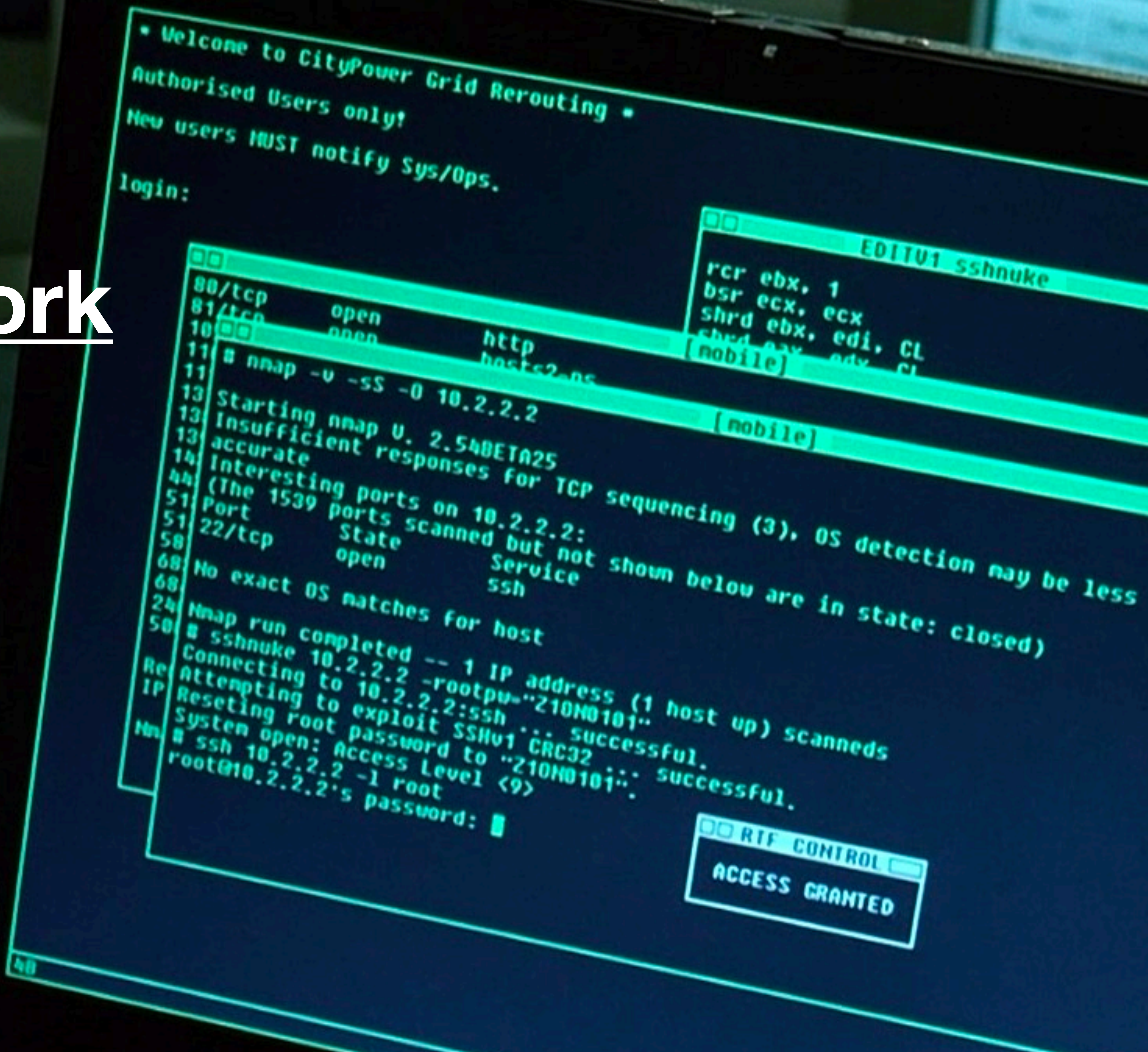
Computer Network Security

ECE 4112/6612

CS 4262/6262

Prof. Frank Li



Logistics

No class next Tuesday, **fall break!**

HW2 due Tuesday, Oct 17 midnight

Project proposal comments provided

End-to-End Secure Communication through TLS

Building Secure End-to-End Channels

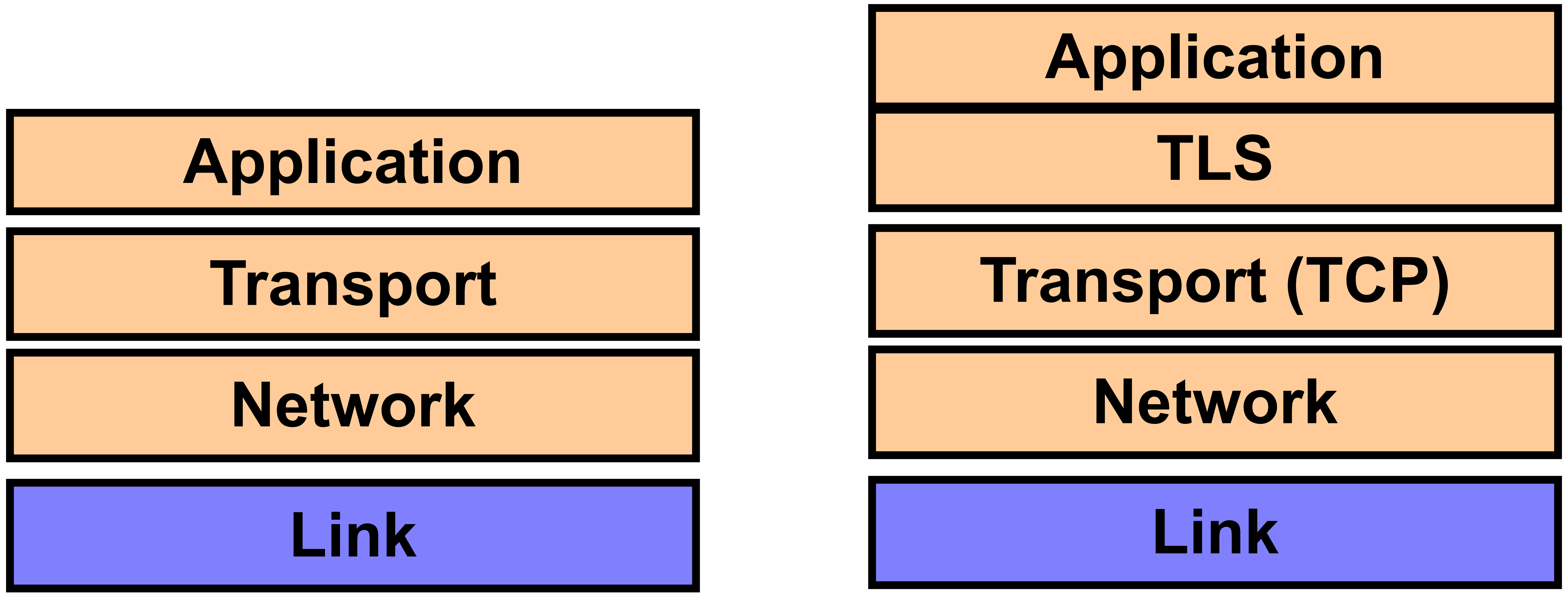
- *End-to-end* = communication protections achieved all the way from originating client to intended server
  - *With no need to trust intermediaries*
- Dealing with threats:
  - Eavesdropping?
    - Encryption (including session keys)
  - Manipulation (injection, MITM)?
    - Integrity (use of a MAC); *replay protection* *(use of nonce)*
  - Impersonation?
    - Signatures

(What's missing? *Availability …*)

Building A Secure End-to-End Channel: SSL/TLS

- SSL = *Secure Sockets Layer* (predecessor, deprecated)
- TLS = *Transport Layer Security* (standard)
- Notion: provide means to secure *any* application that uses TCP

TLS In Network Layering



Building A Secure End-to-End Channel: SSL/TLS

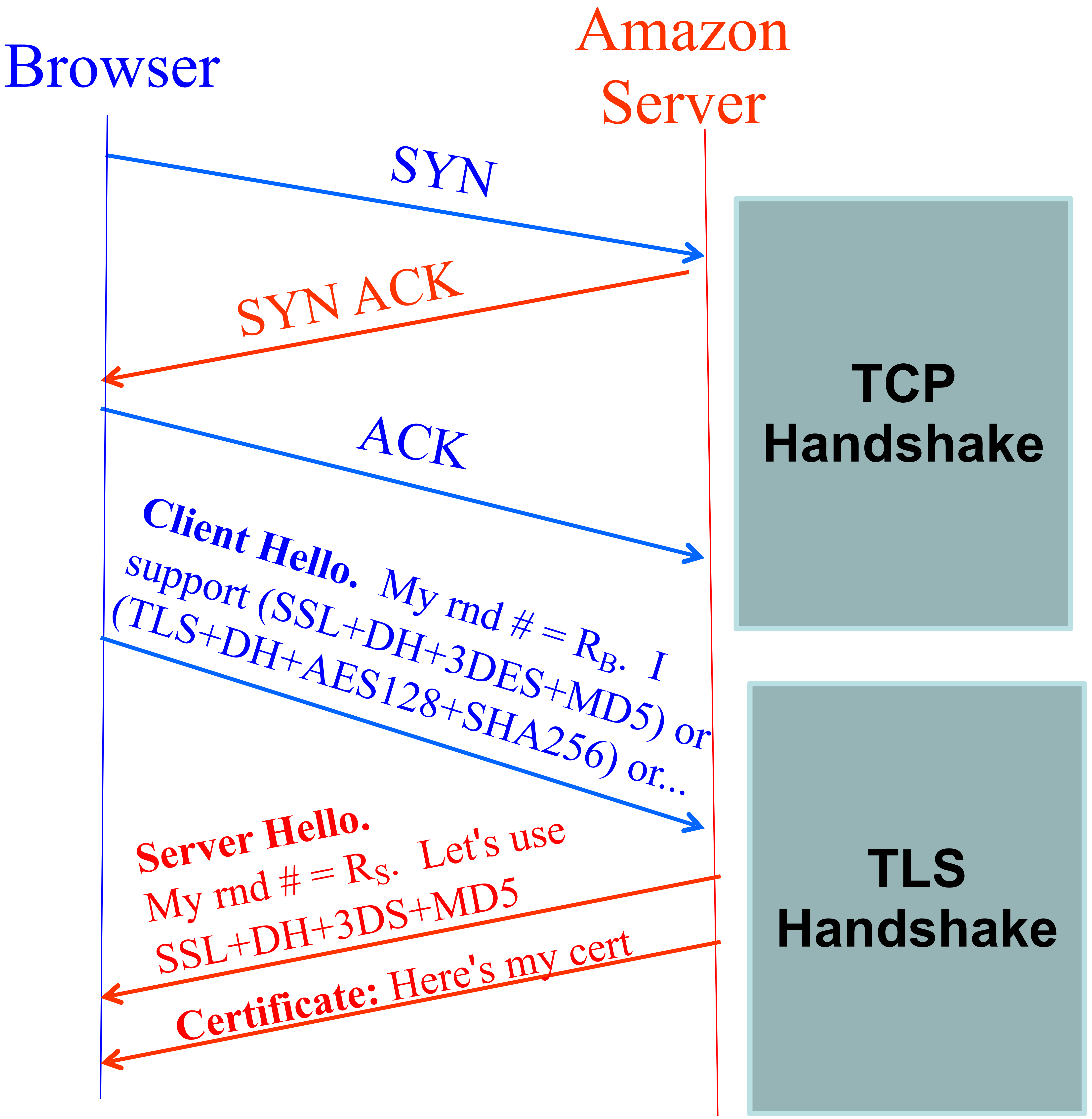
- SSL = *Secure Sockets Layer* (predecessor, deprecated)
- TLS = *Transport Layer Security* (standard)
  - Both terms used interchangeably
- Notion: provide means to secure *any* application that uses TCP
  - Secure = encryption/confidentiality + integrity + authentication (of server, but typically *not* of client)
  - E.g., puts the ‘s’ in “https”

Building A Secure End-to-End Channel: SSL/TLS

- SSL = *Secure Sockets Layer* (predecessor, deprecated)
- TLS = *Transport Layer Security* (standard)
  - Both terms used interchangeably
- Notion: provide means to secure *any* application that uses TCP
  - Secure = encryption/confidentiality + integrity + authentication (of server, but typ. not of client)
  - E.g., puts the ‘s’ in “https”
- TLS 1.3 is newest version, finalized in 2018. TLS 1.2 is still most prevalent.

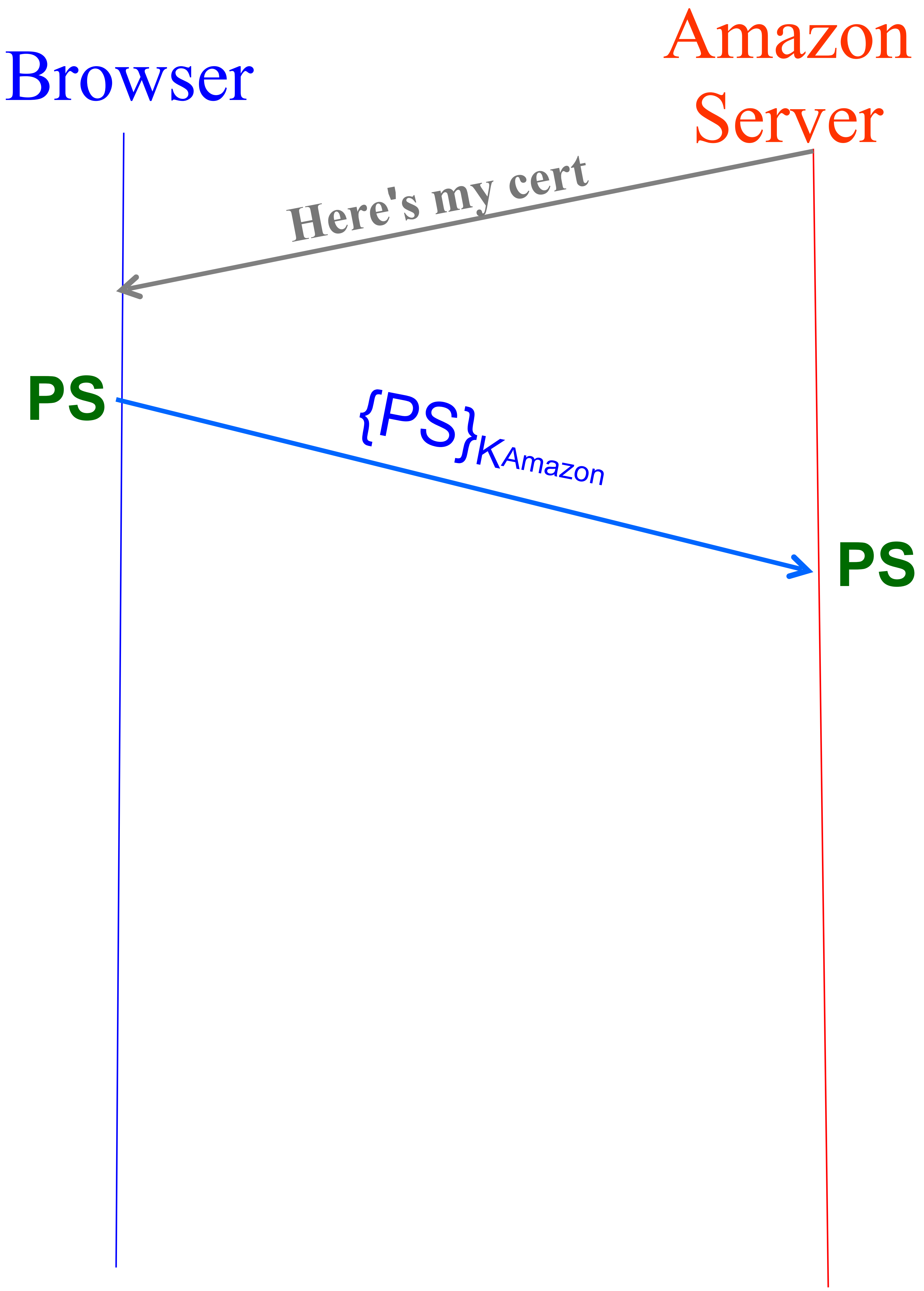
HTTPS Connection (TLS 1.2)

