIZ FNMT-R | | | | | ) |
| Actalis Authentication Root CA | Actalis Authentica | | | | | 30 |
| AddTrust Class 1 CA Root | AddTrust Class 1 | | | | | 20 |



| | | | | |
|---|---|---|---|---|
| QuoVadis Root CA 1 G3 | QuoVadis Root CA 1 G3 | RSA | 4096 bits | SHA-256 |
| QuoVadis Root CA 2 G3 | QuoVadis Root CA 2 G3 | RSA | 4096 bits | SHA-256 |
| QuoVadis Root CA 2 | QuoVadis Root CA 2 | RSA | 4096 bits | SHA-1 |
| QuoVadis Root CA 3 G3 | QuoVadis Root CA 3 G3 | RSA | 4096 bits | SHA-256 |

# Subject alternative names



QuoVadis Root CA 2 G3
↳ QuoVadis Global SSL ICA G3
↳ web-departments.eng.unimelb.edu.au

| | |
|---|---|
| Extension | Subject Alternative Name ( 2.5.29.17 ) |
| Critical | NO |
| DNS Name | web-departments.eng.unimelb.edu.au |
| DNS Name | electrical.eng.unimelb.edu.au |
| DNS Name | chemical.eng.unimelb.edu.au |
| DNS Name | particulatefluids.unimelb.edu.au |
| DNS Name | biomedical.eng.unimelb.edu.au |
| DNS Name | mechanical.eng.unimelb.edu.au |
| DNS Name | infrastructure.eng.unimelb.edu.au |
| DNS Name | nsplab.eng.unimelb.edu.au |
| DNS Name | ceet.unimelb.edu.au |
| DNS Name | cfne.unimelb.edu.au |
| DNS Name | cis.unimelb.edu.au |
| DNS Name | socialnui.unimelb.edu.au |
| DNS Name | mipp.unimelb.edu.au |
| DNS Name | clarkinstitute.unimelb.edu.au |
| DNS Name | eng.unimelb.edu.au |
| DNS Name | mipa.unimelb.edu.au |
| DNS Name | csdila.unimelb.edu.au |

## Summary

- Public Key Infrastructure
- Certificate authorities
- Certificates

## SSL/TLS

Protocol for secure communication over Internet

Transport Layer Security protocol

- Supported by all popular web browsers , web servers, internet commerce sites
- HTTPS: implementation of TLS over HTTP

## Secure Communication

- This objective has been explored since the early days of networks, with proposals as early as 1993
- It is a good case study on why security is difficult, how things can go wrong, and how fixing those problems can be challenging
- Even with the best of intentions and extremely smart people, flaws in security persist

# History of the SSL/TLS Protocol

- SSL (Secure Socket Layer)
  - Secure TCP connection
  - Designed by Netscape in 1994
  - Version 1.0 was never released because of security flaws
  - Version 2.0 was the first to be made public
    - It too had security flaws
  - Version 3.0 (1996) – complete redesign, the basis on which TLS was designed

## TLS – Transport Layer Security

- TLS 1.0 (1999) – upgrade to SSL 3.0, with further improvements to security
- TLS 1.1 (2006) – further upgrades and defences against known attacks
- TLS 1.2 (2008) – updated primitives (moves from MD5-SHA1 to SHA-256 for pseudorandom number generation)
  - Added support for AES and various advanced modes of AES
  - 2011 further update to prevent downgrade attacks
    - Where a server can be tricked into downgrading to weaker SSL 2.0
- TLS 1.3 – significant differences, not backwards compatible, removing weaker crypto primitives, enforcing forward secrecy
  - 11 years and nearly 30 IETF drafts
  - dropped support for less/in secure cryptographic features (MD5)
  - performance (number of roundtrips 0 or 1)

## TLS Basics

- Handshake protocol:
  - Uses public-key cryptography to establish several shared secret keys between the client and the server
  - An initial negotiation between client and server that establishes the parameters of their subsequent interactions within TLS
- Record protocol:
  - Uses the secret keys established in the handshake protocol to protect confidentiality, integrity, and authenticity of data exchange between the client and the server

