Lecture 21 – Transport Layer Security (TLS)

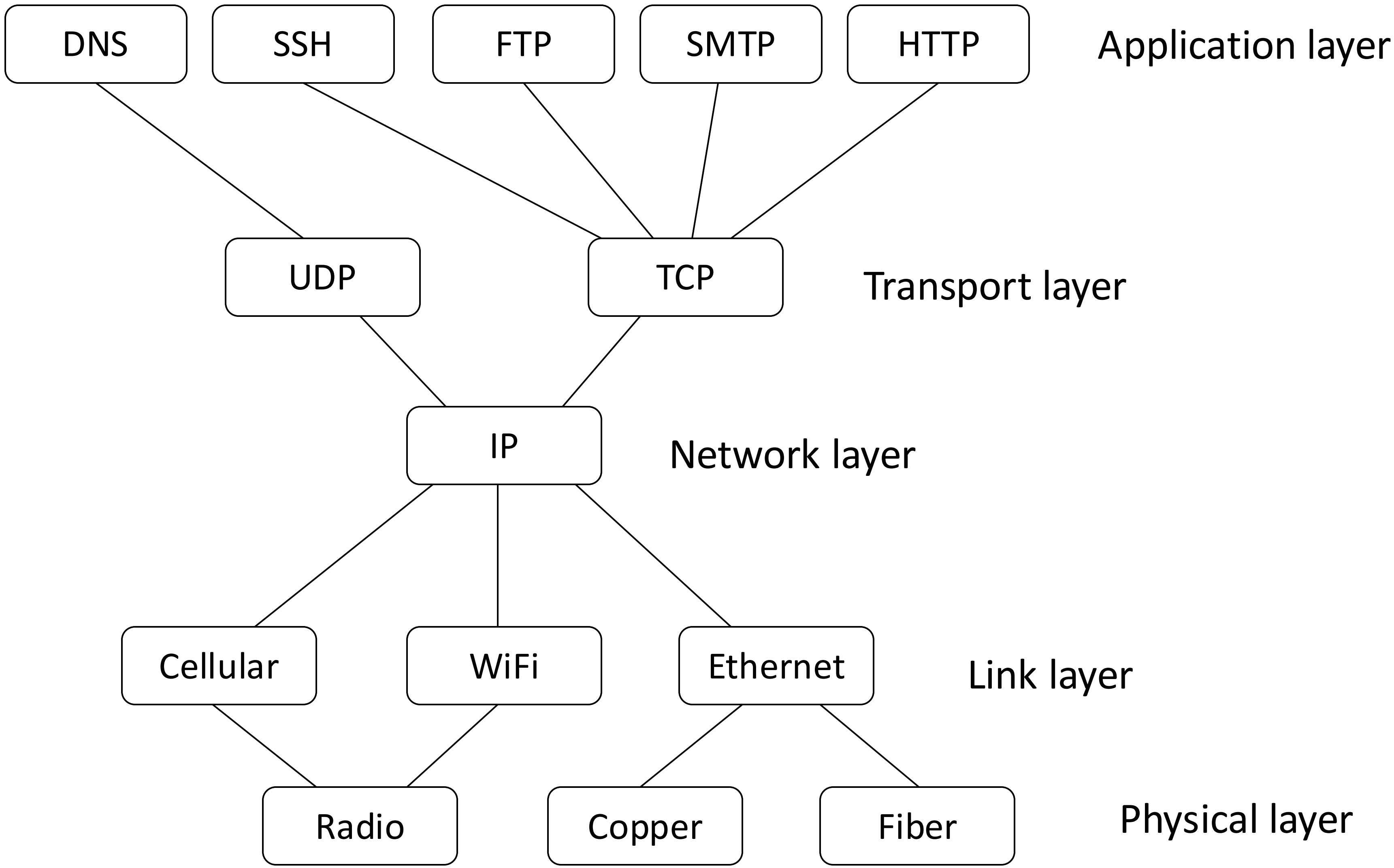
University of Illinois

ECE 422/CS 461

Learning Objectives

- Understand the basics of the TLS ciphersuite
- Consider how TLS addresses the (in)security of transport-layer protocols
- Evaluate the limitations of the Certificate Authority ecosystem

Layering of Protocols



TCP Security Properties

	Passive	Off-Path	MitM
Availability	—	✗	✗
Confidentiality	✗	—	✗
Integrity	—	—	✗
Authenticity	—	✓	✗

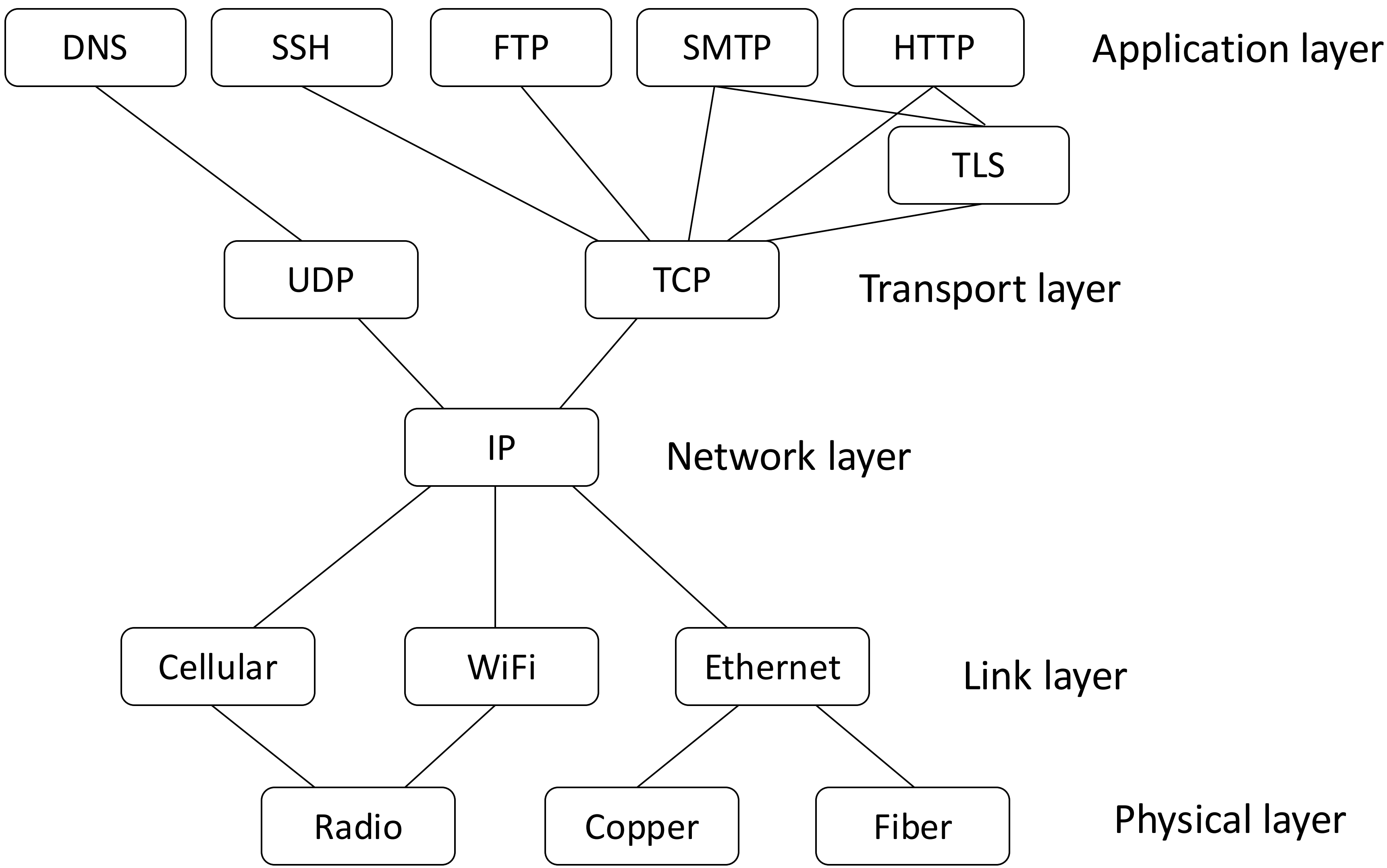
- Against off-path attackers
  - TCP is vulnerable to denial of service attacks
  - TCP (with random initial sequence number) has reasonable authenticity against off-path attackers!