- Browser (client) connects via TCP to Amazon's **HTTPS** server
- **Client Hello:** Client picks 256-bit random number R_B, sends over list of crypto protocols it supports
- **Server Hello:** Server picks 256-bit random number R_S, selects *cipher suite* to use for this session
- **Certificate:** Server sends over its certificate (w/ it's public key)
- (all of this is in the clear)
- ***Client now validates cert***



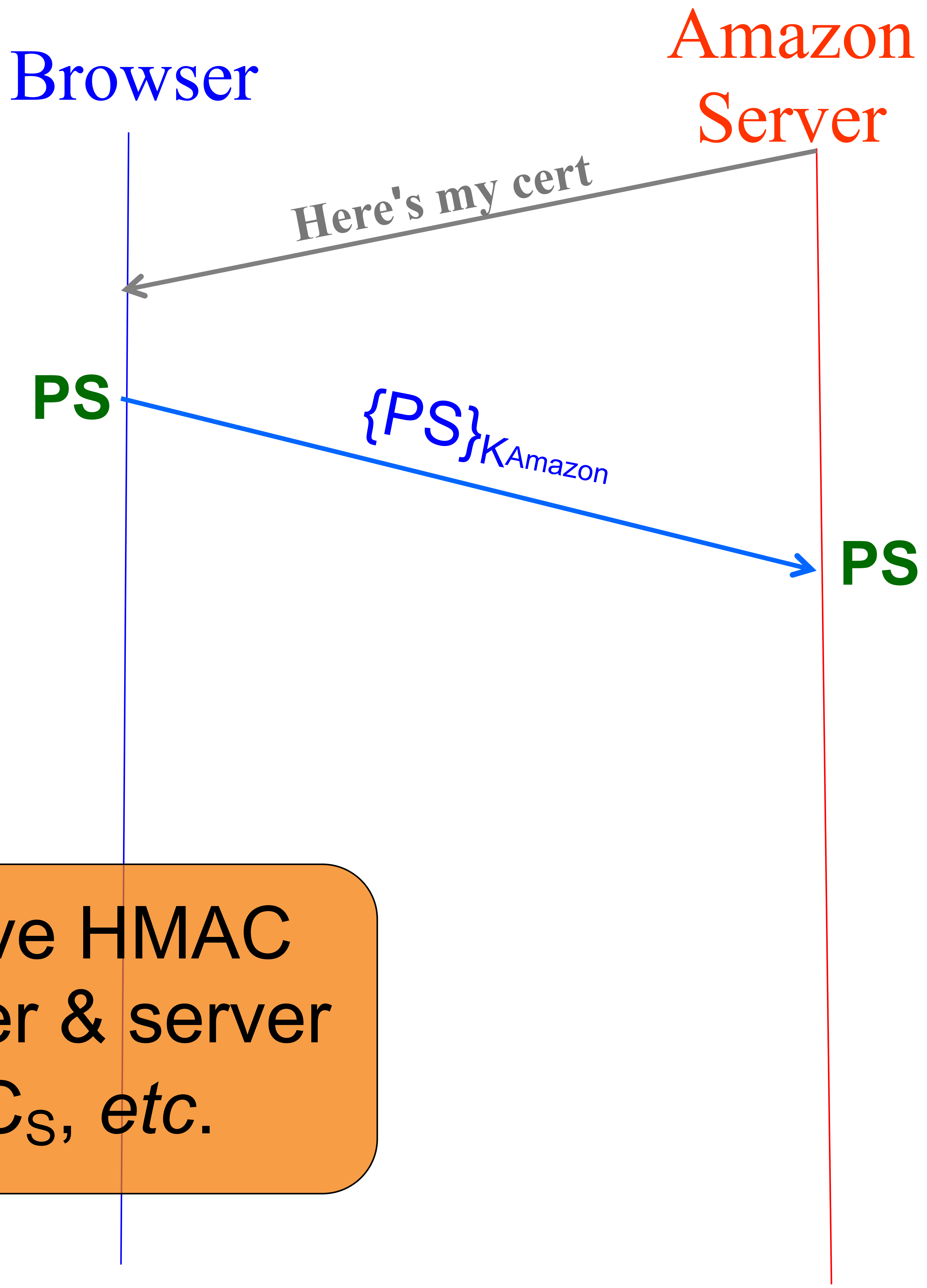
`HTTPS` Connection (TLS 1.2), con't

- For RSA, browser constructs long (368 bits) "Premaster Secret" **PS**
- Browser sends PS encrypted using Amazon's public RSA key K_{Amazon}
- Using PS, R_B, and R_S, browser & server derive symmetric *cipher keys* (C_B, C_S) & MAC *integrity keys* (I_B, I_S)
  - One pair to use in each direction



HTTPS Connection (TLS 1.2), con't

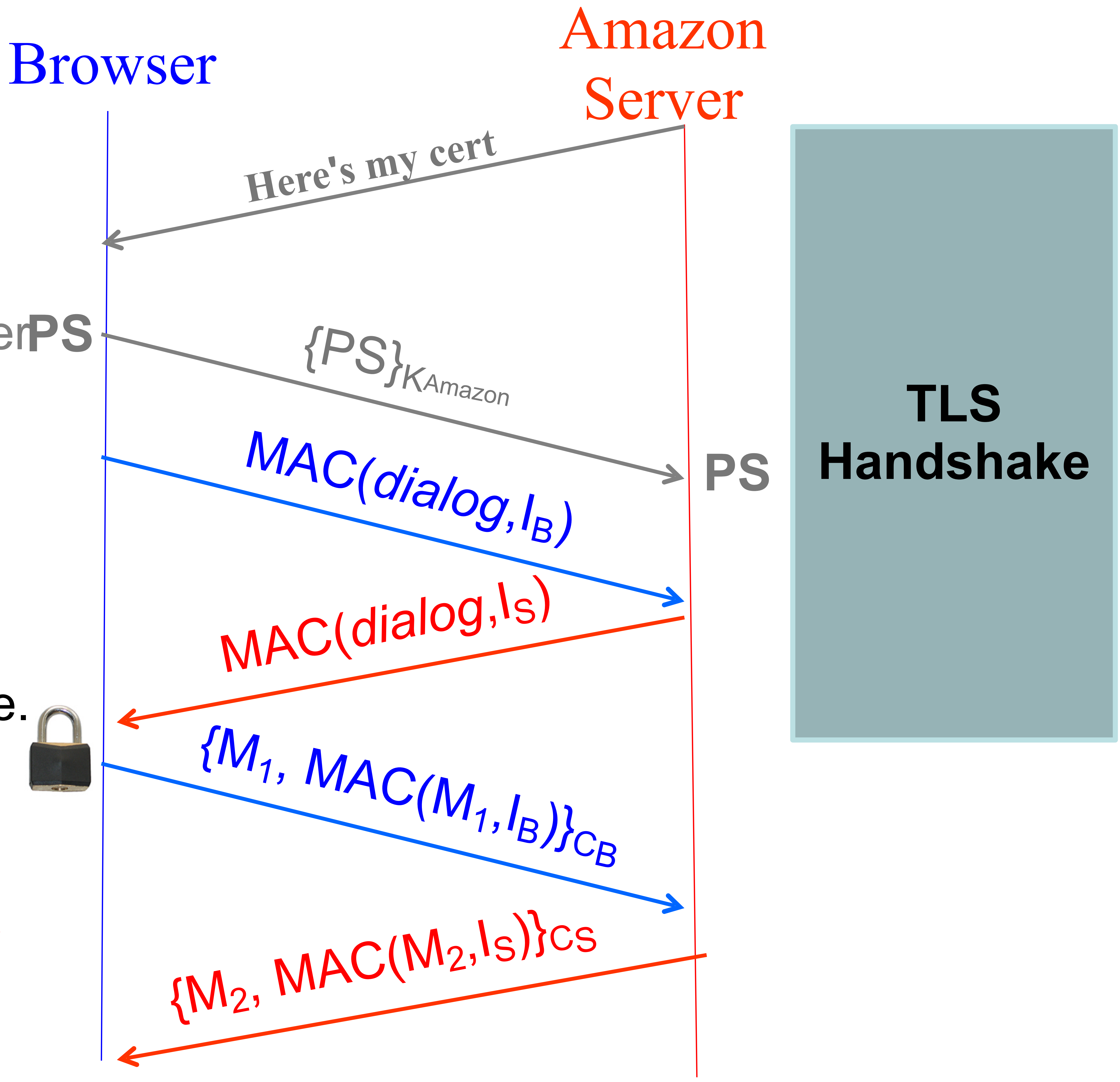
- For RSA, browser constructs long (368 bits) "Premaster Secret" **PS**
- Browser sends PS encrypted using Amazon's public RSA key K_{Amazon}
- Using PS, R_B, and R_S, browser & server derive symmetric *cipher keys* (C_B, C_S) & MAC *integrity keys* (I_B, I_S)
  - One pair to use in each direction



PS is used as the key for iterative HMAC invocations on R_B || R_S. Browser & server use the output to generate C_B, C_S, *etc.*

HTTPS Connection (TLS 1.2), con't

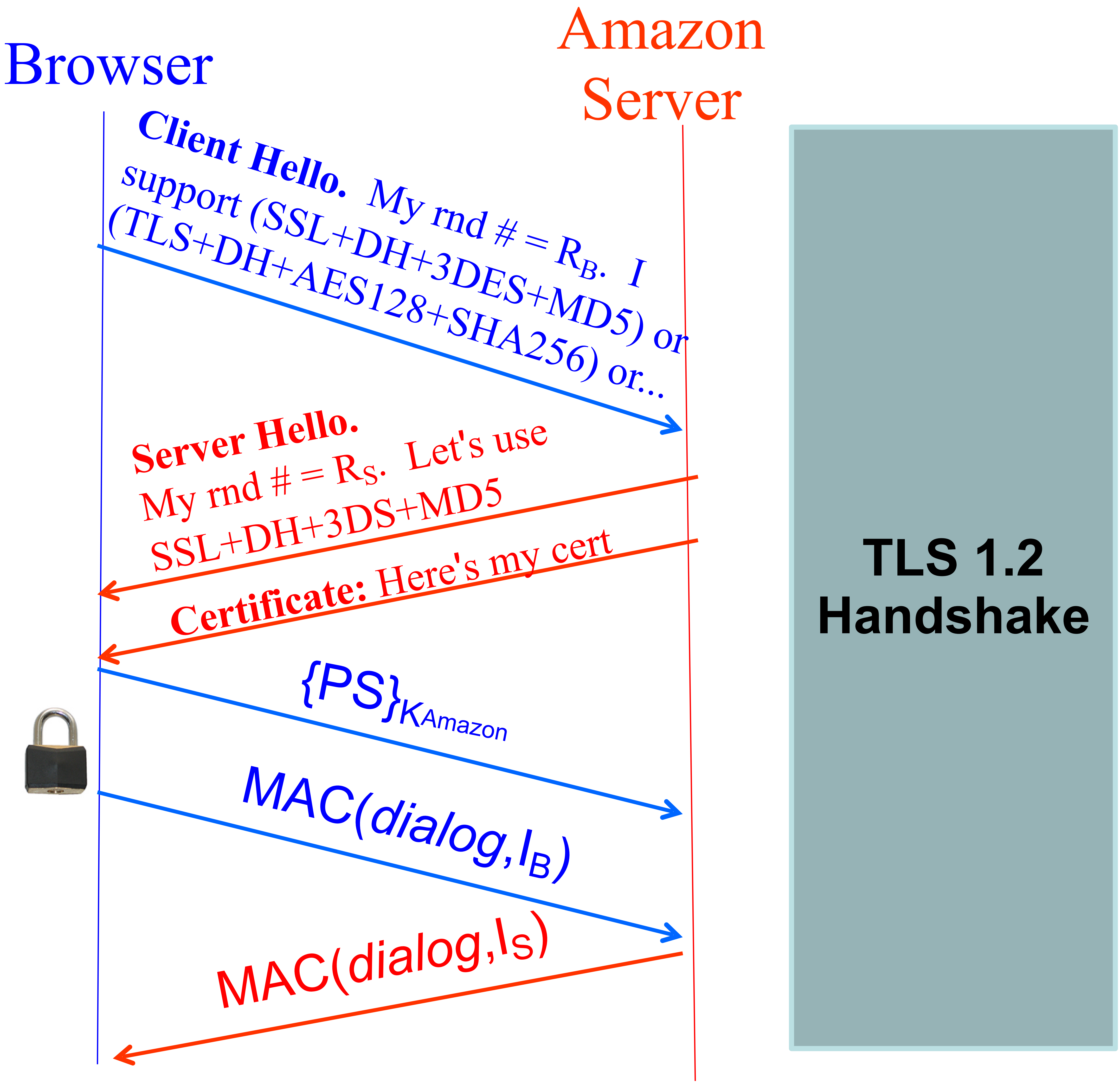
- For RSA, browser constructs long (368 bits) "Premaster Secret" **PS**
- Browser sends PS encrypted using Amazon's public RSA key K_{Amazon}
- Using PS, R_B, and R_S, browser & server derive symmetric *cipher keys* (C_B, C_S) & MAC *integrity keys* (I_B, I_S)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, conclude TLS handshake.
- All subsequent communication encrypted w/ symmetric cipher (e.g., AES128) cipher keys, MACs
  - Messages also numbered to thwart replay attacks