## Handshake

Wash hands for about 20 seconds with soap and hot water or use a sanitiser gel

https://www.bbc.com/news/health-51711227

## Handshake Protocol

Runs between a client and a server

For example, client = Web browser, server = website

Negotiate version of the protocol and the set of cryptographic algorithms to be used:

- Interoperability between different implementations

Authenticate server and client (optional)

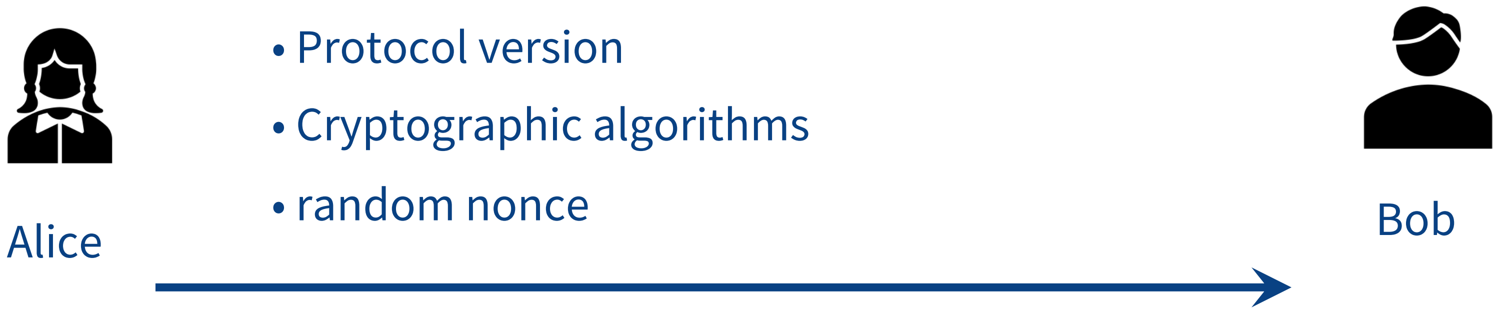
- Use digital certificates to learn each other’s public keys and verify each other’s identity
- Often only the server is authenticated

Use public keys to establish a shared secret

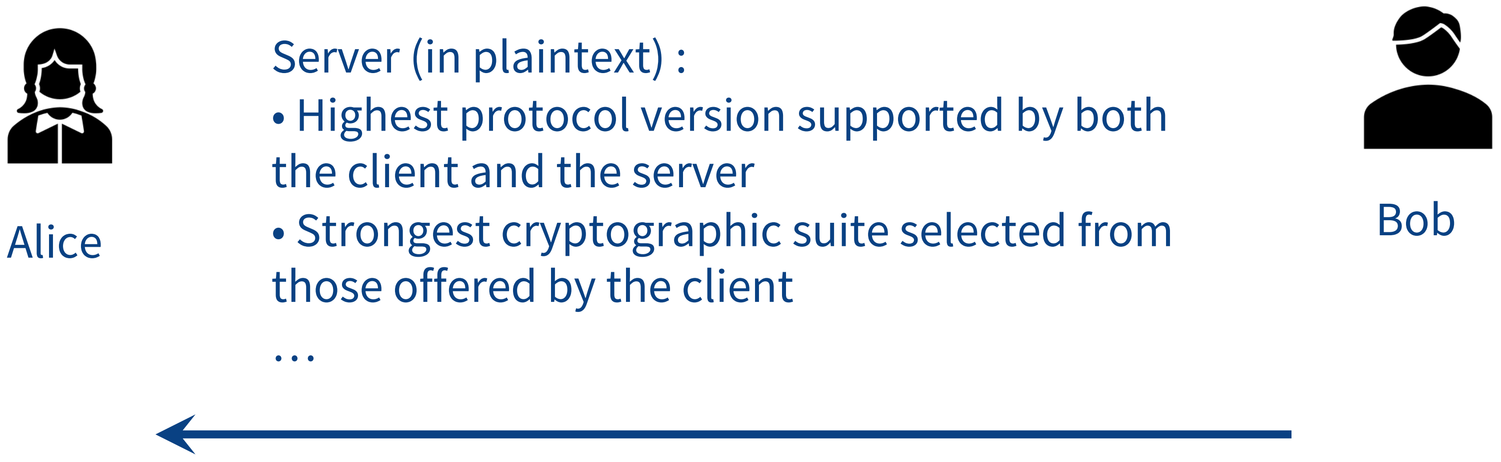
# ClientHello

Client (in plaintext):