TCP Security Properties

	Passive	Off-Path	MitM
Availability	–	✗	✗
Confidentiality	✗	–	✗
Integrity	–	–	✗
Authenticity	–	✓	✗

- Today: we can use cryptography to achieve confidentiality and integrity/authenticity even against MitM attackers

Layering of Protocols



History of SSL/TLS

- Secure Sockets Layer (SSL) developed by Netscape for secure web sessions (1995)
- Transport Layer Security (TLS) evolved from SSL and replaced it (1999)
  - Current version of TLS is 1.3 (2018)
  - Most widely used is TLS 1.2 (2008)

Cryptography Toolbox

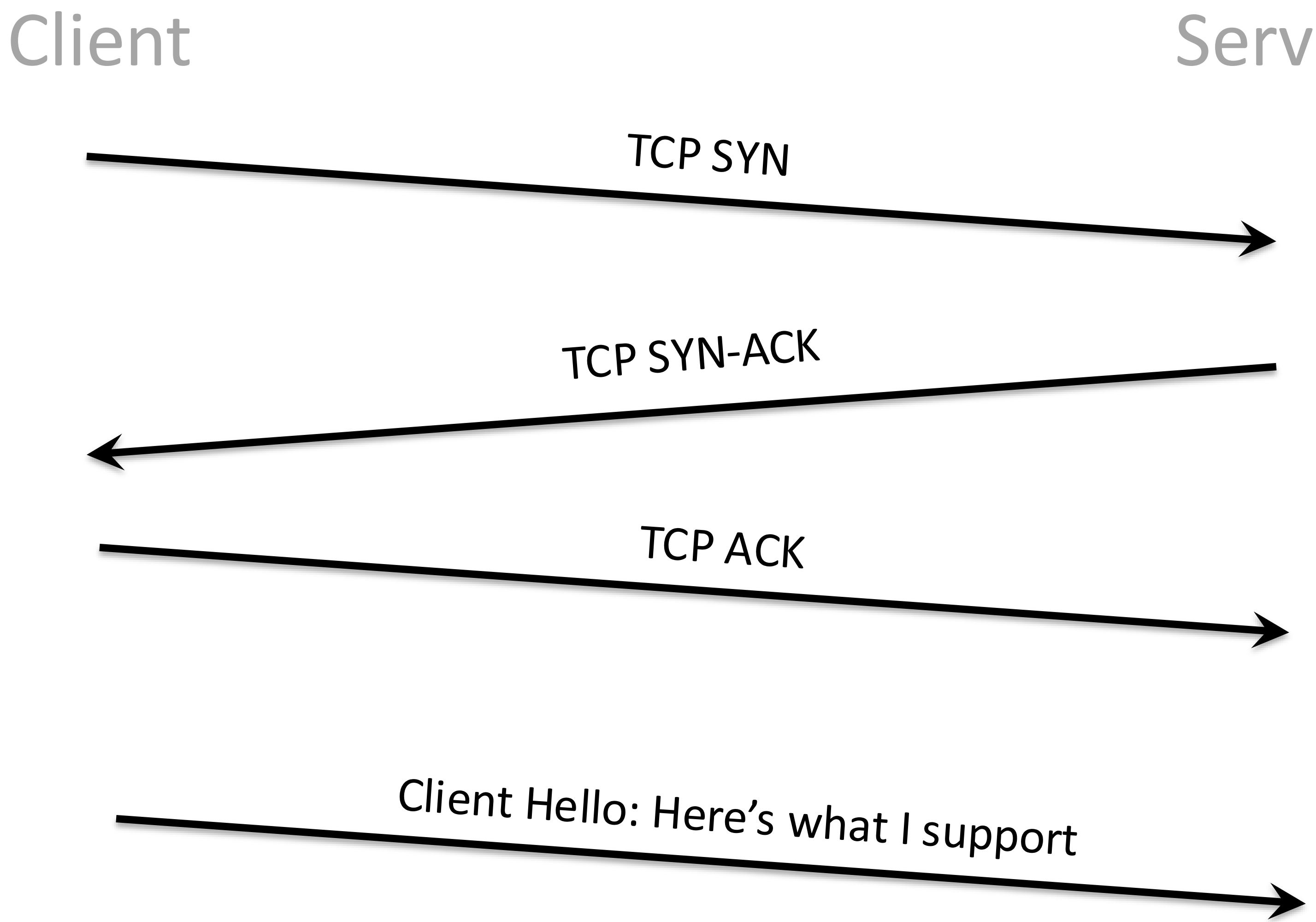
- Cryptographic hash functions
- Message Authentication Codes
- Symmetric encryption
- Asymmetric encryption
- Key exchange
- Digital signatures

Symmetric

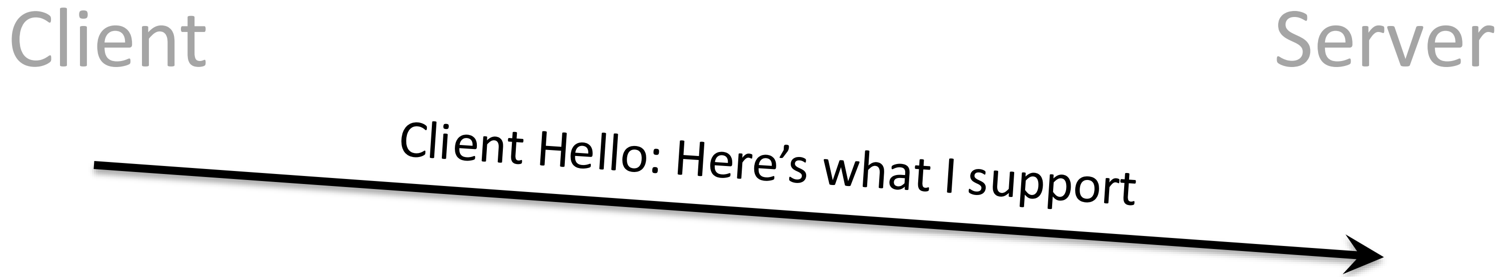
Asymmetric

Basic Idea

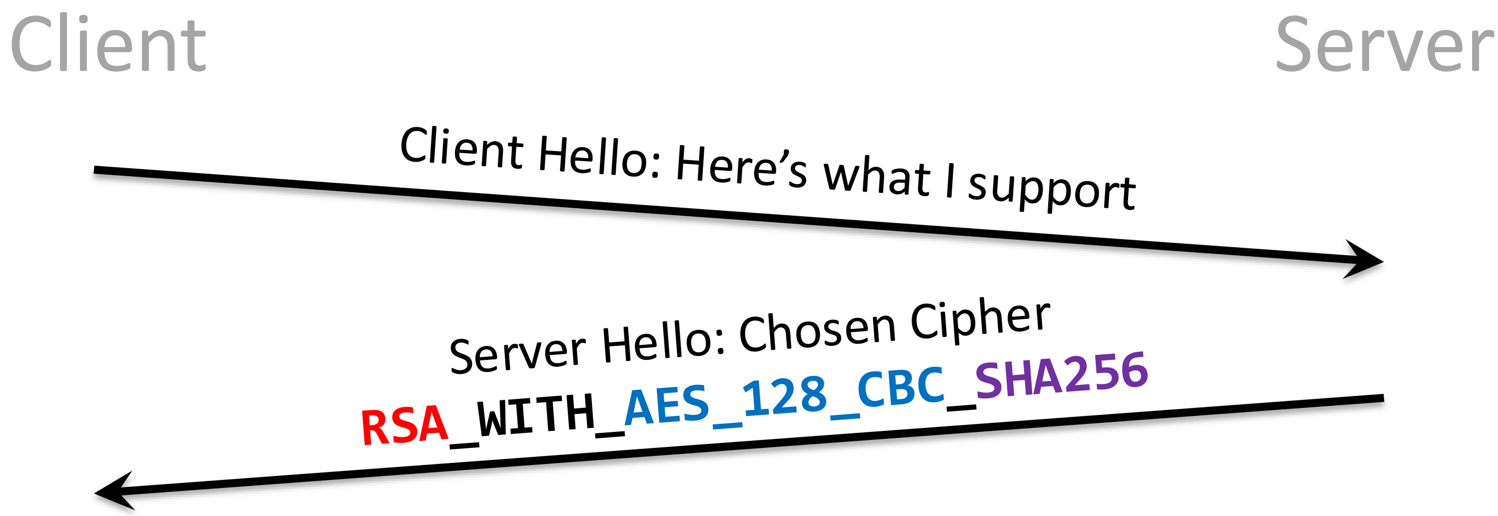
- Use key exchange or asymmetric encryption to establish a shared secret key
- Use the shared secret key for symmetric encryption and message authentication



Illustrated TLS connection with explanations:
`https://tls.ulfheim.net/`



Client Hello includes a random nonce (called client random).