HTTPS Connection (TLS 1.2), con't

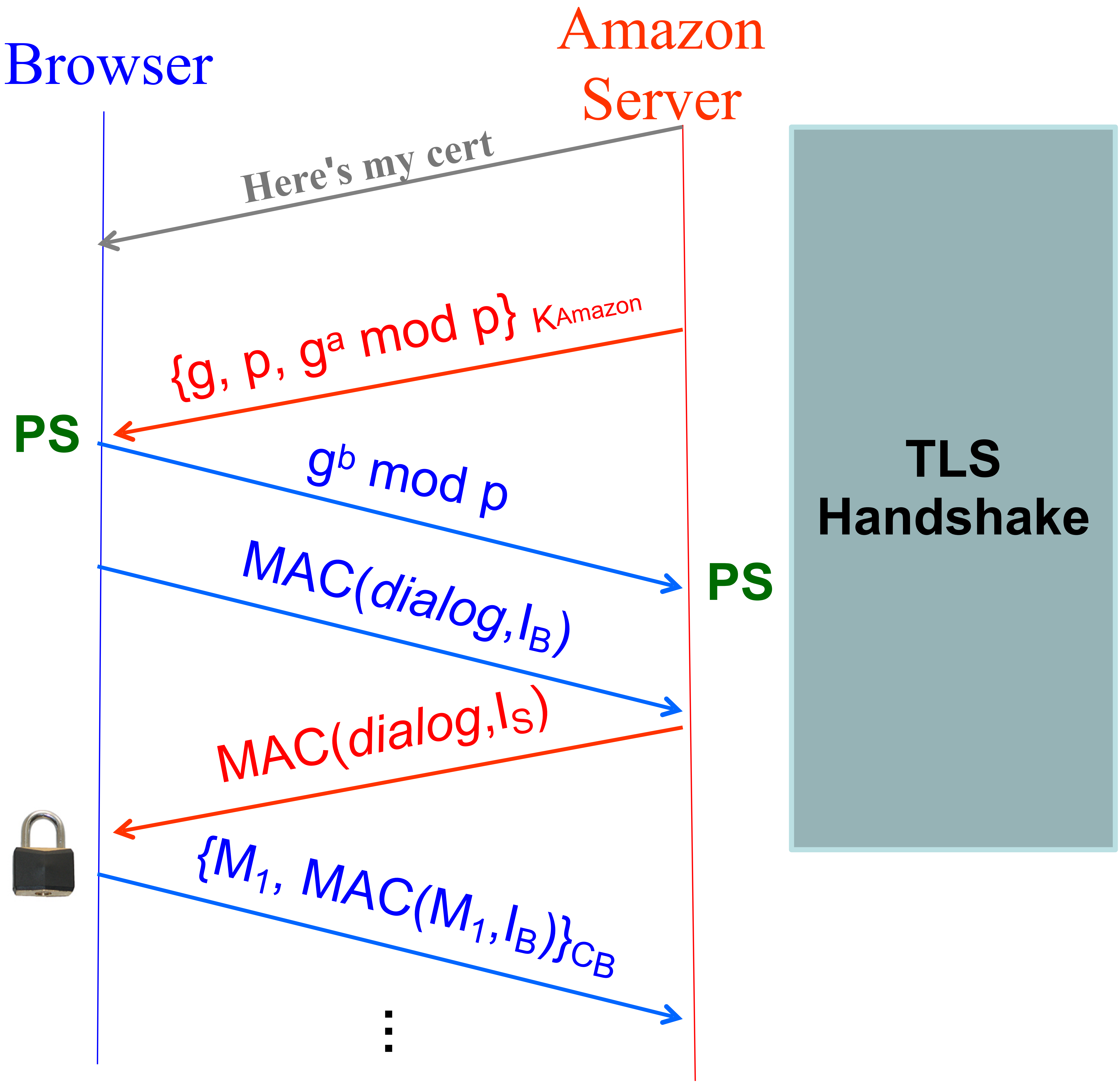
What if Mallory attacks?

- Can Mallory pretend to be Amazon?
  - No! Browser validates Amazon's certificate **and** handshake requires demonstration that server has the private key associated with the cert.
- Can Mallory figure out the TLS keys?
  - No! PS is encrypted under Amazon's public key. Need PS to compute TLS keys.
- Tampers with any handshake data?
  - Dialog check at the end (using MACs) won't pass



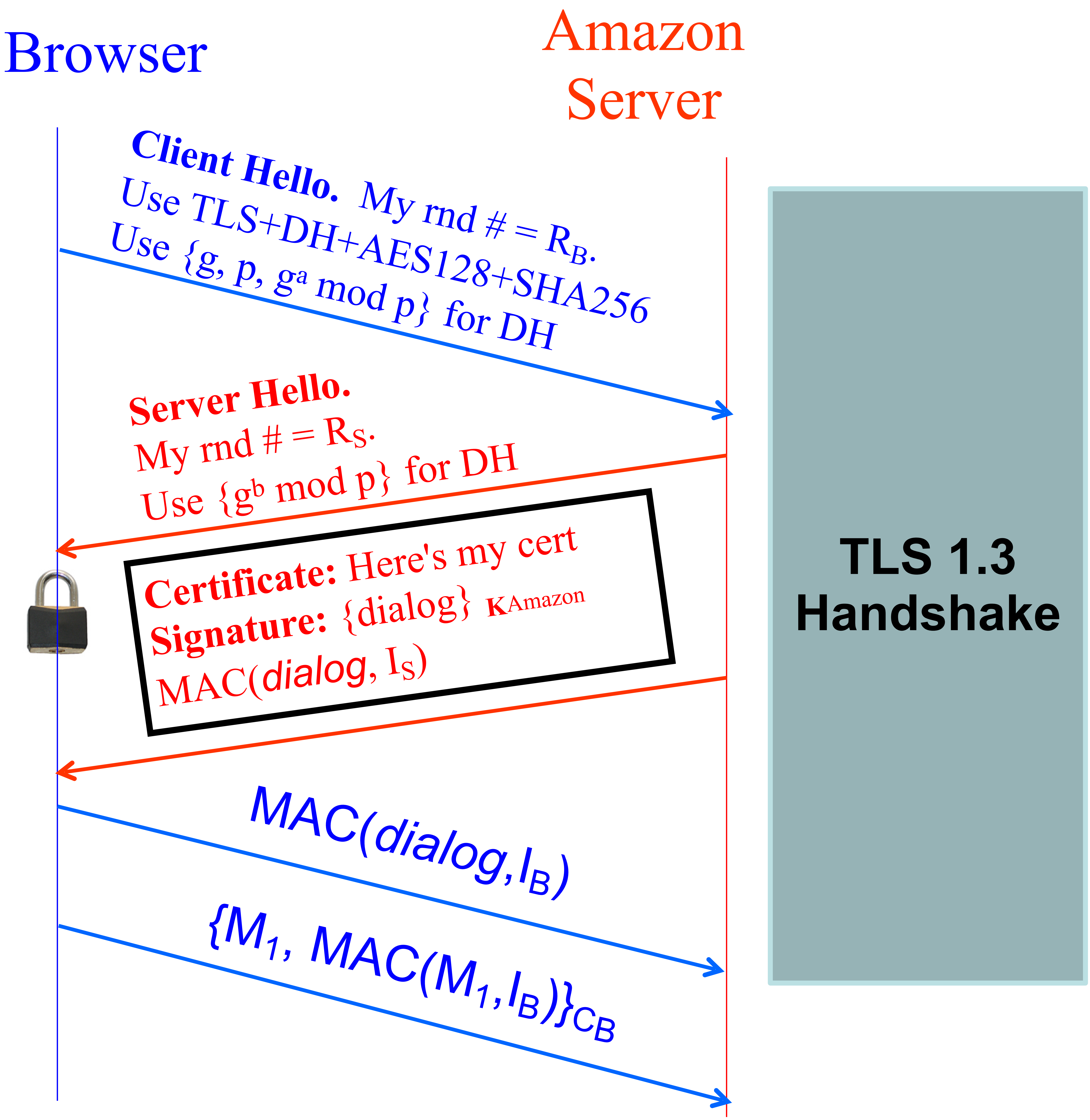
Alternative: Key Exchange via Diffie-Hellman

- For Diffie-Hellman, server generates random a, sends public params and g^a mod p
  - Signed with server's private key
- Browser verifies signature
- Browser generates random b, computes **PS** = g^{ab} mod p, sends to server
- Server also computes **PS** = g^{ab} mod p
- Remainder is as before: from PS, R_B, and R_S, browser & server derive symm. *cipher keys* (C_B, C_S) and MAC *integrity keys* (I_B, I_S), etc…