- Protocol version
- Cryptographic algorithms
- random nonce



# ServerHello

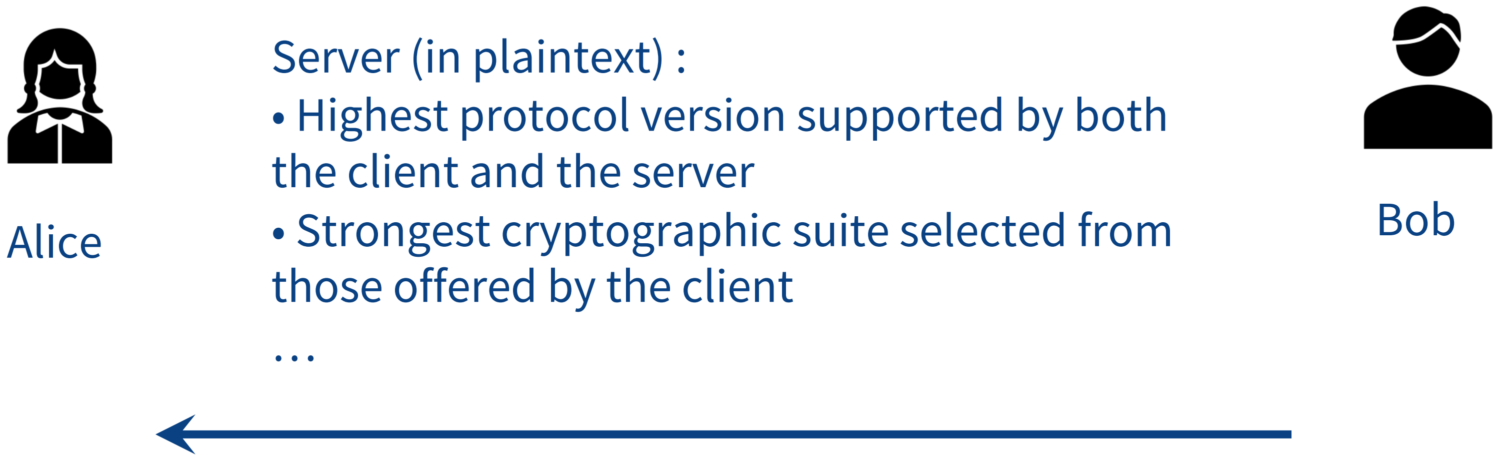


Server (in plaintext) :

- Highest protocol version supported by both the client and the server
- Strongest cryptographic suite selected from those offered by the client

…

## ServerKeyExchange



Server (in plaintext) :

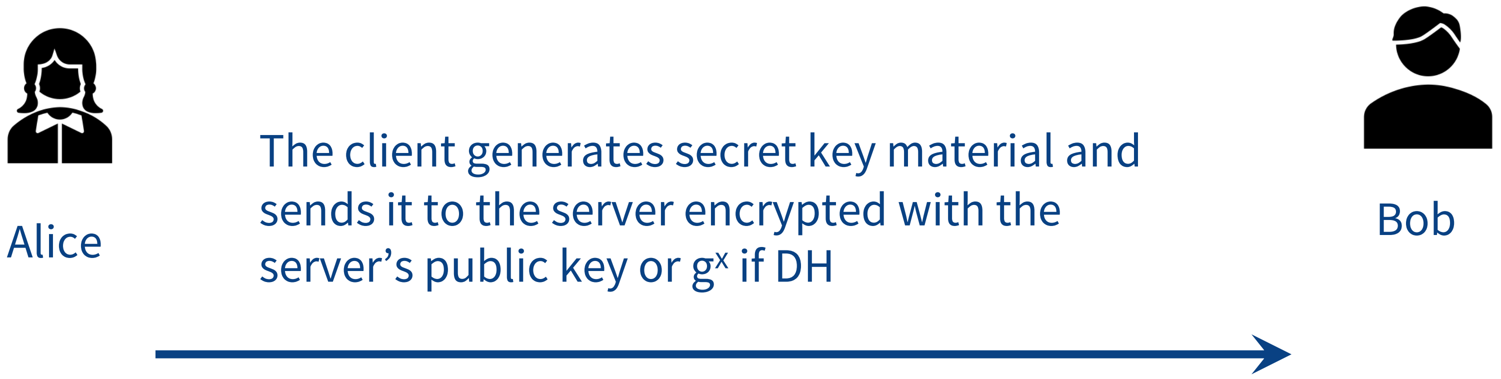
- Highest protocol version supported by both the client and the server
- Strongest cryptographic suite selected from those offered by the client