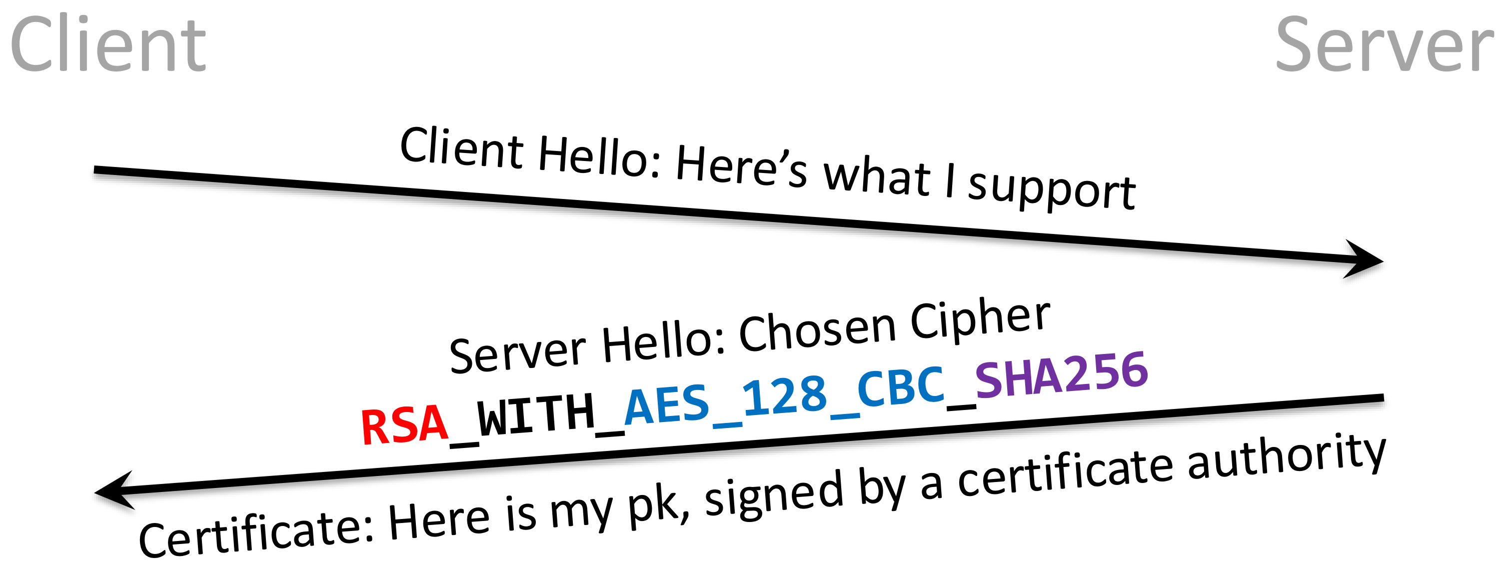


RSA_WITH_AES_128_CBC_SHA256

Asymmetric Encryption

Symmetric Encryption

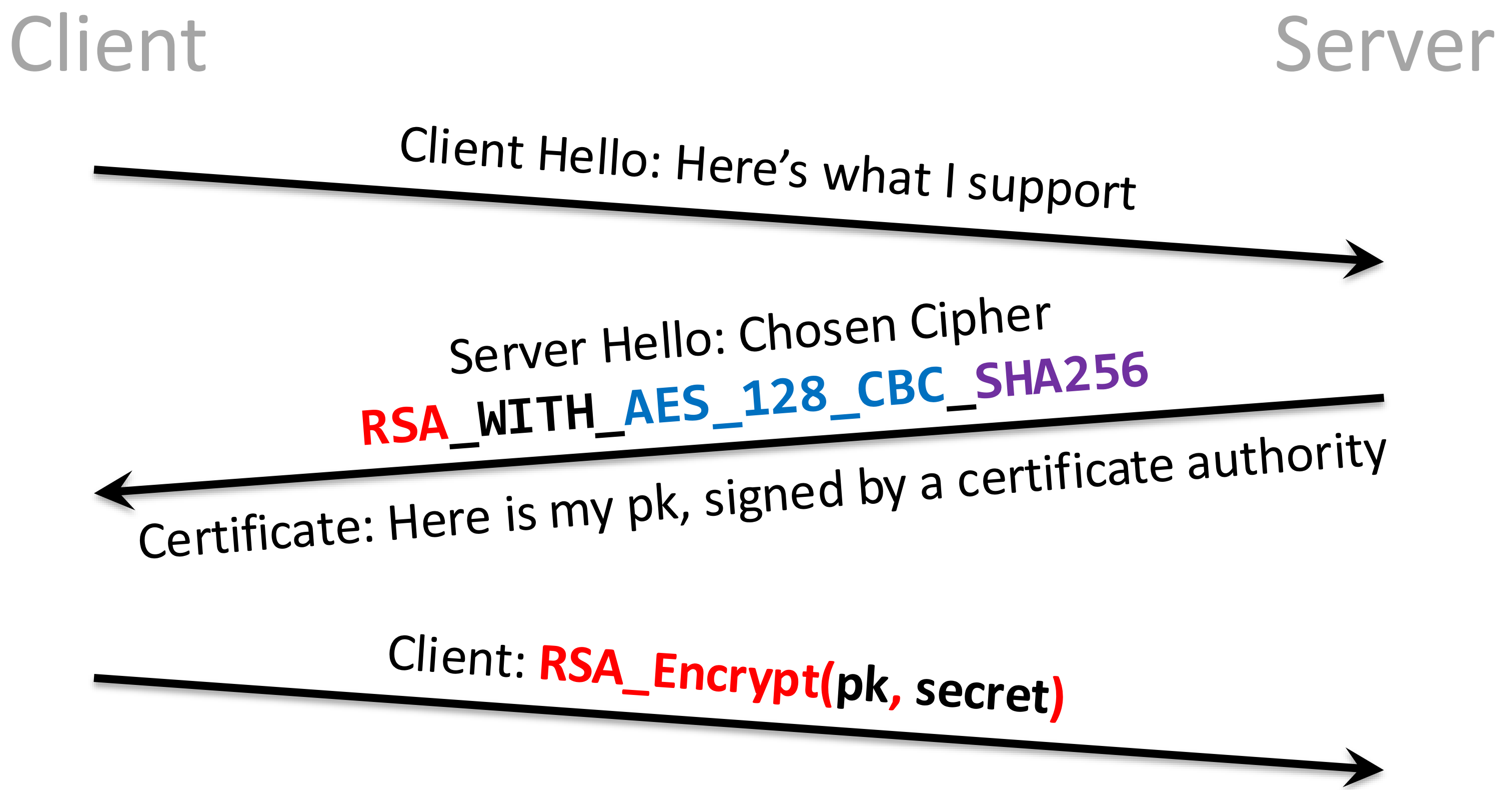
Hash-based Message Authentication Code (HMAC)



Server Hello includes a random nonce (called server random)

