* RSA Encryption (summary)

* TLS / HTTPS

Integers mod p

* DH / STS → key exchange.
* Elgamal → public key encryption.
  - ↳ CPA-secure.
  - ↳ Not CCA-secure.
* DSA / Schnorr → signature scheme.

Integers mod n

↳ $n = p \cdot q$ (safe) prime numbers.

* RSA encryption
  - ↳ OTS-secure (textbook)
    - ↳ Deterministic.
  - ↳ Padding → OAEP
    - ↳ RSA+OAEP
      - ↳ CCA-secure.

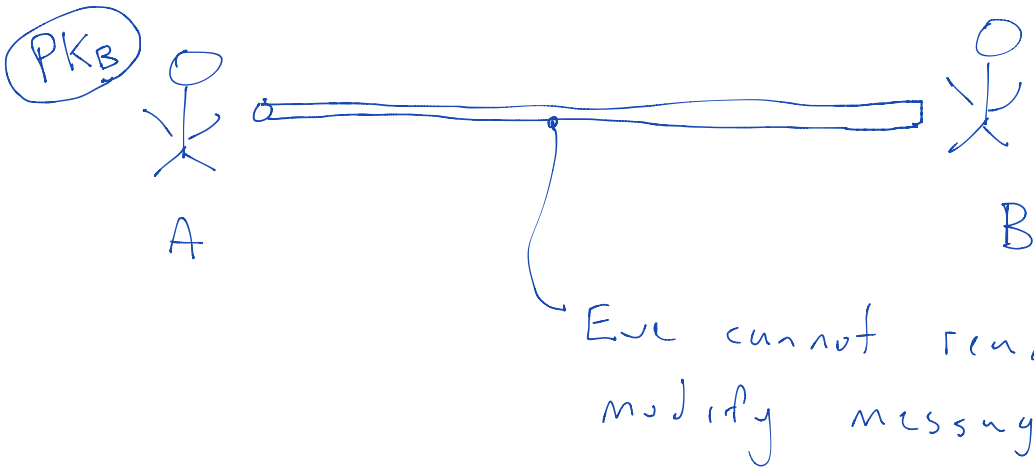
* RSA Signatures
  ↳ RSA + PSS ⇒ highest level of security.
    ↑ padding

Signature Defn (Aside)

Recall a signature on message m is σ. A signature is secure if it is infeasible to generate the correct σ' for message $m' \neq m$ without knowing the secret key.

↳ unforgeability.

Secure Transport



Assumption: Alice knows Bob's public key. How?

↳ INSE 6150

Question

Is PK_B an encryption public key or a signature or both.

↳ we can set-up channel with either.

PK_B → Encryption → key transport.
→ Signature → key agreement

Process:

(1) Get Alice and Bob to agree on symmetric keys.

(2) Use CCA-secure symmetric crypto to transfer messages.

Step 1: Get a shared symmetric key(s)

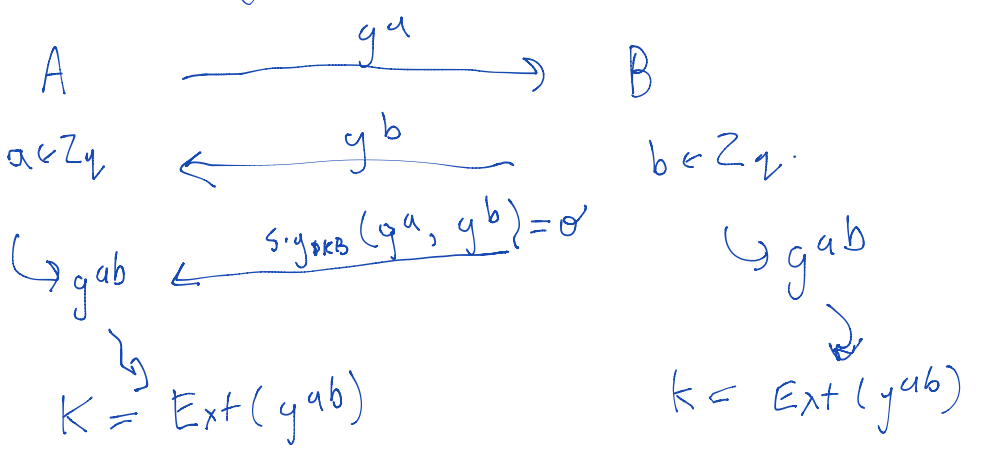
Key Transport (KT)

A $\xrightarrow{c = Enc_{PK_B}(k)}$ B

$k \in \{0,1\}^\ell$ ↑ symmetric keys.