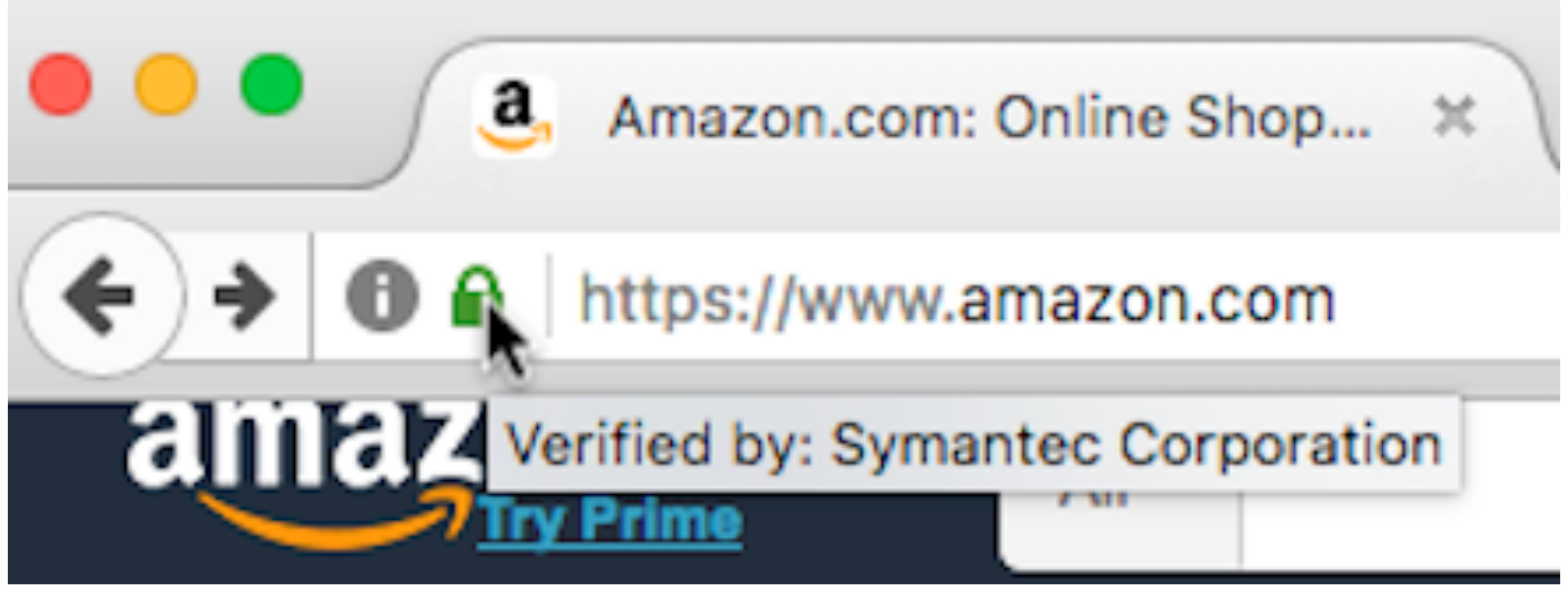


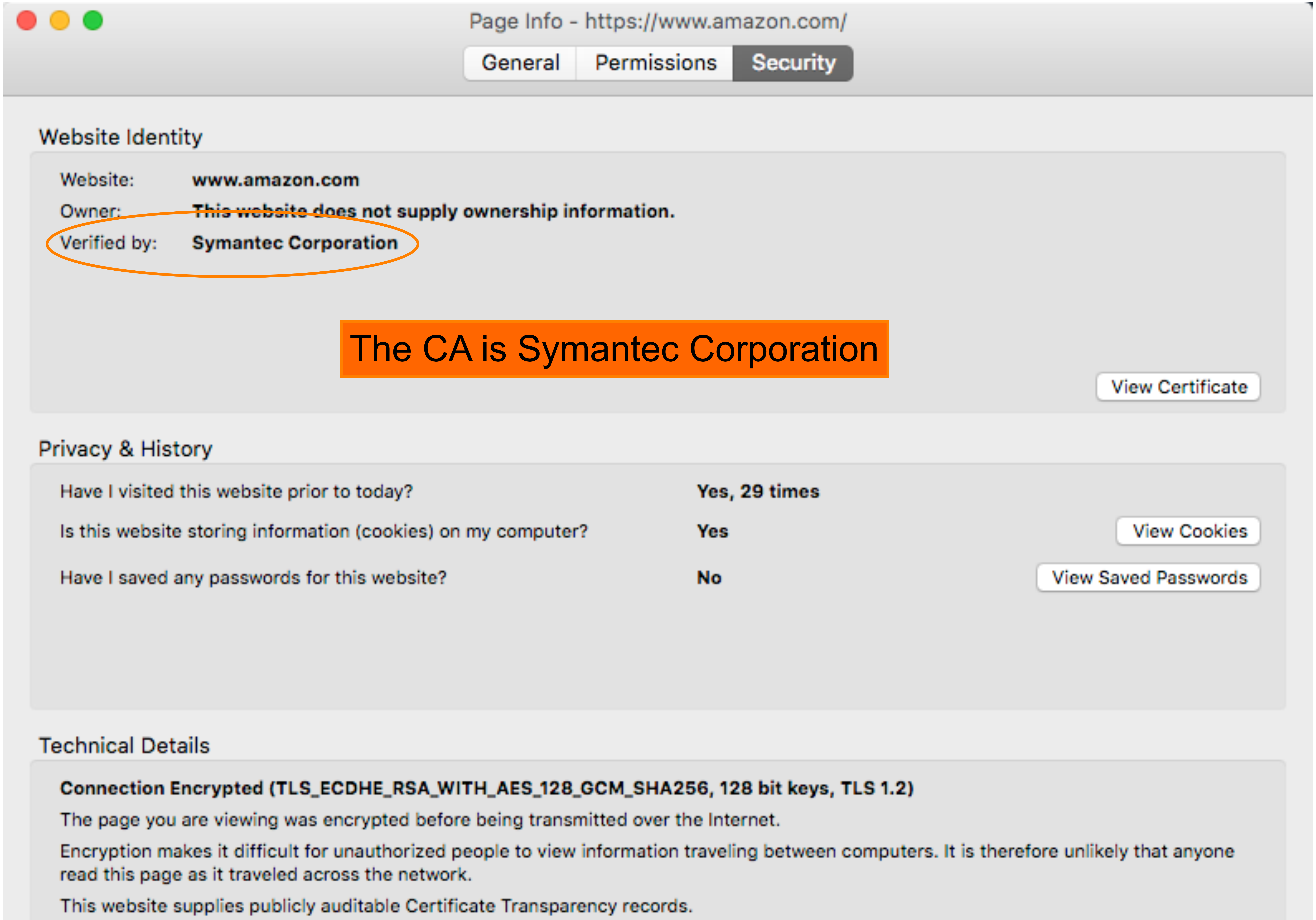
TLS 1.3

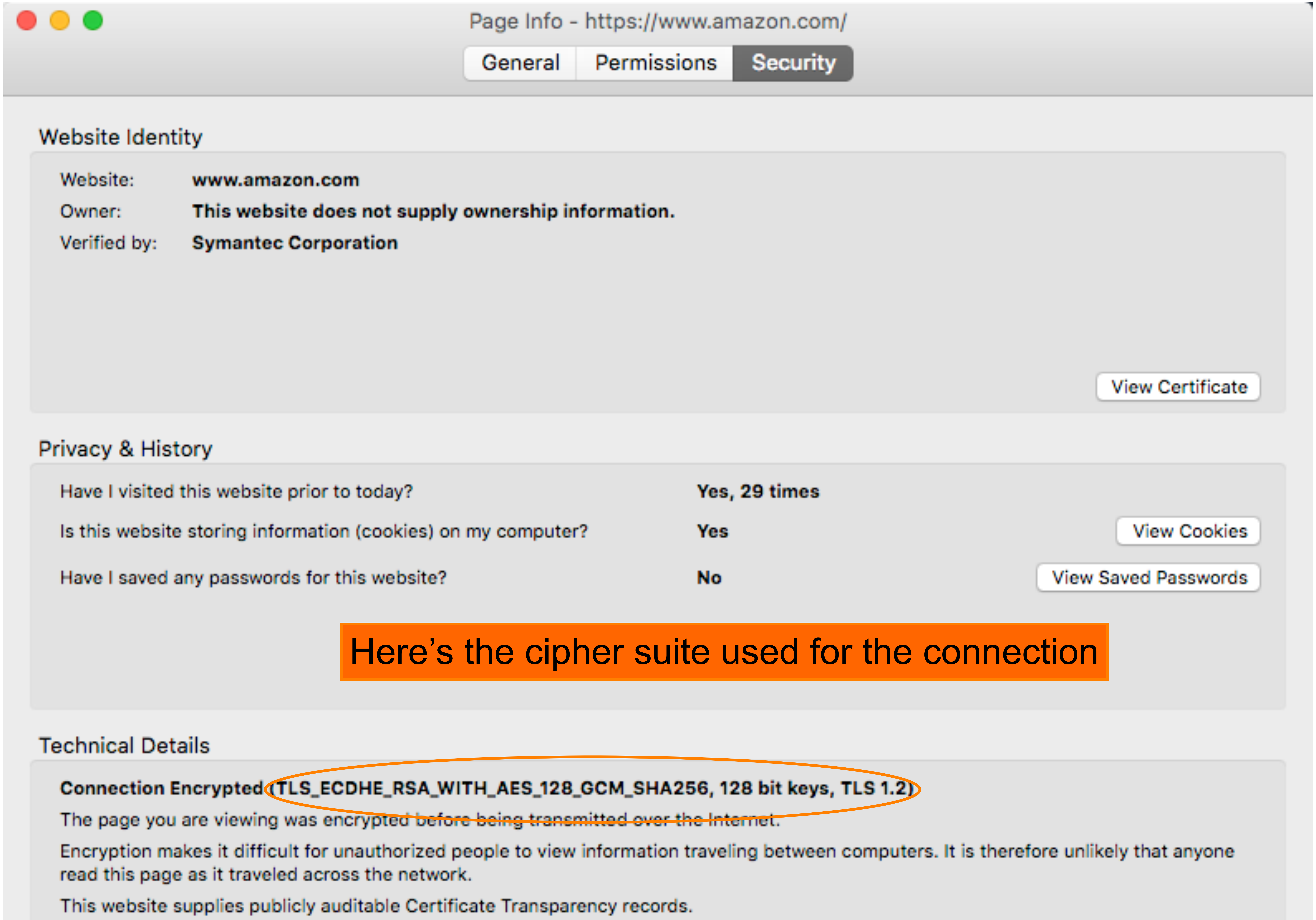
- Only secure ciphersuites allowed (no more RSA!)
- Fast handshake: TLS 1.2 is 2 RTT, TLS 1.3 is 1 RTT
  - Client guesses ciphersuite server supports. If so, just use that.
  - Client sends key material immediately.
- Client Hello packet is not encrypted, but ServerHello is (earlier encryption)
- More performance/security optimizations....



What’s Inside a Cert?


Amazon.com: Online Shop...
https://www.amazon.com
Verified by: Symantec Corporation
Try Prime


Page Info - https://www.amazon.com/
General
Permissions
Security
Website Identity
Website: www.amazon.com
Owner: This website does not supply ownership information.
Verified by: Symantec Corporation
The CA is Symantec Corporation
View Certificate
Privacy & History
Have I visited this website prior to today? Yes, 29 times
Is this website storing information (cookies) on my computer? Yes
View Cookies
Have I saved any passwords for this website? No
View Saved Passwords
Technical Details
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.
This website supplies publicly auditable Certificate Transparency records.


Page Info - https://www.amazon.com/
General
Permissions
Security
Website Identity
Website: www.amazon.com
Owner: This website does not supply ownership information.
Verified by: Symantec Corporation
View Certificate
Privacy & History
Have I visited this website prior to today? Yes, 29 times
Is this website storing information (cookies) on my computer? Yes
View Cookies
Have I saved any passwords for this website? No
View Saved Passwords
Here’s the cipher suite used for the connection
Technical Details
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.
This website supplies publicly auditable Certificate Transparency records.

Certificate Viewer: "www.amazon.com"

General | Details

This certificate has been verified for the following uses:

SSL Client Certificate

SSL Server Certificate

Issued To

Common Name (CN)	www.amazon.com
Organization (O)	Amazon.com, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	1D:4A:BD:AA:78:D0:9A:FE:79:9D:41:BC:EB:7A:76:62

Issued By

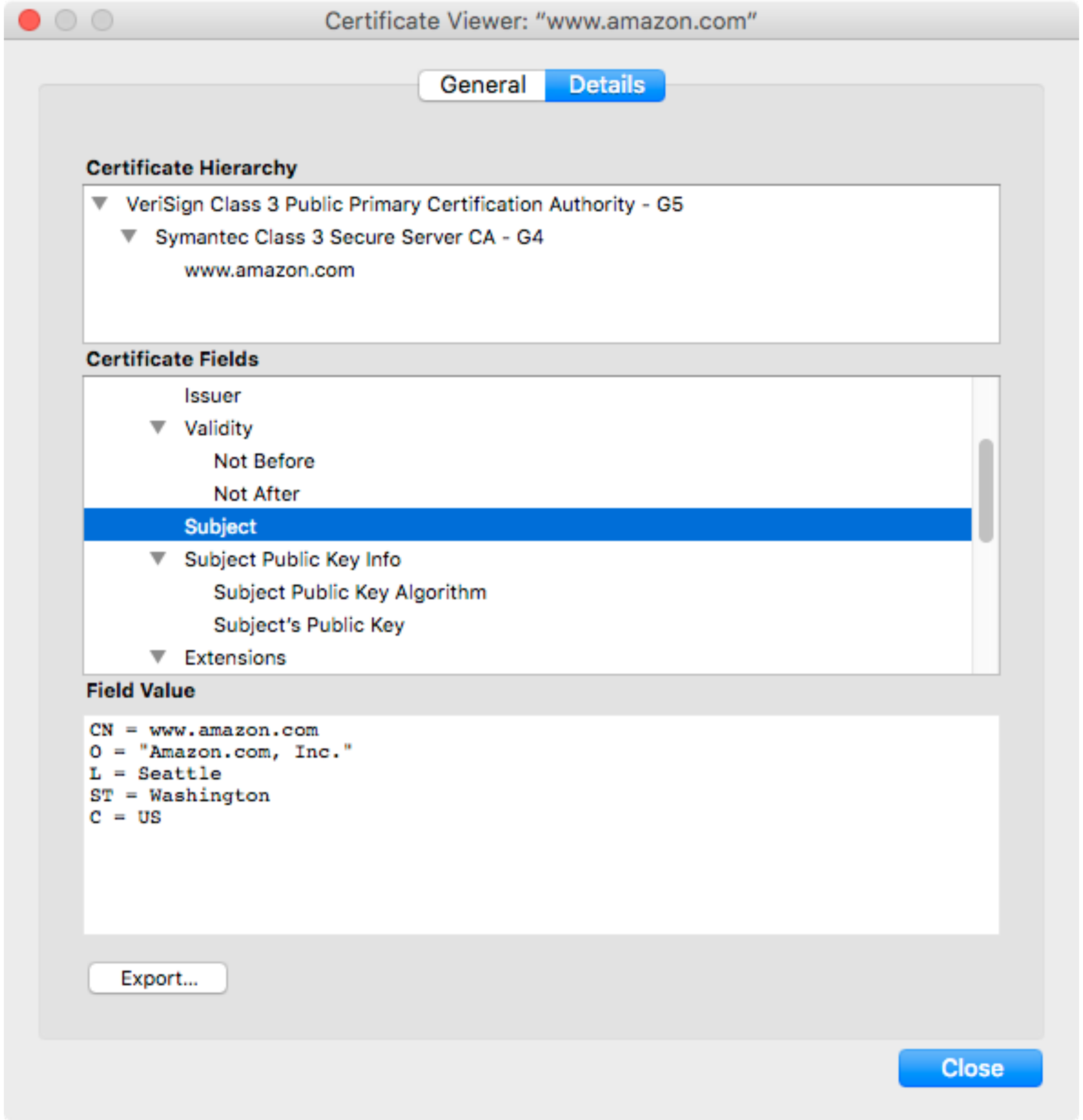
Common Name (CN)	Symantec Class 3 Secure Server CA - G4
Organization (O)	Symantec Corporation
Organizational Unit (OU)	Symantec Trust Network