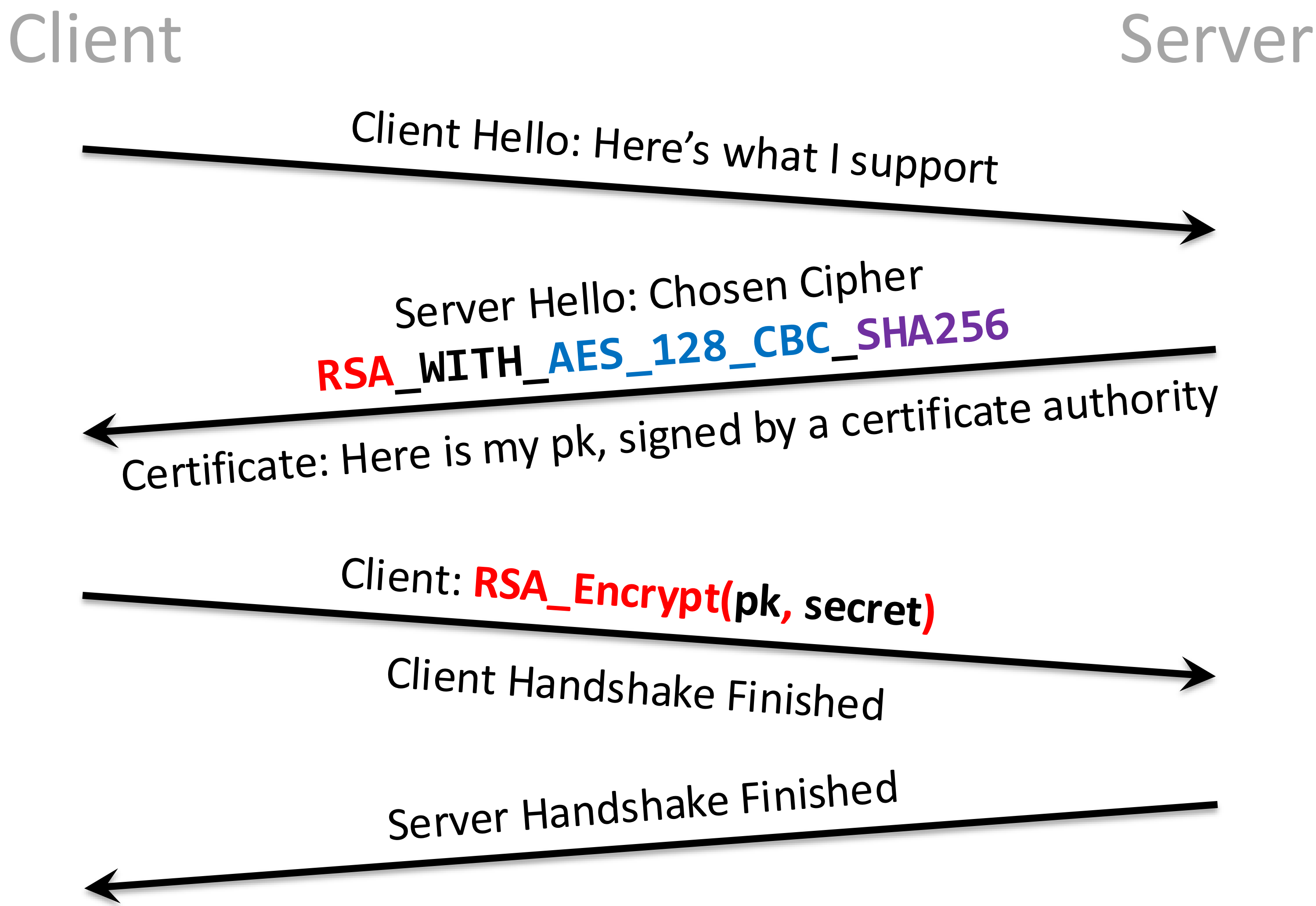
Server Hello includes a **certificate** for its public key.

Client verifies the certificate.

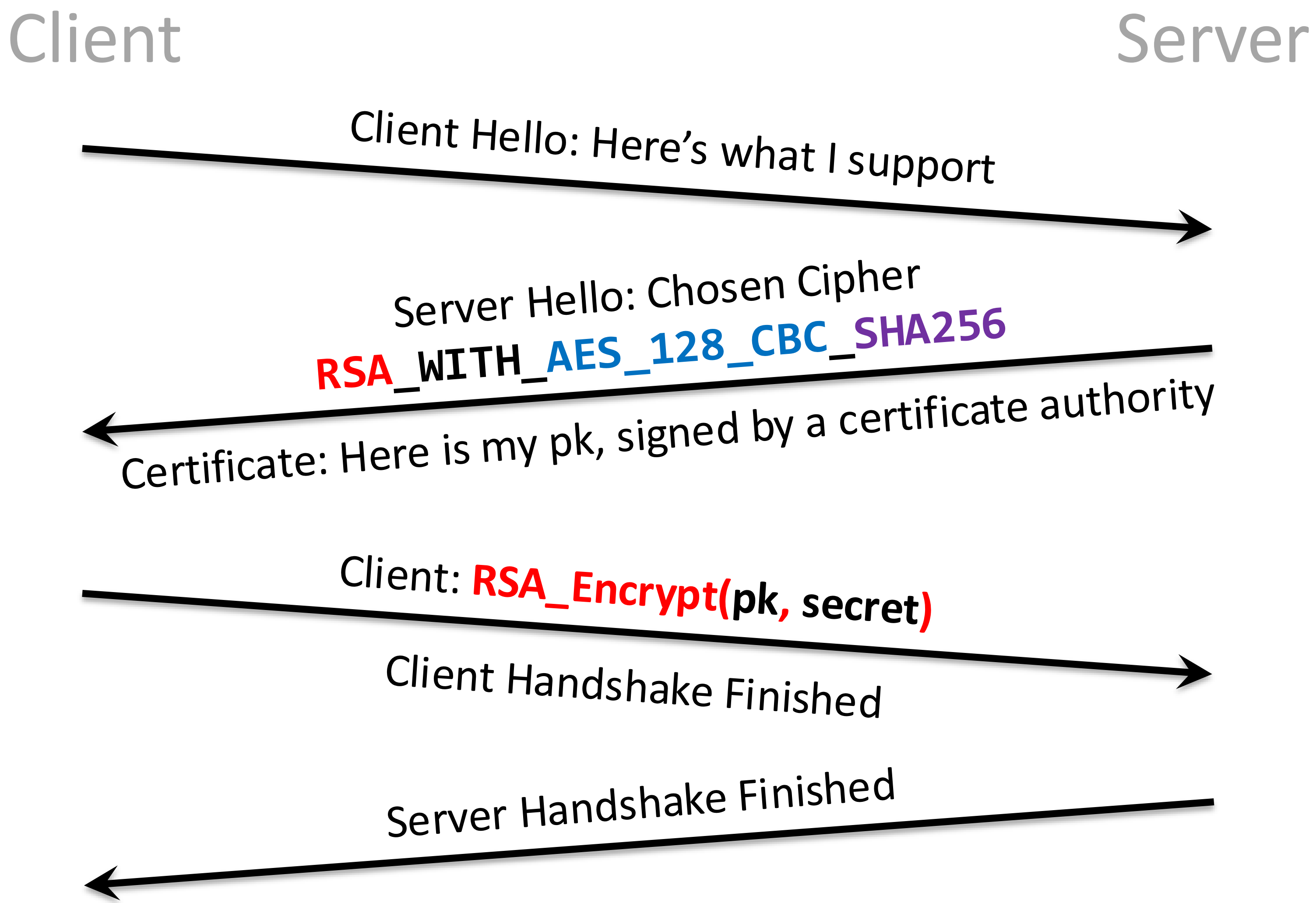


Only the server has the secret key corresponding to pk, so only the server can decrypt the secret.

Session key = Hash(secret, client random, server random)



After Handshake Finish messages, client and server start using AES_128_CBC for confidentiality and SHA256 HMAC for integrity.