…

Server sends its public-key certificate containing either its public key, or its Diffie-Hellman public key $g^y$ (depending on chosen crypto suite)

Validate the certificate

# ClientKeyExchange



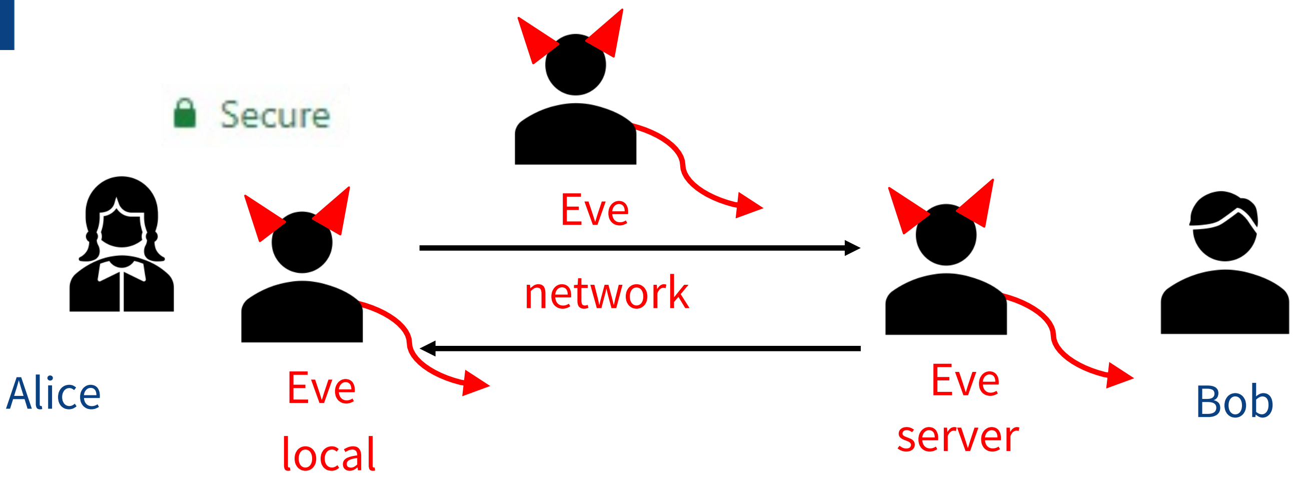
The client generates secret key material and sends it to the server encrypted with the server’s public key or $g^x$ if DH

# TLS Handshake

[over a *reliable* network connection]

- Client sends ClientHello to server asking for secure connection, listing its supported cipher suites
- Server responds with ServerHello and selects one of the cipher suites presented that it supports, also includes its certificate, and can request the client send its certificate (mutual authentication)
- Client confirms validity of certificate
- Client generates session key
  - Either directly by picking a random key and encrypting it with the public key of the server, or
  - By running the Diffie-Hellman Key Exchange protocol that provides better security
- Handshake concludes and both parties share a key that is then used for encrypting/decrypting messages

# Beyond TLS



- Certificates and Certificate Authorities provide authentication
- TLS provides private communication with integrity guarantees
- It’s on this basis that the public are told to check for the padlock to know their communication is secure and protected between their web browser and the server, and is with the genuine server

## Summary

- Secure communication is complex
- Relies on several cryptographic protocols
- TLS (see LMS for links on attacks against TLS)

## Acknowledgement

- The slides were prepared by Olya Ohrimenko based on some material developed previously by Chris Culnane
- Reference: KR 8.3, 8.6