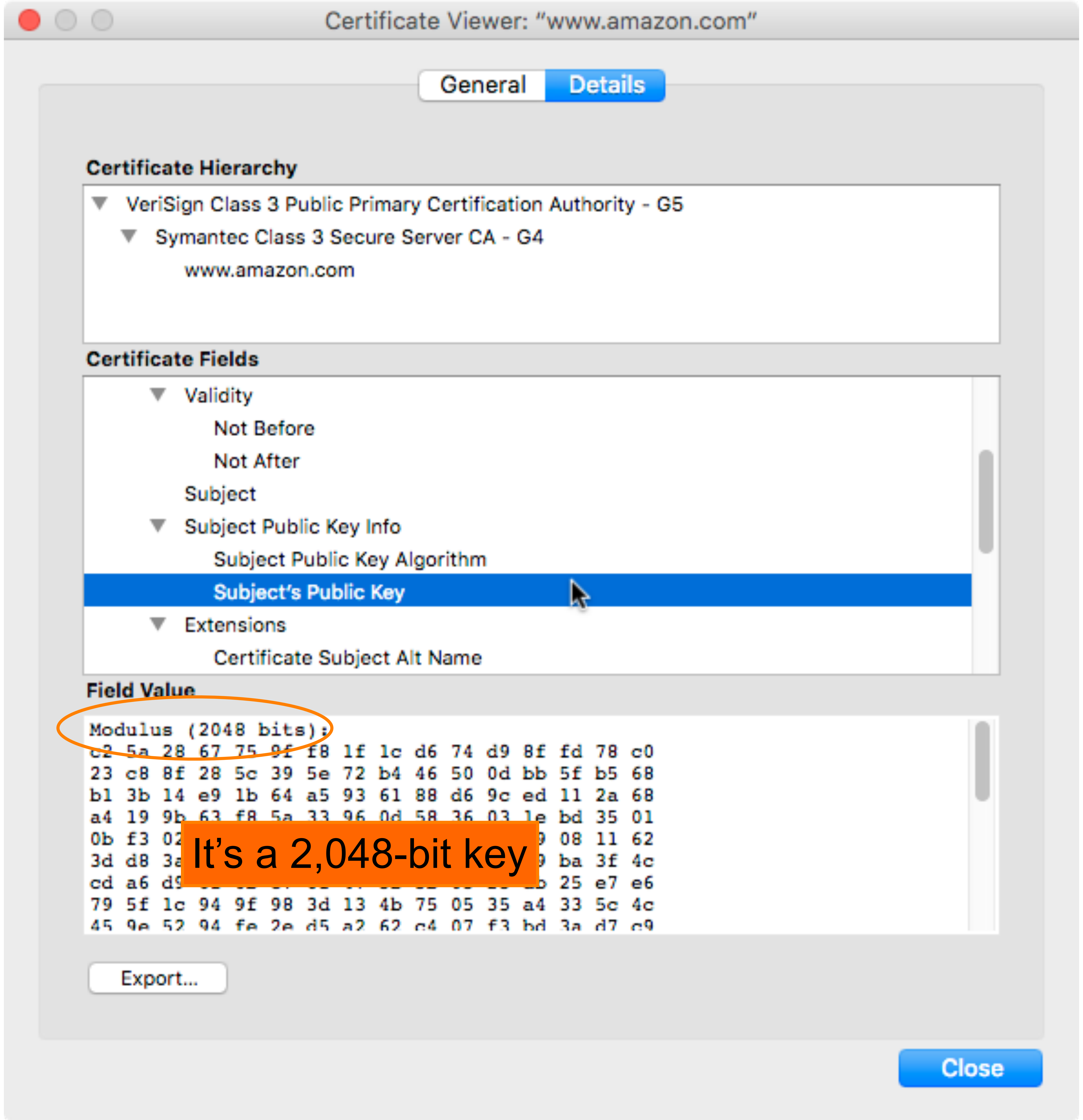
Period of Validity

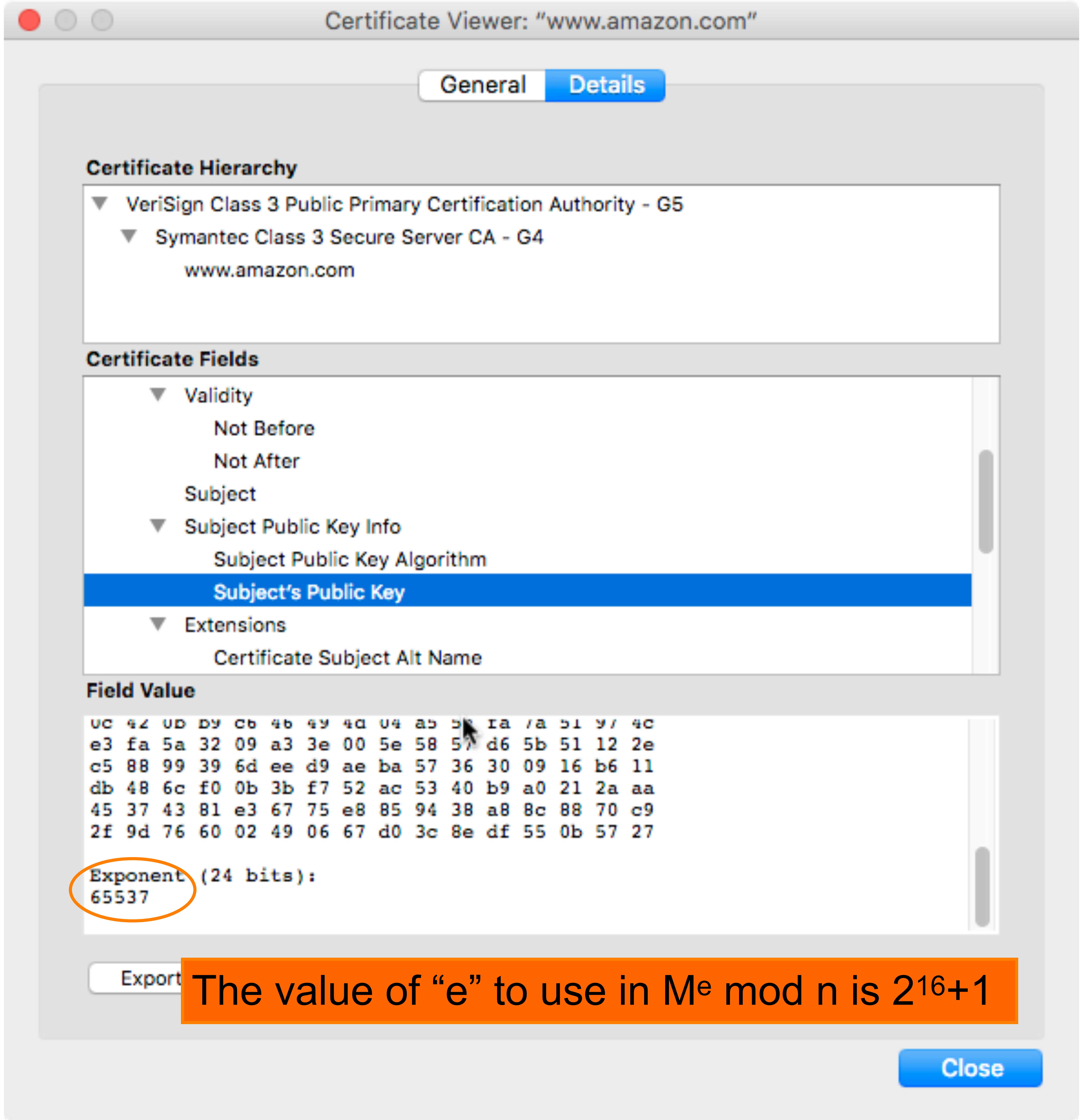
Begins On	October 30, 2016
Expires On	December 31, 2017

Fingerprints

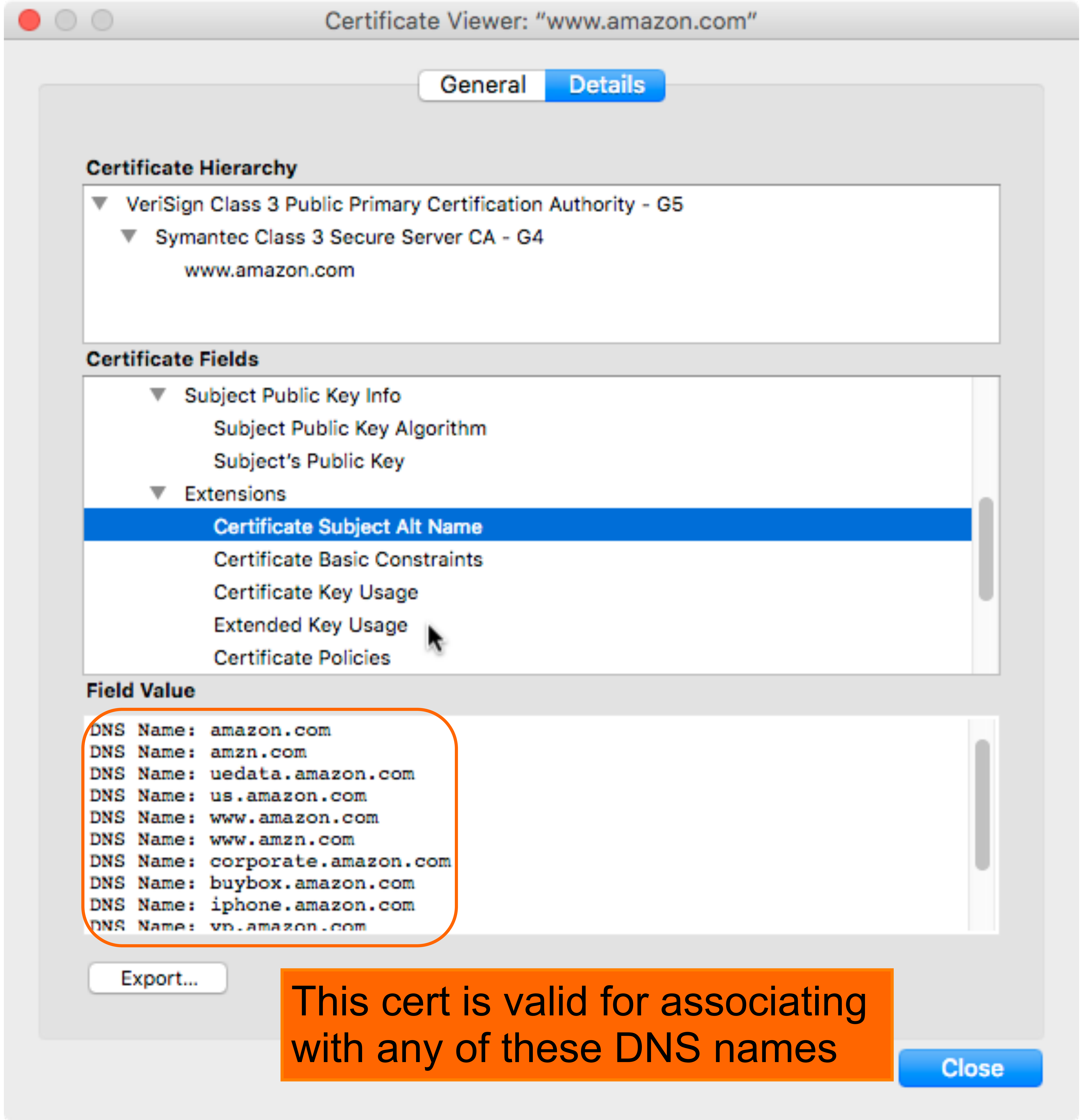
SHA-256 Fingerprint	6A:A0:AB:97:D0:F9:F1:50:58:96:31:3B:E2:37:2D:C3: 94:BD:42:77:57:F6:BD:B6:2D:DE:80:ED:54:D4:19:0D
SHA1 Fingerprint	EF:14:6C:F1:5C:4A:F8:4D:BA:83:C2:1E:6C:5B:ED:C4:FA:34:1C:3E

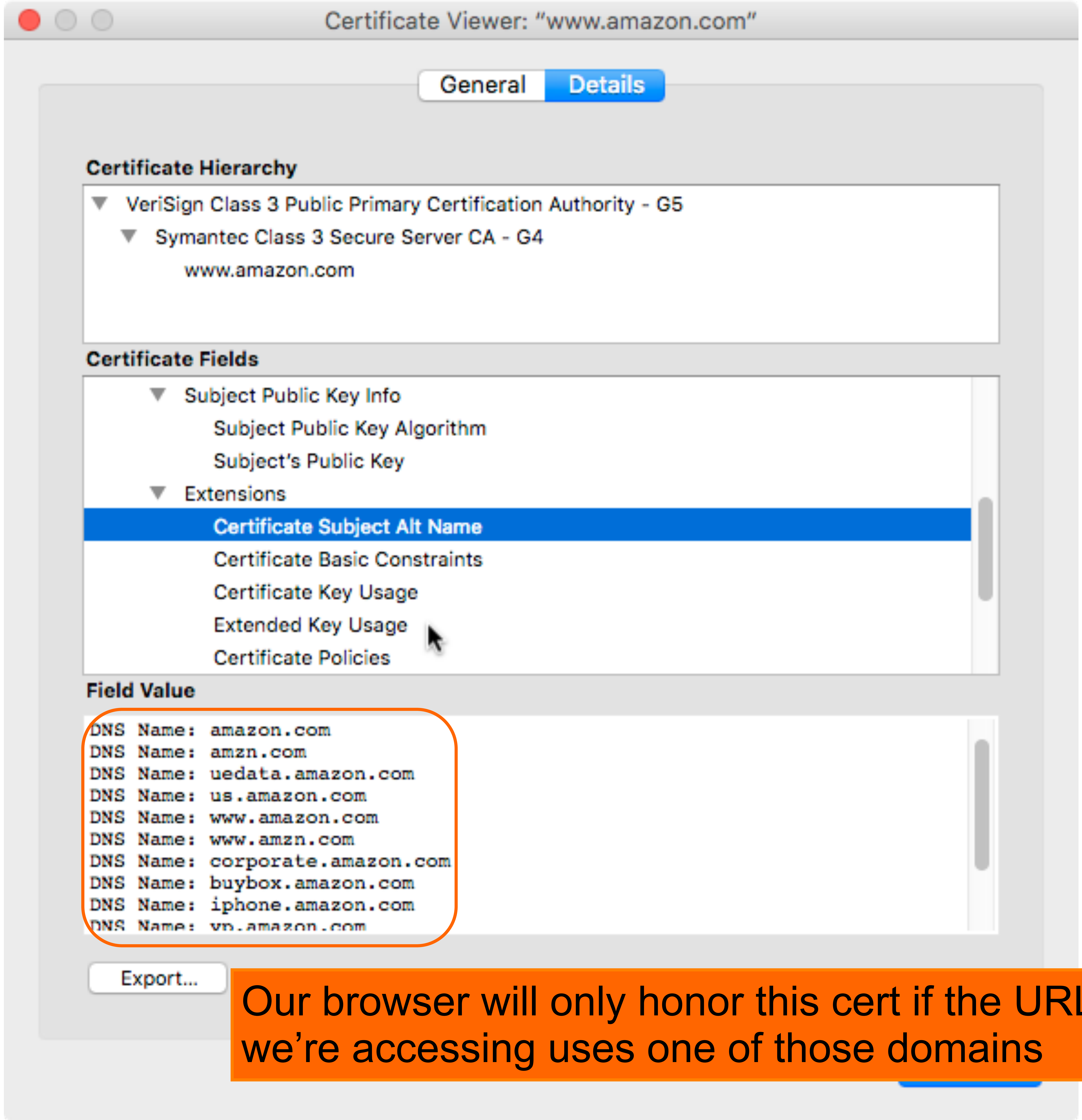

Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Issuer
Validity
Not Before
Not After
Subject
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Field Value
CN = www.amazon.com
O = "Amazon.com, Inc."
L = Seattle
ST = Washington
C = US
Export...
Close


Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Validity
Not Before
Not After
Subject
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Certificate Subject Alt Name
Field Value
Modulus (2048 bits):
c2 5a 28 67 75 9f f8 1f 1c d6 74 d9 8f fd 78 c0
23 c8 8f 28 5c 39 5e 72 b4 46 50 0d bb 5f b5 68
b1 3b 14 e9 1b 64 a5 93 61 88 d6 9c ed 11 2a 68
a4 19 9b 63 f8 5a 33 96 0d 58 36 03 1e bd 35 01
79 5f 1c 94 9f 98 3d 13 4b 75 05 35 a4 33 5c 4c
45 9e 52 94 fe 2e d5 a2 62 c4 07 f3 bd 3a d7 c9
It's a 2,048-bit key
Export...
Close