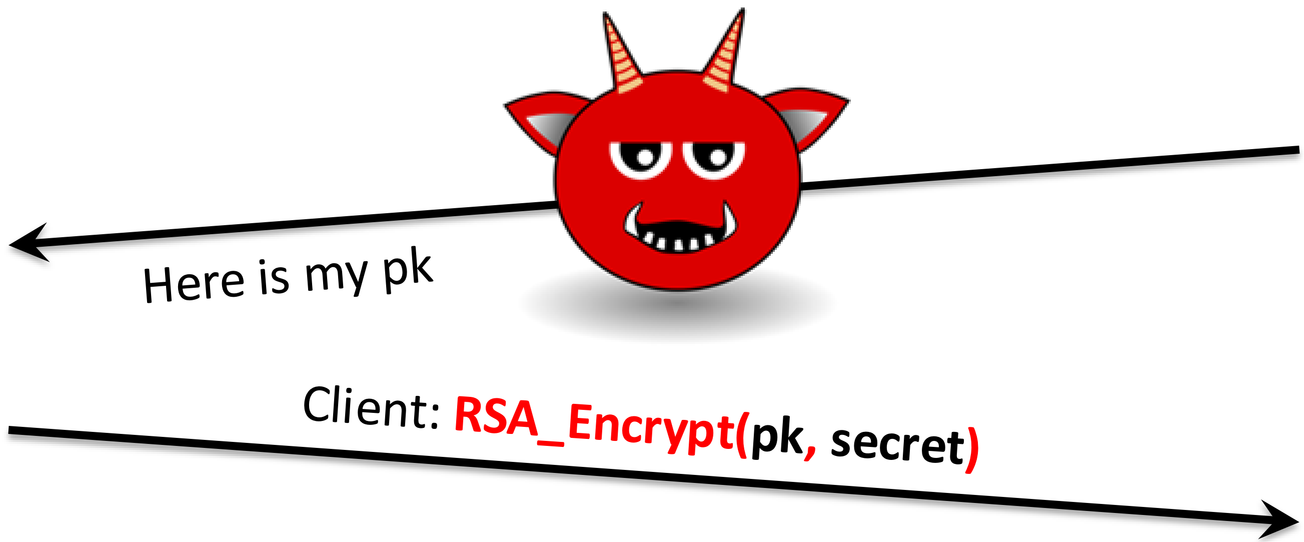


The Handshake Finished message contains a HMAC of the entire interaction so far. Why?

To prevent cipher suit downgrade attack.

Man-in-the-Middle Attack

- Recall that asymmetric encryption is secure only if the client knows the **authentic** public key of the server



Certificates

- A certificate consists of
  - Identity of server (e.g., its domain name)
  - A public key belonging to the identity
  - Some restrictions (e.g., expiration date)
  - A digital signature by a **Certificate Authority (CA)** on the above information

- Essentially, the CA is vouching for the server's public key

Recall Digital Signatures

- Interface
  - KeyGen() → (vk, sk)
    - A private signing key and a public verification key
  - Sign(sk, m) → σ (called a signature)
  - Verify(vk, m, σ) → True/False
  - Very common to sign and verify Hash(m)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0f:77:30:d4:eb:75:d6:c4:22:1e:4b:a1:f6:16:2b:83
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com,
                CN=DigiCert High Assurance CA-3
        Validity
            Not Before: Sep  7 00:00:00 2012 GMT
            Not After : Nov 11 12:00:00 2015 GMT
        Subject: C=US, ST=California, L=La Jolla,
                 O=University of California, San Diego,
                 OU=ACT Data Center, CN=*.ucsd.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:cf:73:a9:a0:dd:69:de:98:c5:65:2d:fa:c0:dc:
                    47:ed:ff:f9:0b:16:3a:ee:e4:74:6a:de:26:37:7b:
                    ce:f7:de:3e:50:25:13:49:23:ec:c8:b3:19:5f:05:
                    9e:05:72:41:a9:f7:26:b3:d2:bd:88:37:51:e8:d5:
                    c3:01:d9:c2:15:bf:eb:87:a3:4b:80:3b:6c:f6:ce:
                    c5:78:4c:d2:b3:24:af:3d:8b:d8:ba:b9:c9:eb:16:
                    b4:83:68:06:b6:1e:96:0e:2e:1c:78:91:41:b4:8d:
                    3c:fe:2a:f5:93:ac:e5:bd:98:78:e5:db:4a:c2:88:
                    46:3a:1f:1e:07:fd:79:8a:96:c7:e9:b7:05:4d:40:
                    5d:4d:52:2c:e4:bc:6b:eb:2c:3e:09:e1:27:49:1b:
                    46:ab:53:cf:d9:df:8f:35:74:b4:40:1f:0b:7f:c1:
                    e4:ac:3d:5a:7b:98:e1:c4:fb:d1:e7:16:47:d9:ba:
                    51:28:1b:bf:77:f7:42:f2:dc:53:e2:38:18:b9:d2:
                    59:9a:e2:44:2a:cc:e5:99:60:a1:d1:dc:aa:2f:ba:
```

The issuing CA

Restrictions

Identify of the subject

Public key of the subject

```
 TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 CRL Distribution Points:
                URI:http://crl3.digicert.com/ca3-g14.crl
                URI:http://crl4.digicert.com/ca3-g14.crl

            X509v3 Certificate Policies:
                Policy: 2.16.840.1.114412.1.1
                  CPS: http://www.digicert.com/ssl-cps-repository.htm
                  User Notice:
                    Explicit Text:

            Authority Information Access:
                OCSP - URI:http://ocsp.digicert.com
                CA Issuers - URI:http://cacerts.digicert.com/DigiCertHighAssuranceCA-
3.crt

            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: sha1WithRSAEncryption
        21:9f:9b:89:0d:43:02:0e:07:cd:dd:3c:2a:7b:aa:f2:4c:f2:
        5e:f4:fa:2f:74:db:38:0e:51:5c:76:fe:36:06:d7:6d:00:b3:
        aa:3a:4a:8c:c3:86:f1:61:c6:9d:35:4d:0c:17:c9:90:2c:8f:
        db:d8:f2:2b:46:37:00:ca:92:7b:25:86:17:b4:44:92:dc:a7:
        45:bc:1c:eb:2a:35:a5:03:bb:0b:57:c2:aa:22:a9:08:60:32:
        90:99:55:9b:c7:4c:99:25:6e:07:0d:ae:21:4a:b5:01:4e:dc:
        7e:eb:dc:3f:83:18:19:e8:b5:d1:22:e8:40:a6:61:17:6d:8a:
        cc:64:a9:ab:c3:31:d4:d3:90:db:18:14:1a:d4:8a:17:dd:0a:
        c7:c8:64:68:94:49:88:0a:1b:c2:9e:74:1a:23:15:96:91:10:
        50:13:ea:88:01:c9:79:12:93:19:29:27:12:78:9d:66:10:5c:
        72:bc:a4:f5:59:07:7a:0e:0c:69:09:ab:44:d8:24:39:ec:a3:
        53:8b:1b:18:25:aa:57:9e:e6:7a:64:87:0f:e8:6b:42:1f:ad:
        d1:38:0f:44:a8:a3:31:4f:bc:e8:74:cc:50:f6:69:10:4f:db:
```

RSA Signature by the CA
(not encryption!)

Certificates

- The client needs to know the CA's public verification key to verify the signature.

- Hardcoded in the application (e.g., browser)
  – Certificates can be chained: one CA's public verification key is signed by another CA, …

  – Root of trust is provided by application, not TLS!

TLS Security Properties

	Passive	Off-Path	MitM
Availability	—	✗	✗
Confidentiality	✓	—	✓
Integrity	—	—	✓
Authenticity	—	✓	✓

- Assumption: crypto + certificate (also called public-key infrastructure, PKI)
  - More details next slide

TLS Security

- Assumptions
  - Crypto (RSA, DH, AES, SHA, …)
  - Client and server keep their secret keys safe
  - Application (browser) hardcodes the right CA public verification keys (**root of trust**)
  - **Each** CA keeps its signing key safe
  - **Each** CA issues certificates responsibly (after checking subject identity)

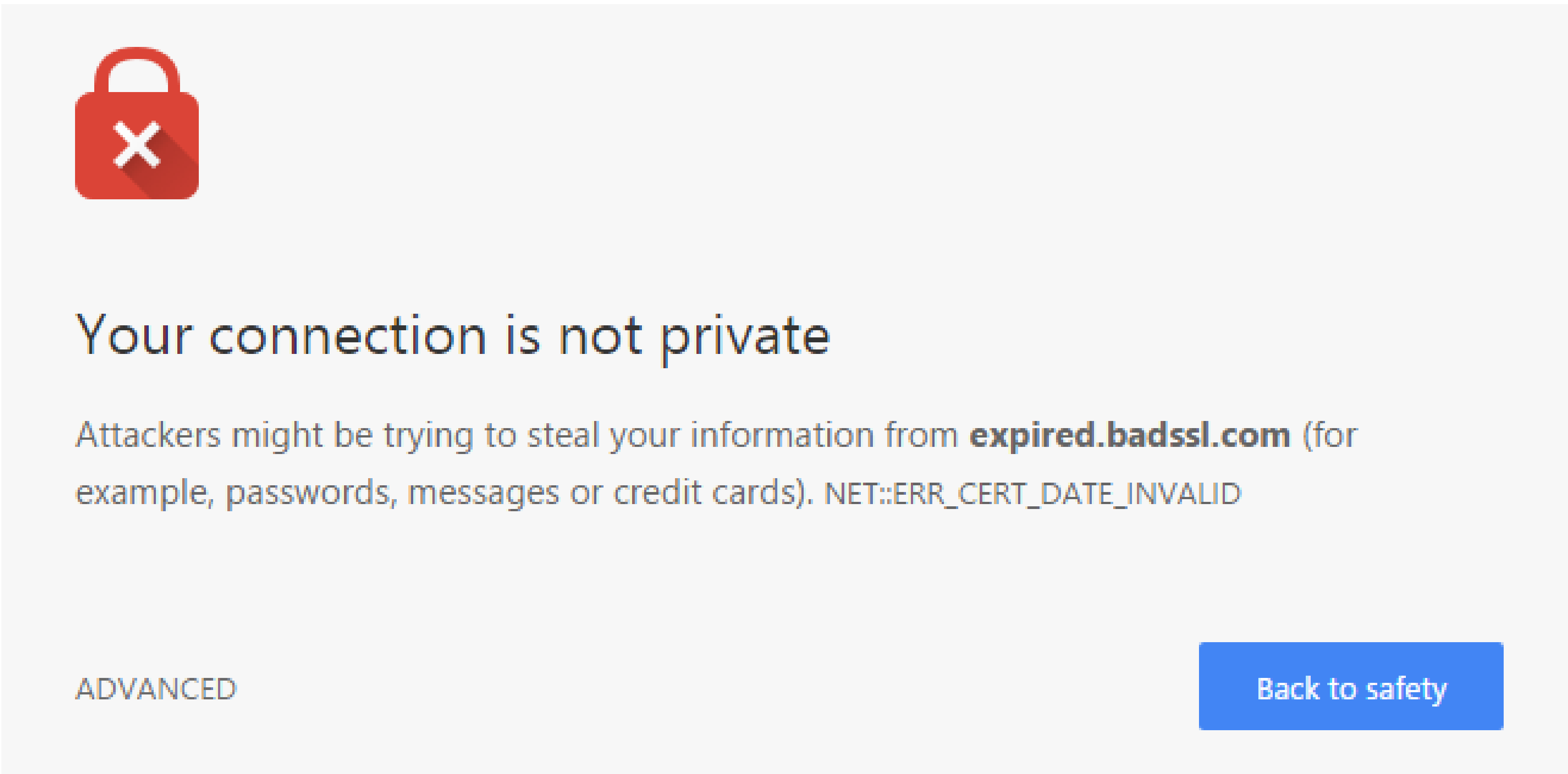
TLS Security

- Very strong security: confidentiality and integrity against even MitM attackers
- Assumptions: crypto, keys, root of trust, CAs
- No assumption on other network layers