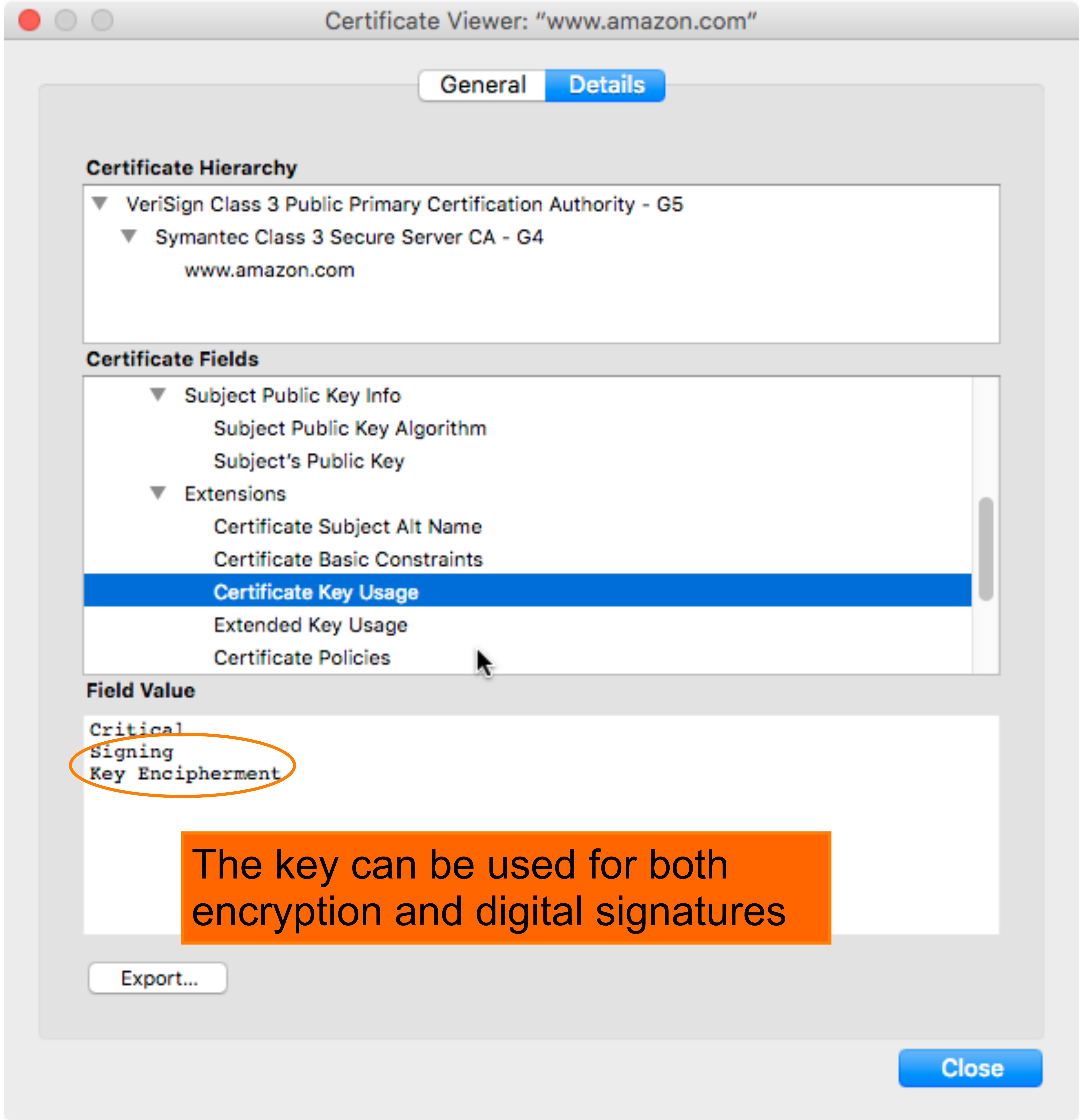


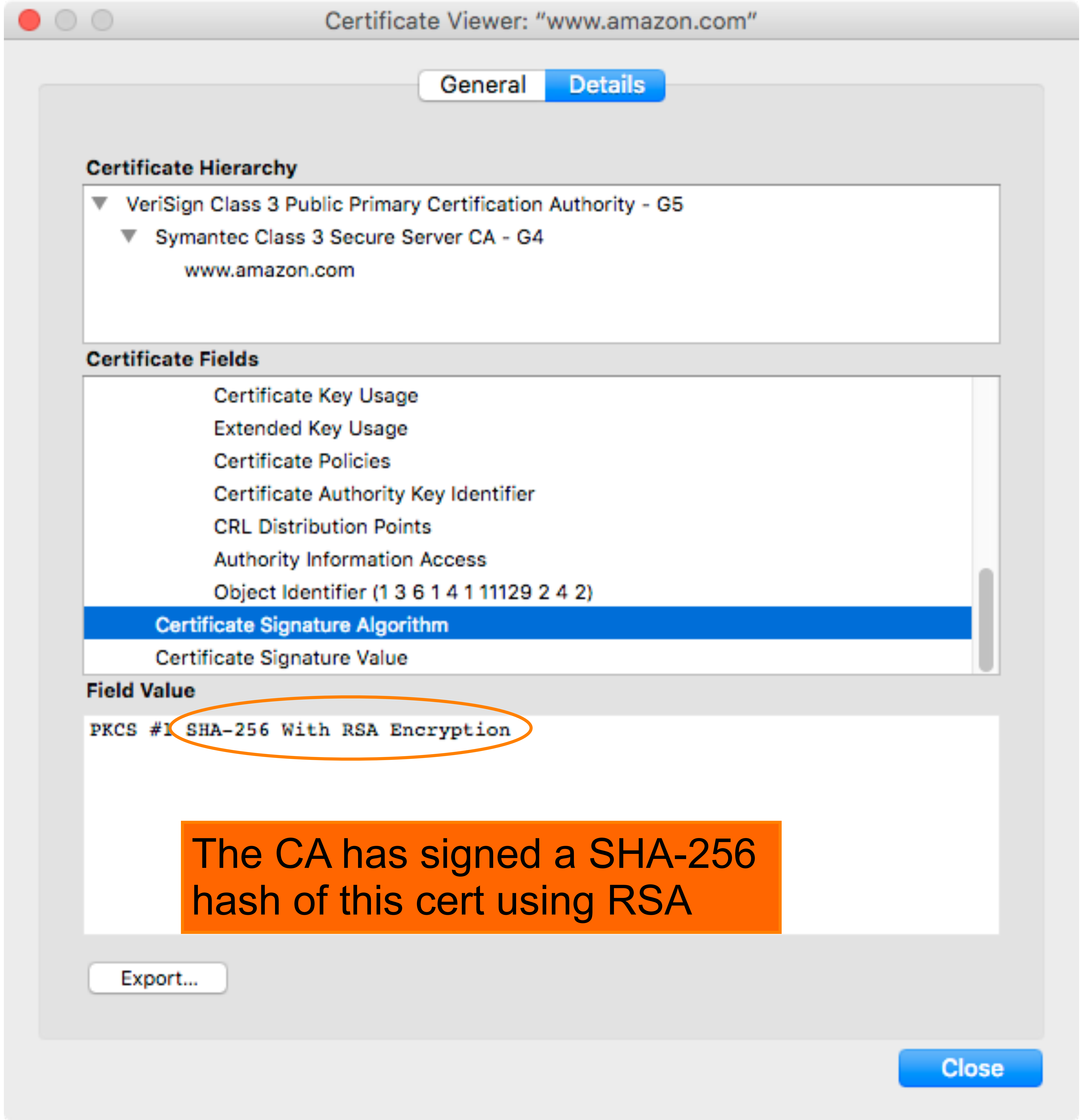
The value of "e" to use in M^e mod n is $2^{16}+1$

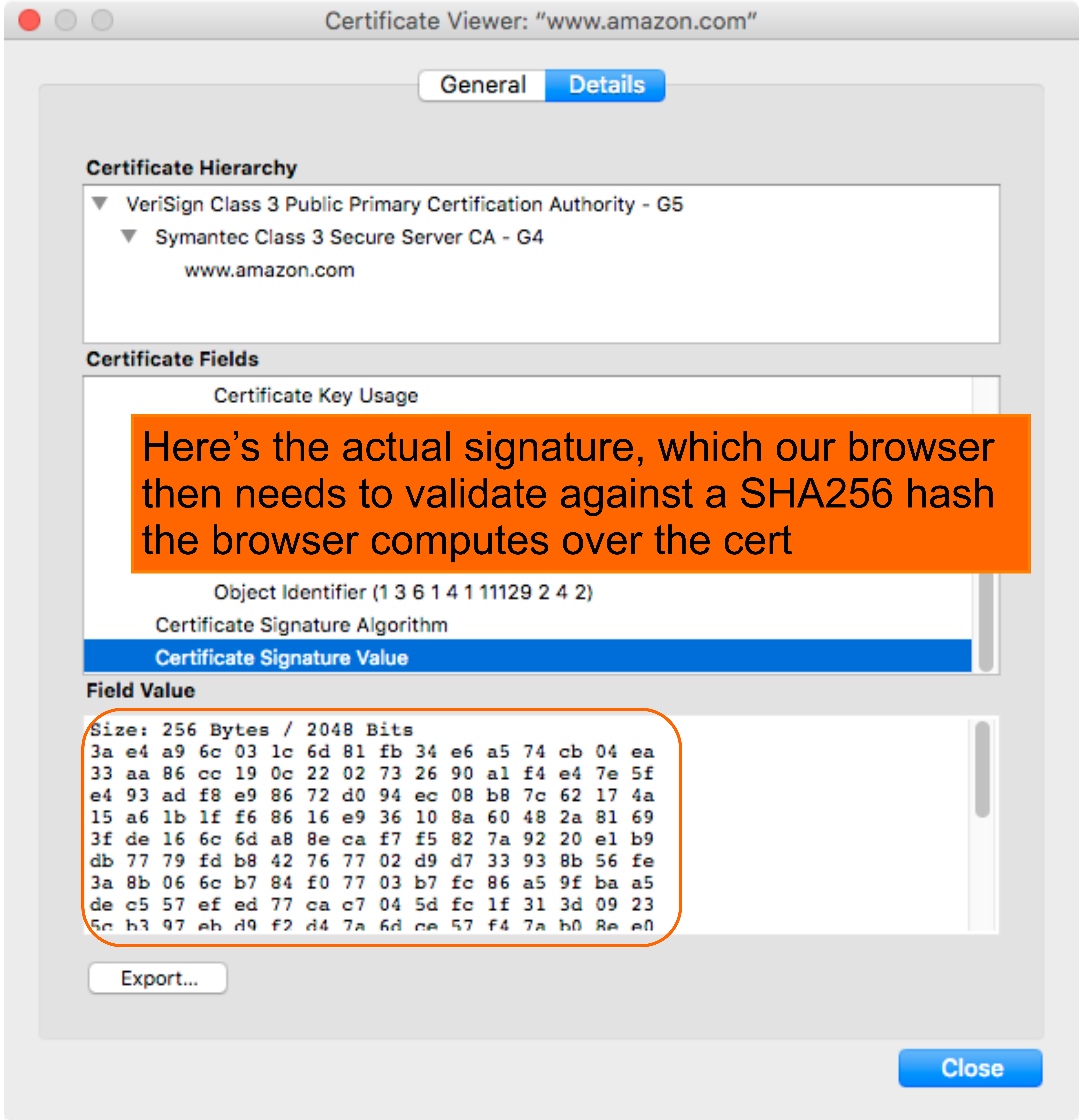

Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Certificate Subject Alt Name
Certificate Basic Constraints
Certificate Key Usage
Extended Key Usage
Certificate Policies
Field Value
DNS Name: amazon.com
DNS Name: amzn.com
DNS Name: uedata.amazon.com
DNS Name: us.amazon.com
DNS Name: www.amazon.com
DNS Name: www.amzn.com
DNS Name: corporate.amazon.com
DNS Name: buybox.amazon.com
DNS Name: iphone.amazon.com
Export...
This cert is valid for associating with any of these DNS names
Close



Our browser will only honor this cert if the URL we're accessing uses one of those domains


Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Certificate Subject Alt Name
Certificate Basic Constraints
Certificate Key Usage
Extended Key Usage
Certificate Policies
Field Value
Critical
Signing
Key Encipherment
The key can be used for both encryption and digital signatures
Export...
Close

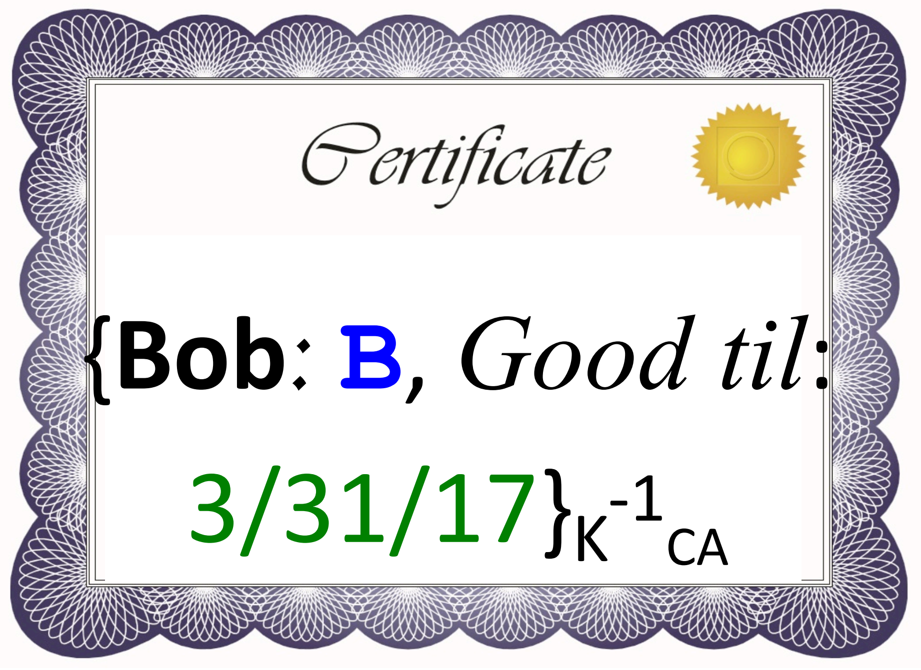

Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Certificate Key Usage
Extended Key Usage
Certificate Policies
Certificate Authority Key Identifier
CRL Distribution Points
Authority Information Access
Object Identifier (1 3 6 1 4 1 11129 2 4 2)
Certificate Signature Algorithm
Certificate Signature Value
Field Value
PKCS #1 SHA-256 With RSA Encryption
The CA has signed a SHA-256 hash of this cert using RSA
Export...
Close



Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Certificate Key Usage
Here's the actual signature, which our browser then needs to validate against a SHA256 hash the browser computes over the cert
Object Identifier (1 3 6 1 4 1 11129 2 4 2)
Certificate Signature Algorithm
Certificate Signature Value
Field Value
Size: 256 Bytes / 2048 Bits
3a e4 a9 6c 03 1c 6d 81 fb 34 e6 a5 74 cb 04 ea
33 aa 86 cc 19 0c 22 02 73 26 90 a1 f4 e4 7e 5f
e4 93 ad f8 e9 86 72 d0 94 ec 08 b8 7c 62 17 4a
15 a6 1b 1f f6 86 16 e9 36 10 8a 60 48 2a 81 69
3f de 16 6c 6d a8 8e ca f7 f5 82 7a 92 20 e1 b9
db 77 79 fd b8 42 76 77 02 d9 d7 33 93 8b 56 fe
3a 8b 06 6c b7 84 f0 77 03 b7 fc 86 a5 9f ba a5
de c5 57 ef ed 77 ca c7 04 5d fc 1f 31 3d 09 23
Export...
Close

Revocation

- What do we do if a CA screws up and issues a cert in Bob's name to Mallory?
  - E.g. Verisign issued a `Microsoft.com` cert to a *Random Joe*
  - (Related problem: Bob realizes b has been stolen)
- *How do we recover from the error?*
- Approach #1: expiration dates
  - Mitigates possible damage
  - But adds management burden
  - Also slow response to issue



Revocation, con't

- Approach #2: CA announce revoked certs
  - Users periodically download *cert revocation list* (CRL)

Revocation, con't

- Approach #2: CA announce revoked certs
  - Users periodically download *cert revocation list* (CRL)
- Issues?
  - Lists can get large
  - Need to *authenticate the list* itself – how?