  - Exercise: what if I connect to adversary's WiFi?
  - Exercise: what if DNS is broken?
- Can things still go wrong?

CA is a Weak Link

- Certificate Authorities are periodically in the news for high-profile compromises and blunders

Human is another Weak Link



Human is another Weak Link

From Paypal [redacted]

Subject **Your account access has been limited** 14/11/19, 5:51 am

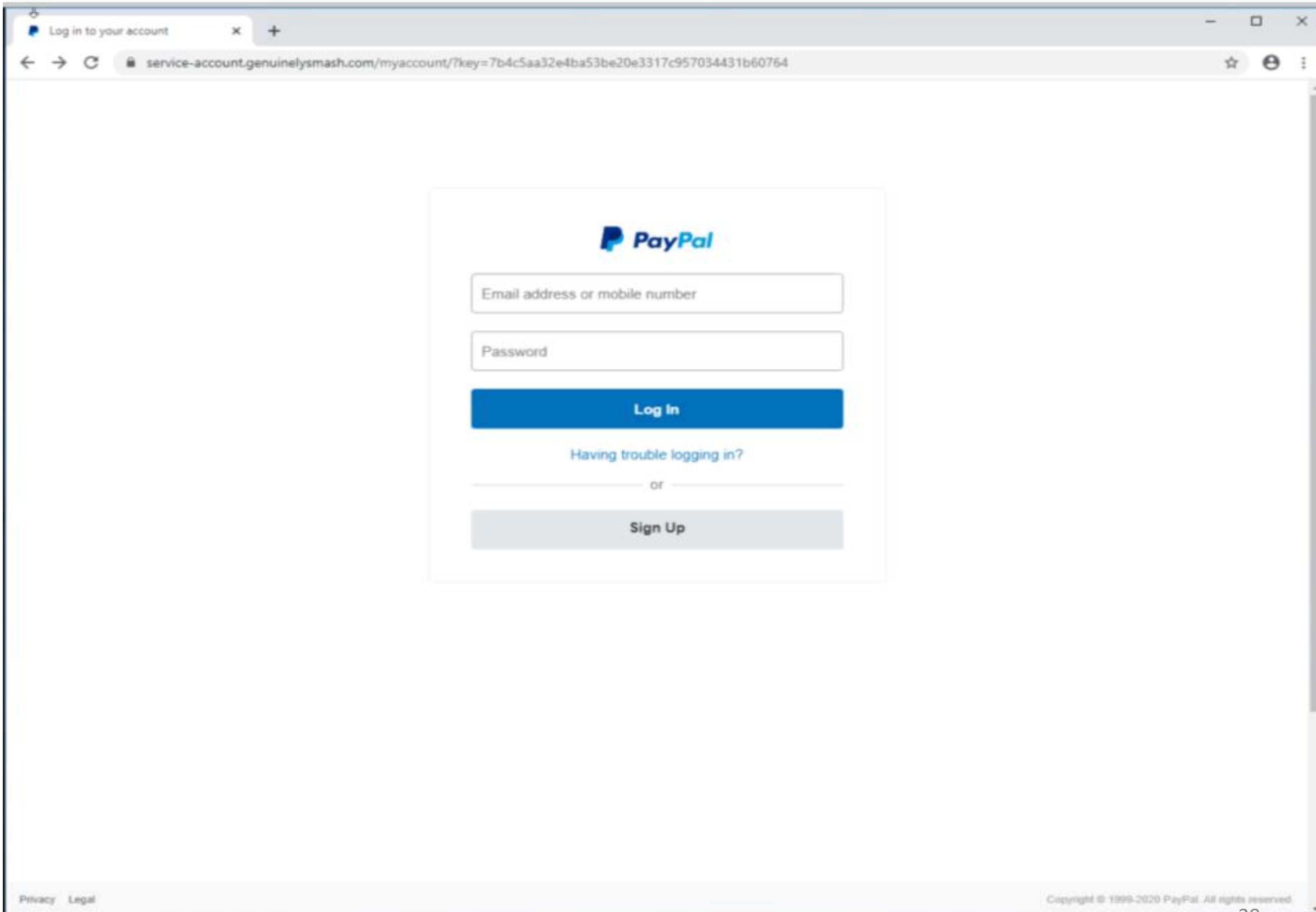
PayPal

Hello Dear Customer,

recently we have limited your account access due suspected and illegal uses.

Please Check your account as soon as you can by Clicking the button below

Check it now

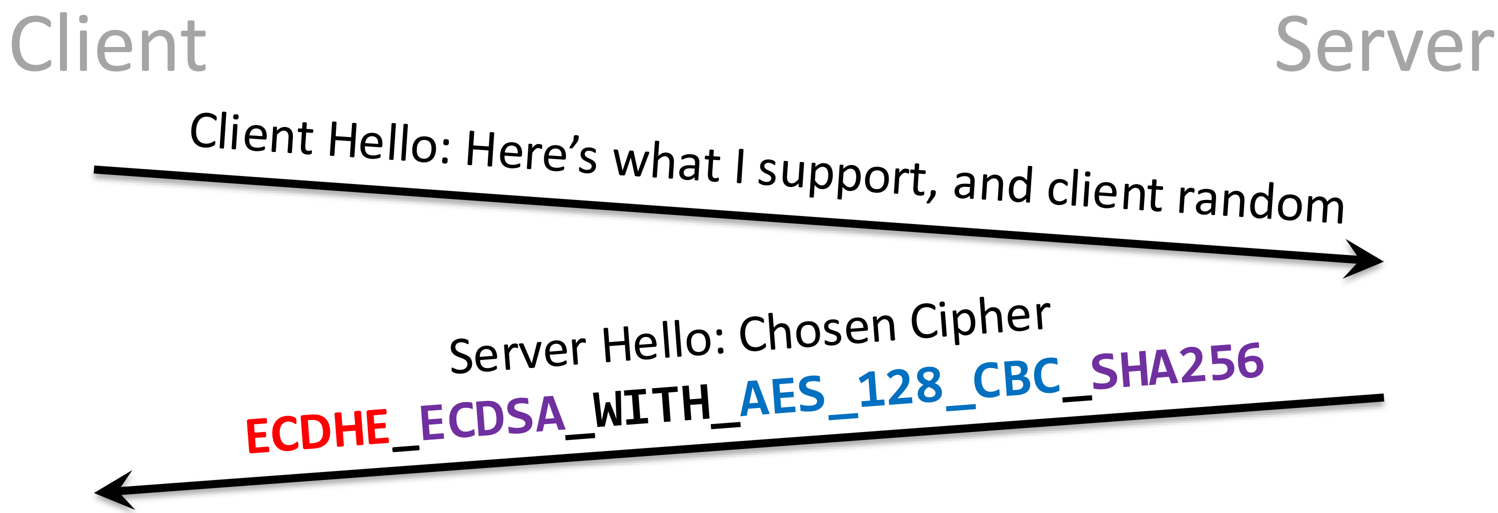


Log in to your account
service-account.genuinelysmash.com/myaccount/?key=7b4c5aa32e4ba53be20e3317c957034431b60764
PayPal
Email address or mobile number
Password
Log In
Having trouble logging in?
or
Sign Up
Privacy
Legal
Copyright © 1999-2020 PayPal. All rights reserved.

TLS with Key Exchange

- We just saw TLS with RSA. Recall that we recommend **key exchange** over RSA.

- Diffie-Hellman key exchange using elliptic curves is faster and less error prone



ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

ECDHE	ECDSA	AES_128_CBC	SHA256
Elliptic Curve Diffie-Hellman Ephemeral	Elliptic Curve Digital Signature	Symmetric Encryption	HMAC

Client Server

Client Hello: Here’s what I support, and client random

Server Hello: Chosen Cipher

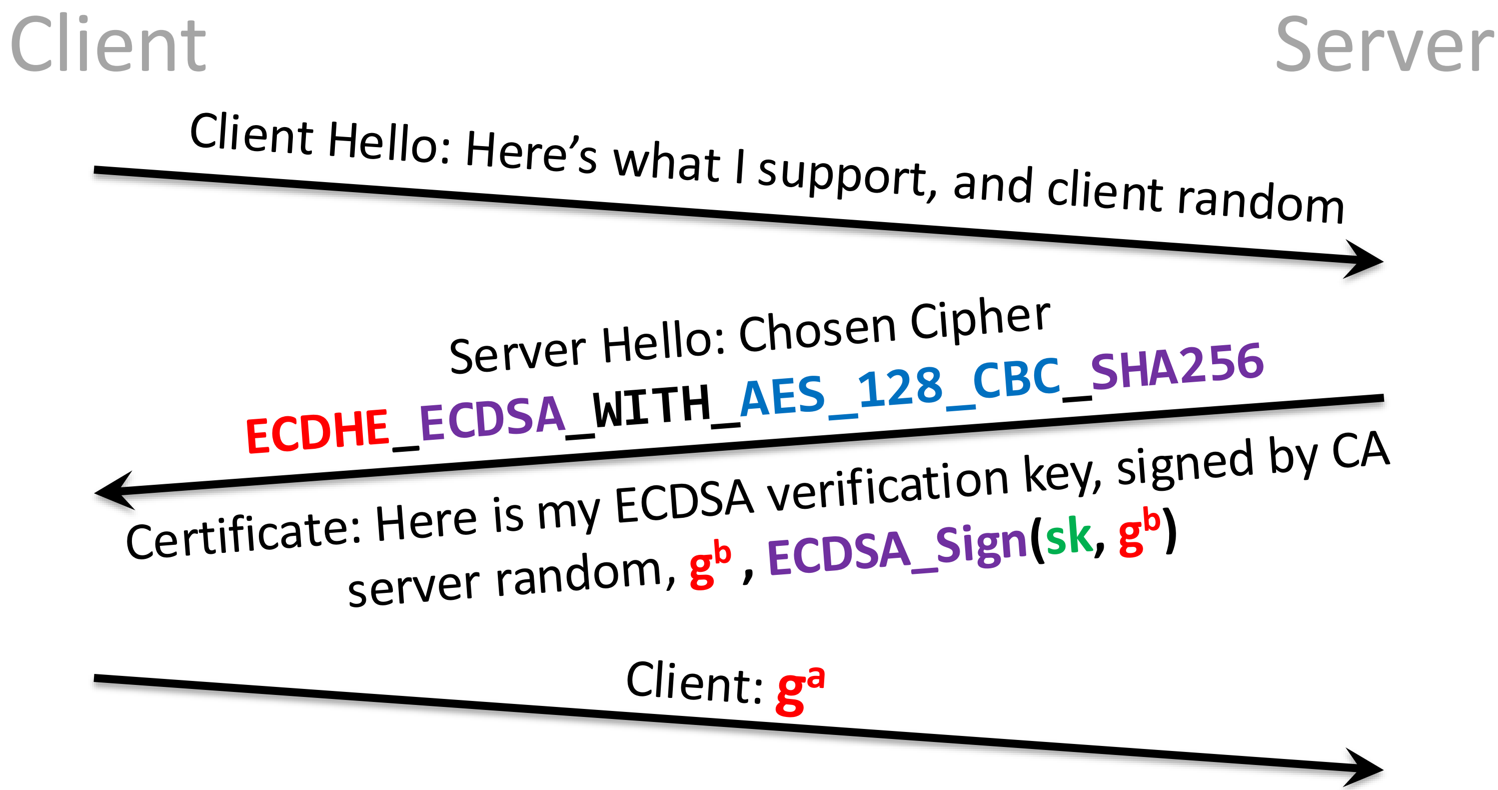
ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

Certificate: Here is my ECDSA verification key, signed by CA

server random, g^b , **ECDSA_Sign(sk, g^b)**

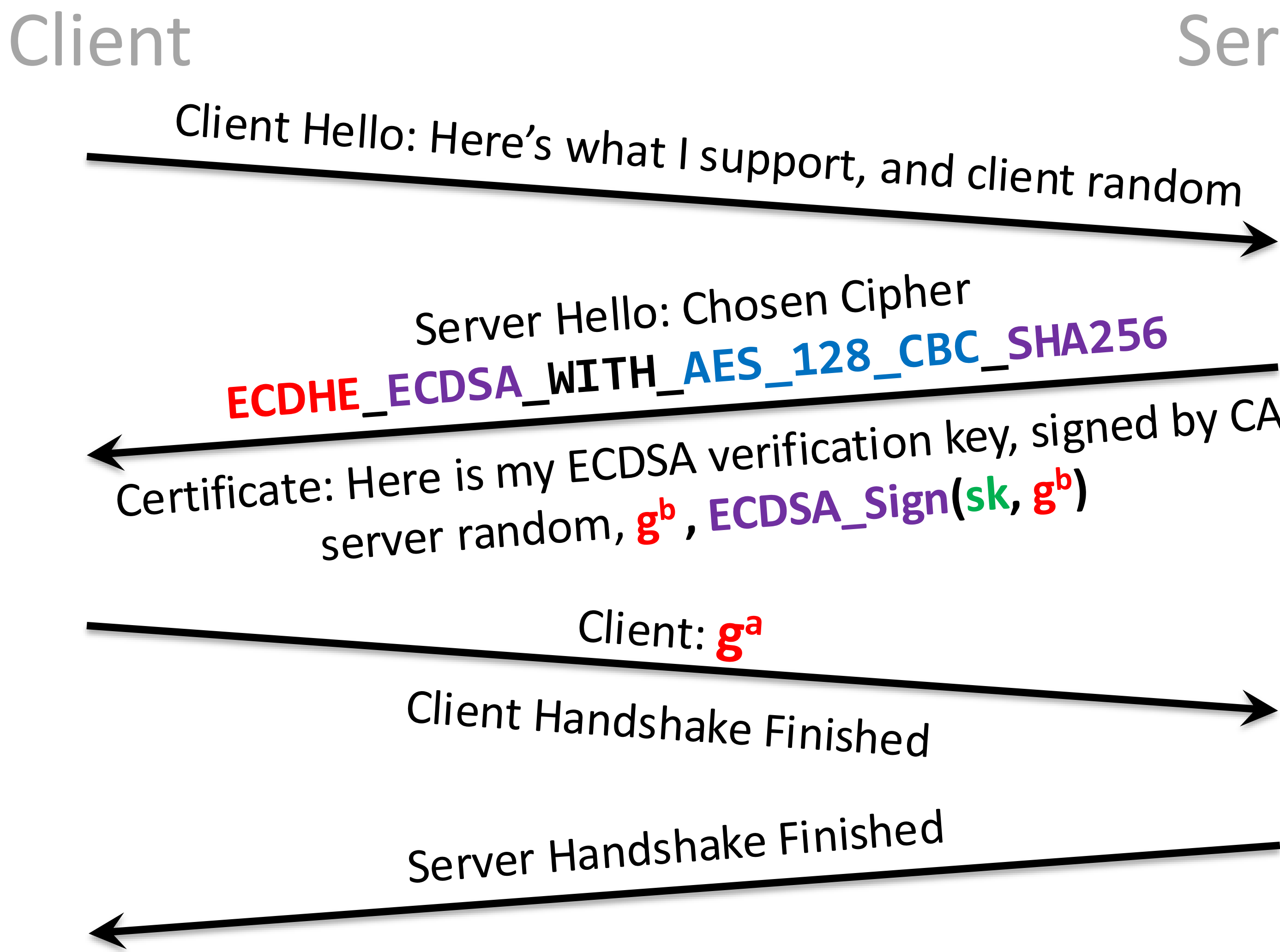
Note the chain of signatures:

- CA verification key hardcoded in browser →
- possibly multiple hops of intermediate CA verification key →
- server’s ECDSA verification key →
- server’s randomly generated g^b



Session key = Hash(secret = g^{ab}, client random, server random)

where a and b are randomly selected by client and server