Revocation, con’t

- Approach #2: CA announce revoked certs
  - Users periodically download *cert revocation list* (CRL)
- Issues?
  - Lists can get large
  - Need to authenticate the list itself – how? Sign it!
  - Mallory can exploit download lag
  - What does Alice do if can’t reach CA for download?
    1. Assume all certs are invalid (*fail-safe defaults*)
       - Wow, what an unhappy failure mode!
    2. Use old list: widens exploitation window if Mallory can “DoS” CA (DoS = denial-of-service)

Revocation, con’t

- Approach #3: CA provides service to query
  - OCSP: *Online Certificate Status Protocol*

Revocation, con't

- Approach #3: CA provides service to query
  - OCSP: *Online Certificate Status Protocol*

- Issues?
  - Can't be used if Alice doesn't have connectivity to CA
  - CA learns that Alice talks to Bob
    - **OCSP Stapling:** If enabled, Bob/server queries the CA periodically and provides the CA-signed proof that cert is not revoked.
  - CA outages ⇒ big headaches
    - What does Alice do if OCSP inaccessible??

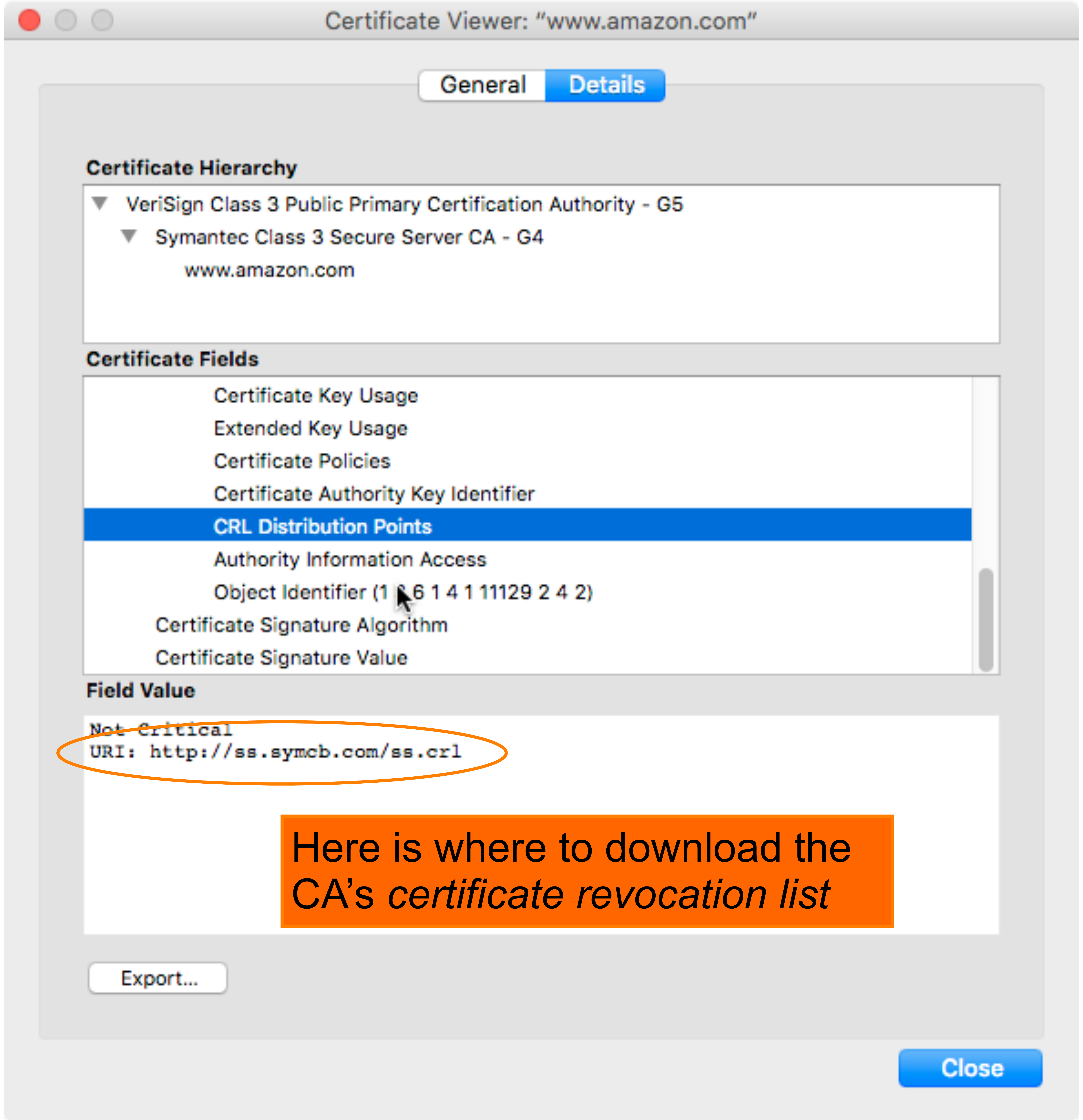
Certificate Pinning

- **One final approach:**
  - Clients "remember" a previously seen certificate and only use that, so even if CA issues an errant certificate to the wrong person, clients won't accept it.
  - This is certificate pinning.
  - When first connecting to a TLS server, save the cert's public key (probably a hash)

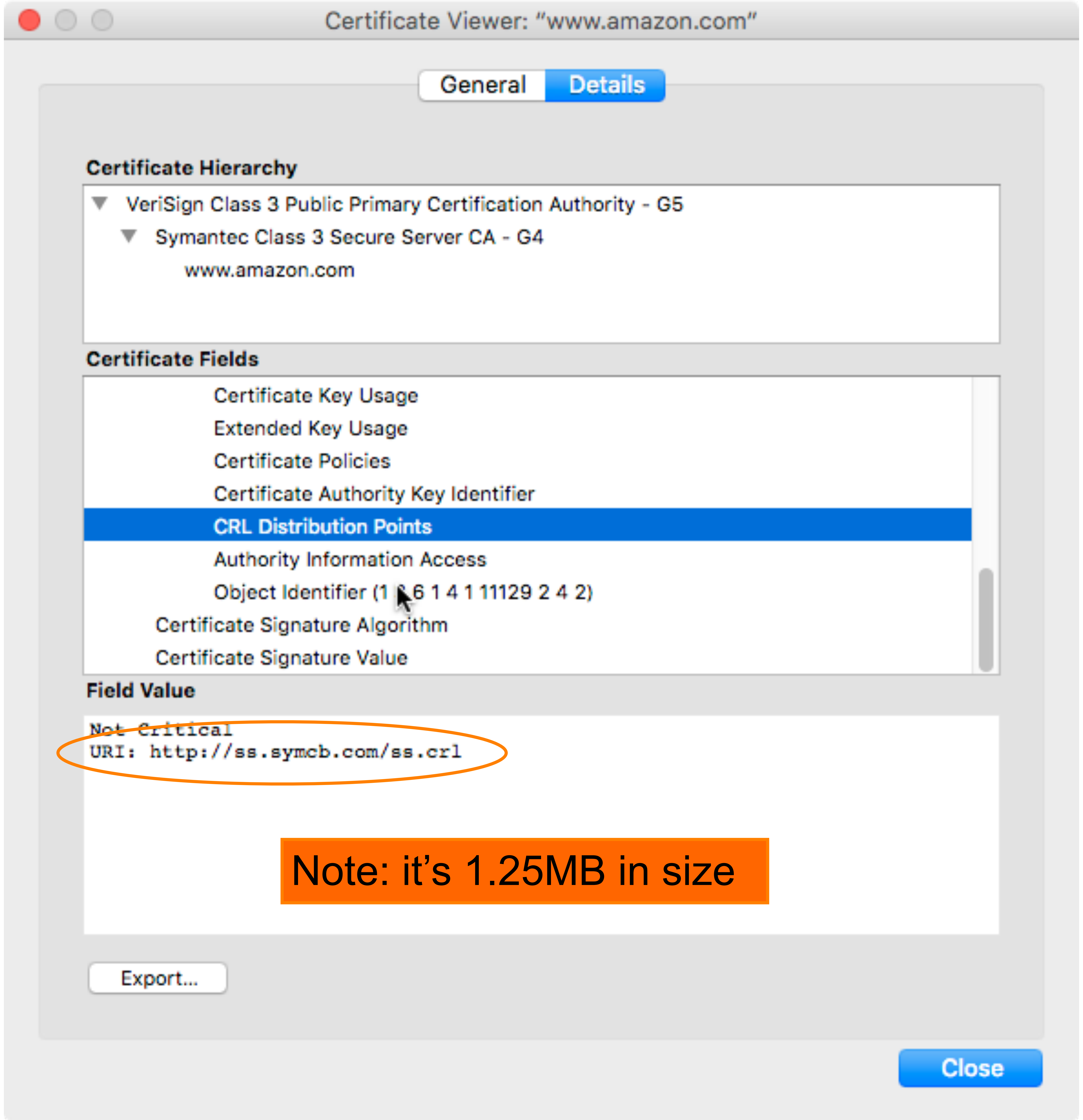
Certificate Pinning

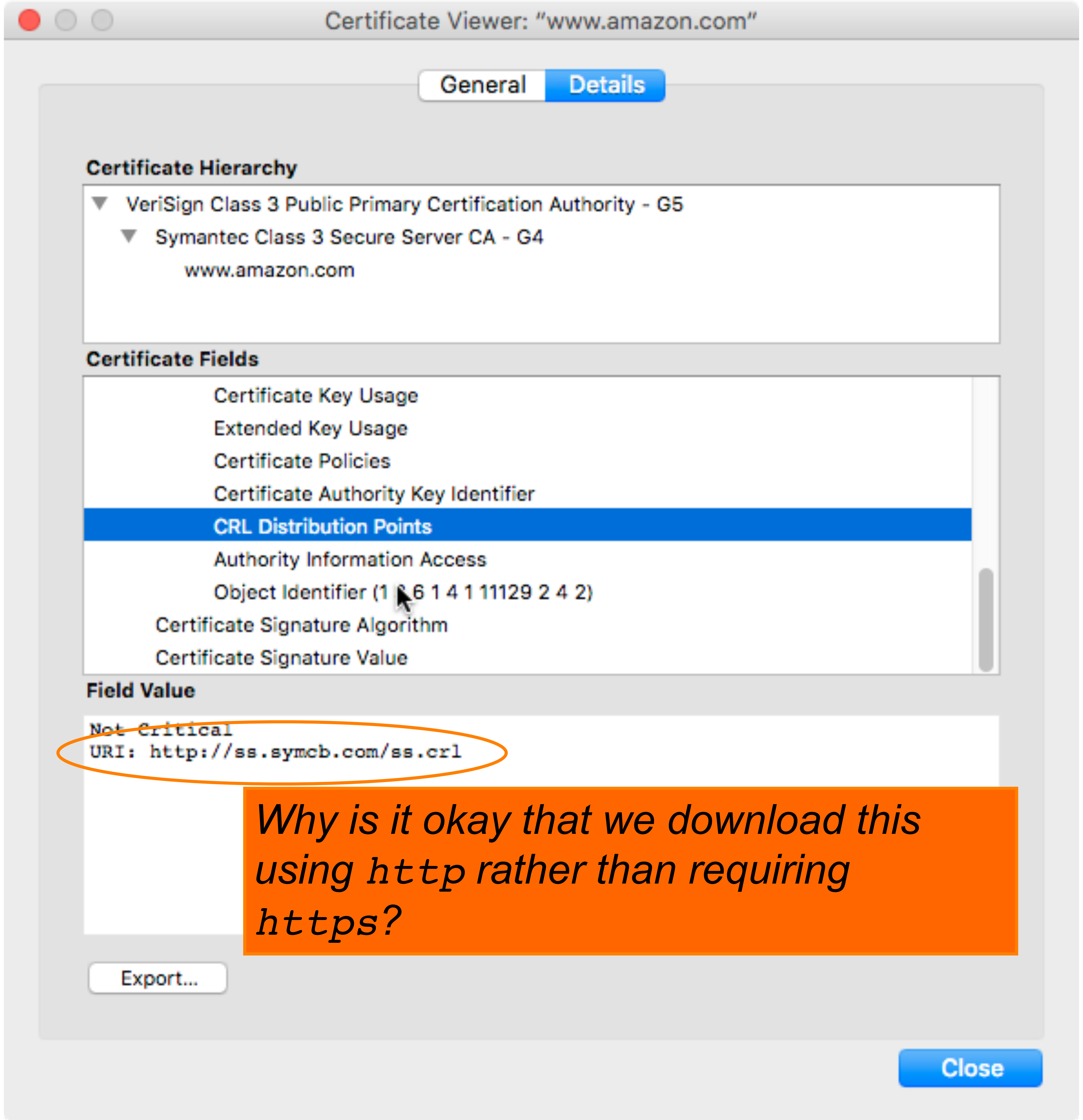
- **Issues?**
  - Must trust the first connection...
  - If your cert does need to change...that can cause big problems...

  - While considered best practice for many years, no longer considered so, given the practical issues and deployment headaches.

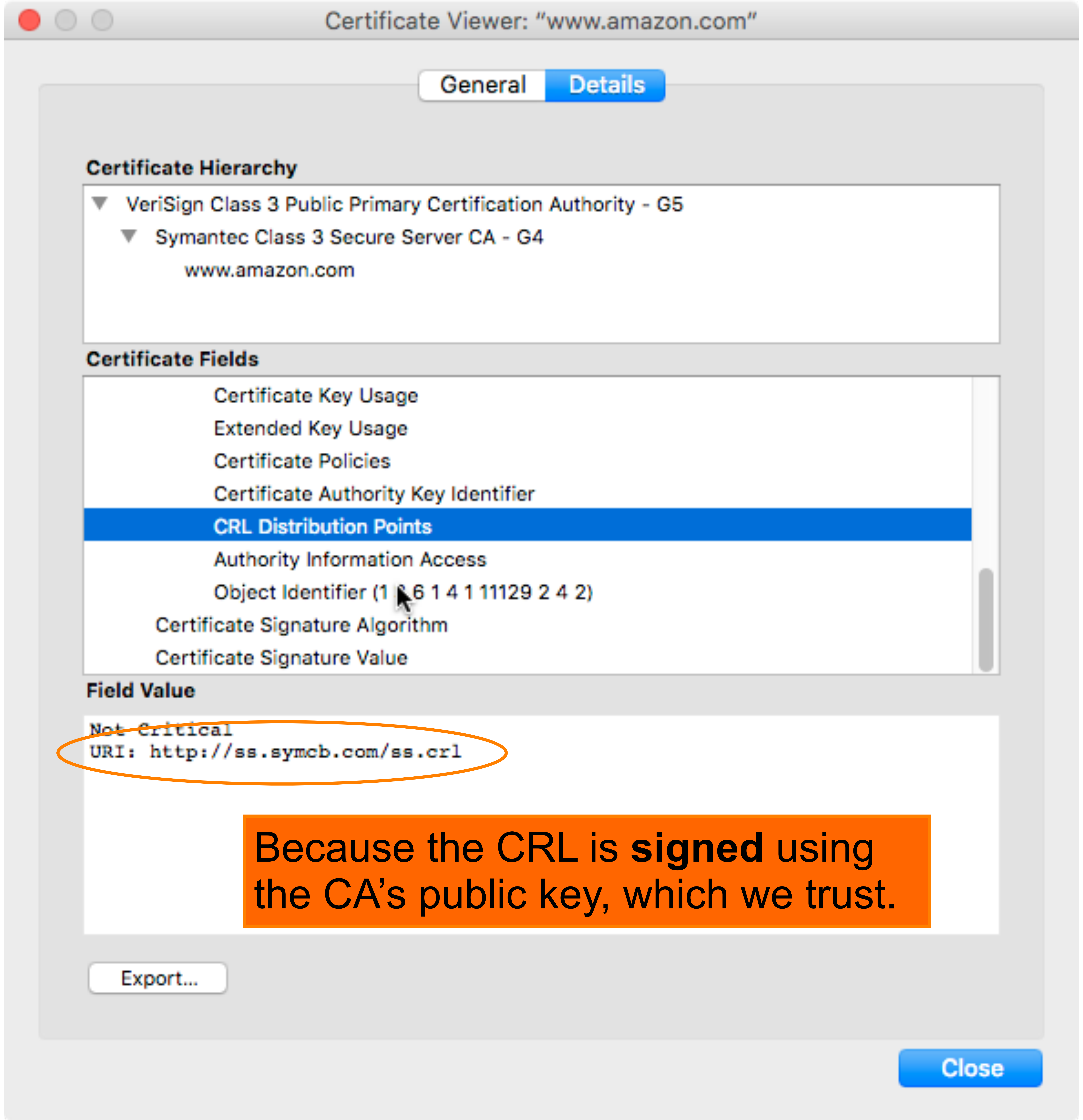


Here is where to download the CA's *certificate revocation list*

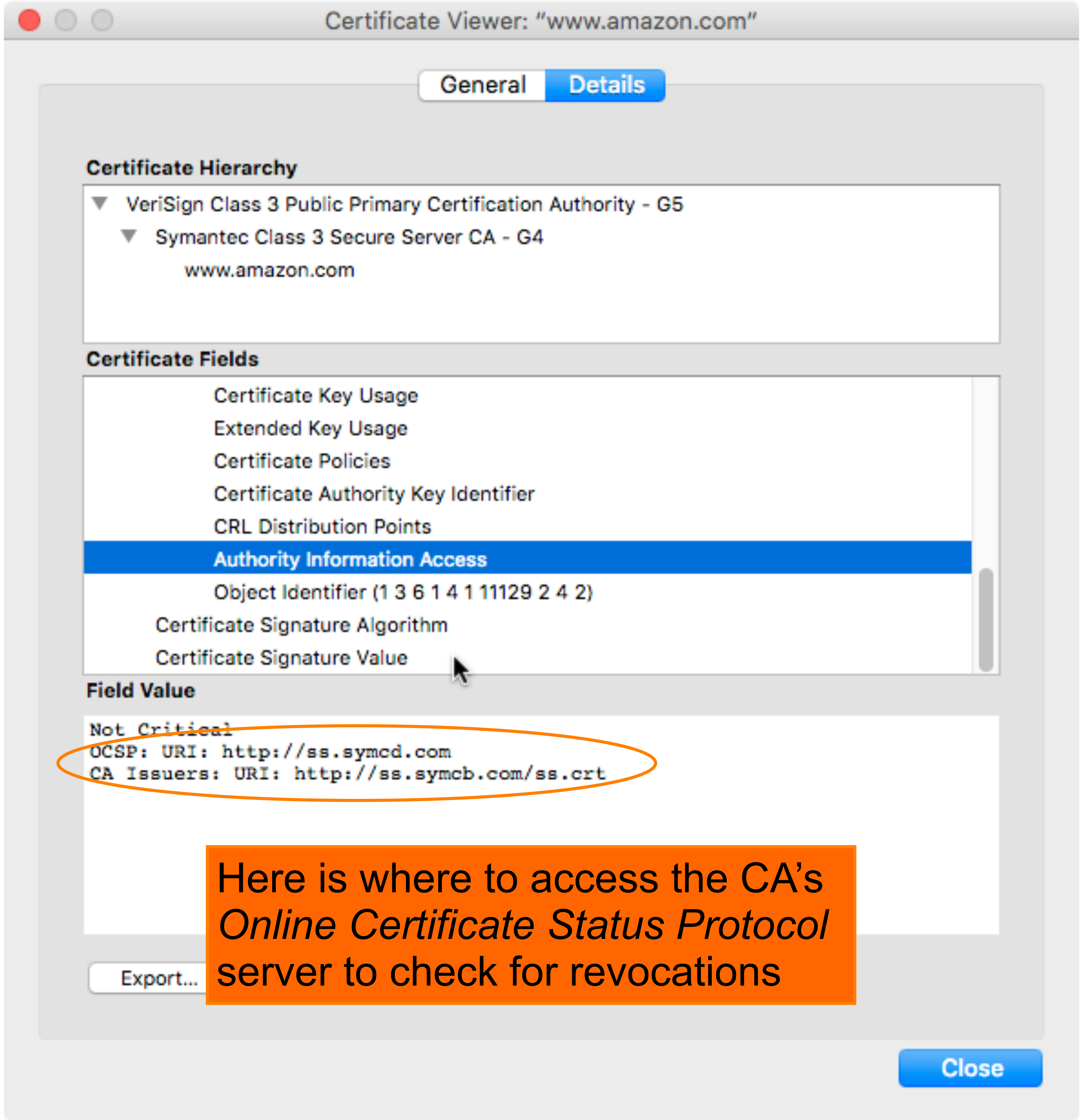

Certificate Viewer: "www.amazon.com"
General
Details
Certificate Hierarchy
VeriSign Class 3 Public Primary Certification Authority - G5
Symantec Class 3 Secure Server CA - G4
www.amazon.com
Certificate Fields
Certificate Key Usage
Extended Key Usage
Certificate Policies
Certificate Authority Key Identifier
CRL Distribution Points
Authority Information Access
Object Identifier (1 3 6 1 4 1 11129 2 4 2)
Certificate Signature Algorithm
Certificate Signature Value
Field Value
Not Critical
URI: http://ss.symcb.com/ss.crl
Note: it's 1.25MB in size
Export...
Close



Why is it okay that we download this using `http` rather than requiring `https`?



Because the CRL is **signed** using the CA's public key, which we trust.



Here is where to access the CA's *Online Certificate Status Protocol* server to check for revocations

Validating Amazon's Identity

- Browser compares domain *name* in cert w/ URL
- Browser accesses *separate* cert belonging to **issuer**
  - Might be hardwired into the browser - **trusted root CAs!**
  - There could be a *chain* of these …
- Browser applies issuer's public key to verify certificate's signature
  - Validates signature on its own SHA-256 hash of Amazon's cert
- Assuming signature validates, now have high confidence it's indeed Amazon …
  - ***assuming signatory is trustworthy***

End-to-End ⇒ Powerful Protections

- Attacker runs a sniffer to capture our WiFi session?
  - (maybe by buying a cup of coffee to get the password)
  - But: encrypted application data is unreadable
    - No problem!
- DNS cache poisoning?
  - Client goes to wrong server
  - But: detects impersonation since attacker lacks valid cert
    - No problem!
- Attacker hijacks our connection, injects new traffic
  - But: data receiver rejects it due to failed integrity check
    - No problem!

Powerful Protections, con't

- DHCP spoofing?
  - Client goes to wrong server
  - But: they can't read; we detects impersonation
    - No problem!
- Attacker manipulates BGP routing to run us by an eavesdropper or take us to the wrong server?
  - But: they can't read; we detect impersonation
    - No problem!
- Attacker slips in as a Man In The Middle?
  - But: they can't read, they can't inject
  - They can't even replay previous encrypted traffic
  - **No problem!**

How do we know to use TLS?

- How do we know if example.com is over HTTP or HTTPS?
  - Method 1: Could try HTTPS, and if we don't get any response or an error, switch to HTTP.
  - Method 2: Could try HTTP, and if HTTPS is available, the server's HTTP response can redirect us to HTTPS

How do we know to use TLS?

- What if there's a network MITM attacker??
  - W/ Method 1: MITM can block/drop HTTPS attempt, causing us to switch to HTTP
  - W/ Method 2: Block the HTTP redirection response so we don't know HTTPS is available

  - In both cases, then MITM attacker can pretend to be the HTTP server. We will send our data to the attacker over HTTP and the attacker can relay data to the real HTTPS website.
  - [SSL-Stripping attack]()

How do we know to use TLS?

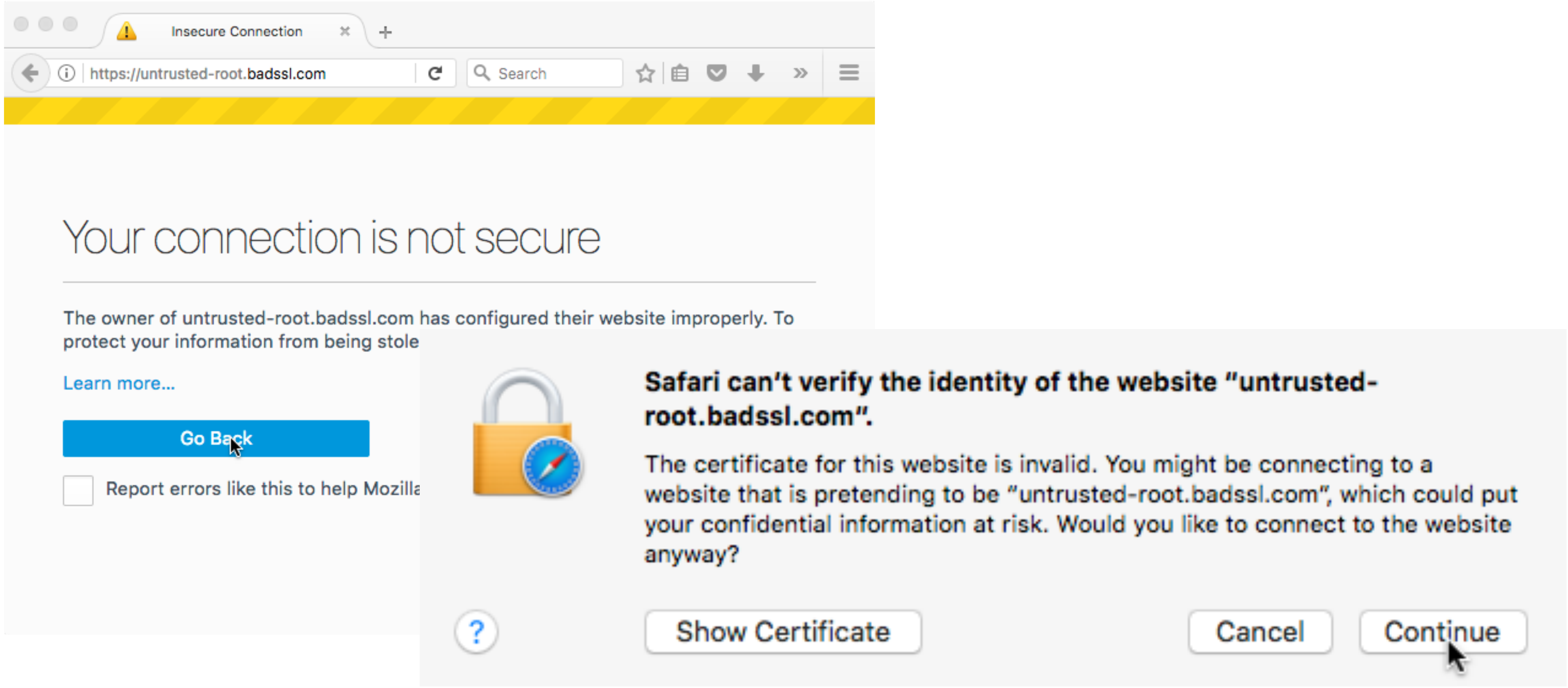
- Solutions?
  - **HTTP Strict Transport Security (HSTS):** Server tells client that it always uses HTTPS, so client knows to use HTTPS moving forward.
    - Done using an HTTP header

```
Strict-Transport-Security: max-age=31536000;
```

    - Assumes secure first connection though...
  - **(Proposed) HTTPS DNS Resource Records:** Add a new DNS RR indicating HTTPS is available, which clients can query for.
    - Unless DNSSEC used, may not help in MITM case

Validating Amazon's Identity, con't

- What if browser can't find a cert for the issuer or the cert doesn't pass?



Validating Amazon’s Identity, con’t

- What if browser can’t find a cert for the issuer or the cert doesn't pass?
  - Then warns the user that site has not been verified
    - Note, can still proceed, just without authentication
- Q: Which end-to-end security properties do we lose if we incorrectly trust that the site is whom we think?
- A: **All of them!**
  - Goodbye confidentiality, integrity, authentication
  - Attacker can read everything, modify, impersonate

TLS Limitations

- Properly used, TLS provides powerful end-to-end protections
- So why not use it for *everything*??
- Issues:
  - Cost of public-key crypto
    - Takes non-trivial CPU processing (but today a minor issue)
    - Note: *symmetric* key crypto on modern hardware is non-issue
  - Hassle of buying/maintaining certs (fairly minor)


https://letsencrypt.org
let's encrypt
LINUX FOUNDATION COLLABORATIVE PROJECTS
Let's Encrypt
Documentation
Get Help
Donate
About Us
Let's Encrypt is a free, automated, and open Certificate Authority.
Get Started
Donate
FROM OUR BLOG
Mar 23, 2017
OVH Renews Platinum Sponsorship of Let's Encrypt
We're pleased to announce that OVH has renewed their support for Let's Encrypt as a Platinum sponsor for the next three years.
Read more
MAJOR SPONSORS
moz://a
Akamai
CISCO
EFF
OVH
chrome
IdenTrust
gemalto
FORD FOUNDATION
facebook
AUTOMATTIC
shopify

TLS Limitations

- Properly used, TLS provides powerful end-to-end protections
- So why not use it for *everything*??
- Issues:
  - Cost of public-key crypto
    - Takes non-trivial CPU processing (but today a minor issue)
    - Note: *symmetric* key crypto on modern hardware is non-issue
  - Hassle of buying/maintaining certs (fairly minor)
  - **Latency**: extra round trips $\Rightarrow$ pages take longer to load

TLS Limitations, con't

- Problems that TLS does **not** take care of ?
- TCP-level attacks
  - RST injection
    - (but does protect against data injection!)
- Denial of Service Attacks
- Application-level vulnerabilities:
  - SQL injection / XSS / server-side coding/logic flaws
  - Browser coding/logic flaws
  - User flaws
    - Weak passwords
    - Phishing
  - HTTP server vulnerabilities