Handshake Finish contains an HMAC for the interaction so far to prevent cipher suite downgrade attacks.

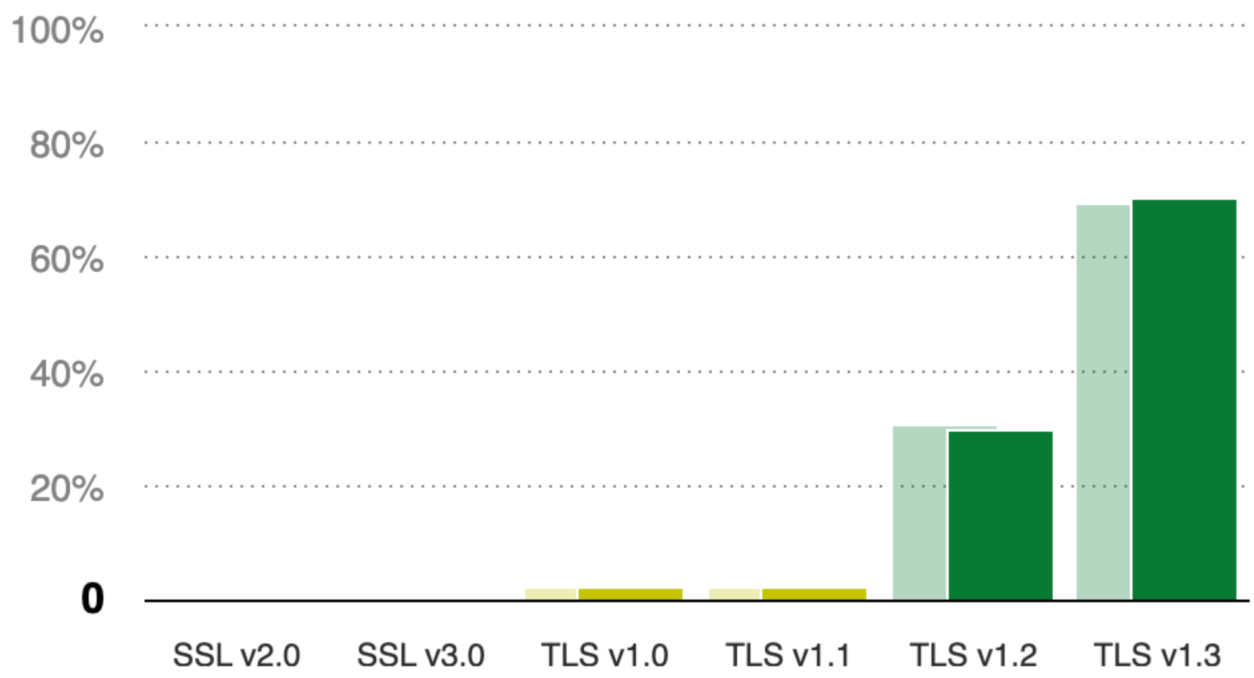
After Handshake Finish messages, client and server start using AES_128_CBC for confidentiality and SHA256 HMAC for integrity.

Forward Secrecy

- If secret key is stolen, prior communication remains secure
- RSA-based TLS is not forward secret. Why?
- DHE-based TLS is forward secret.
  - Long-term secret is a signing key that is used to sign Diffie-Hellman message g^b
  - Once stolen, attacker can impersonate server by signing its own $g^{b'}$
  - But it cannot figure out past b or g^{ab}
    - which should be deleted once used (hence ephemeral)

TLS 1.3 Adoption

- TLS 1.3 is the latest version, released in 2018



TLS 1.2 vs. TLS 1.3

- TLS 1.2 has lots of bad options of cipher suites

TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA
TLS_RSA_WITH_NULL_SHA256
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DH_anon_WITH_AES_128_CBC_SHA
TLS_DH_DSS_WITH_AES_256_CBC_SHA
TLS_DH_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA256
TLS_DH_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DH_anon_WITH_AES_128_CBC_SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256

TLS 1.2 vs. TLS 1.3

- TLS 1.2 has lots of bad options of cipher suites
- TLS 1.3 has only five, all highly recommended
  - All use Diffie-Hellman ephemeral (hence omitted)

TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_CCM_SHA256
TLS_AES_128_CCM_8_SHA256

TLS 1.2 vs. TLS 1.3

- TLS 1.2 has lots of bad options of cipher suites
- TLS 1.3 has only five, all highly recommended
  - All use Diffie-Hellman ephemeral (hence omitted)
- TLS 1.3 removes the round trip for cipher suite selection

Summary

- TLS provides confidentiality and integrity against even MitM attackers
  - Sits in between TCP and application
  - Makes use of most of the crypto tools we learned
  - Diffie-Hellman ephemeral is preferred (and the only option in TLS 1.3)
- Assumptions: crypto, keys, root of trust, CAs
  - CA and users are weak links