$Dec(c) = k$

Key Agreement (KA)

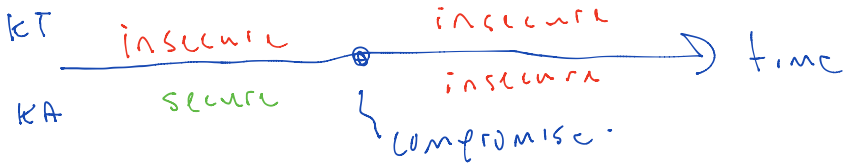


* End product of KT and KA is the same: a shared key K.

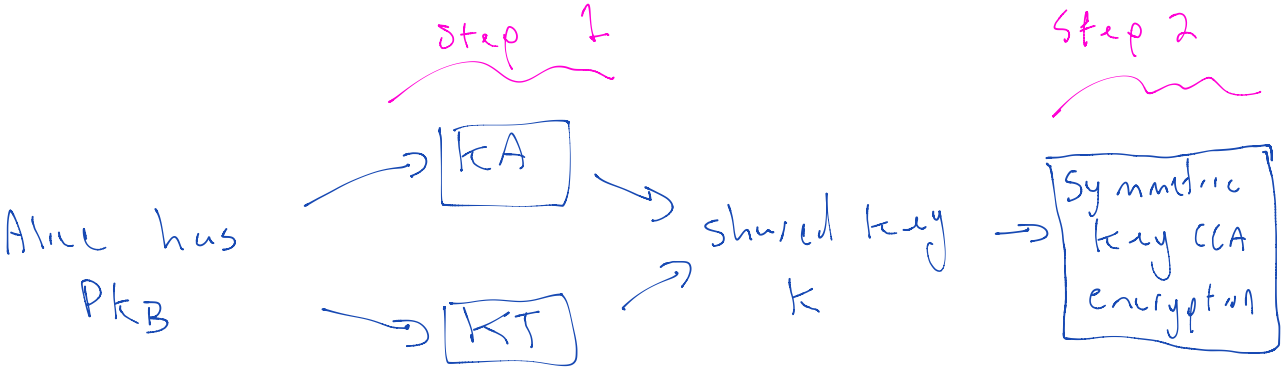
* KT is faster than KA
  ↳ one message / no round-trip

* KA is more secure than KT against compromises of SK_B corresponding to pK_B.

→ Perfect Forward Secrecy. (PFS).



* Diffie Hellman / STS has PFS
* Key Transport cannot



Step 2

Once Alice and Bob share a secret key K, they can transfer data.

* K is called a "master secret"

{a set of keys} = PRG(K)

↳ set will depend on what is being used.

Enc + MAC

A

$\langle k_E, k_m \rangle = PRG(k)$

B

$\langle k_E, k_m \rangle = PRG(k)$

$Enc_{k_E}+MAC_{k_m}(m)$

HMAC → MD5 (*), SHA1 (~), SHA256 (✓), SHA3

CBC → AES (✓), 3DES (~), DES (✗)

CTR

Stream Cipher → RC4 (✗), chacha20 (✓)

Authenticated Encryption

$k_E \leftarrow PRG(k)$ — $Enc_{k_E}(m)$ → — $k_E \leftarrow PRG(k)$

CCA secure

↳ GCM → AES

HTTPS

↳ HTTP over SSL/TLS.

crypto

soon

{1.0, 1.1, 1.2, 1.3}.

latest

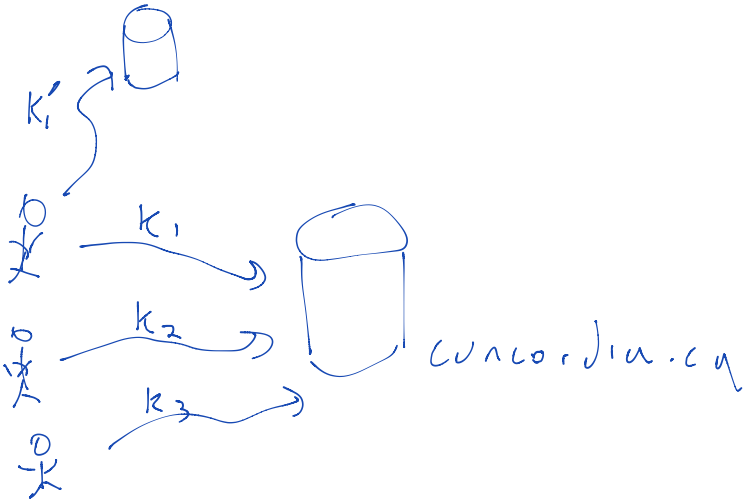
{2.0, 3.0}

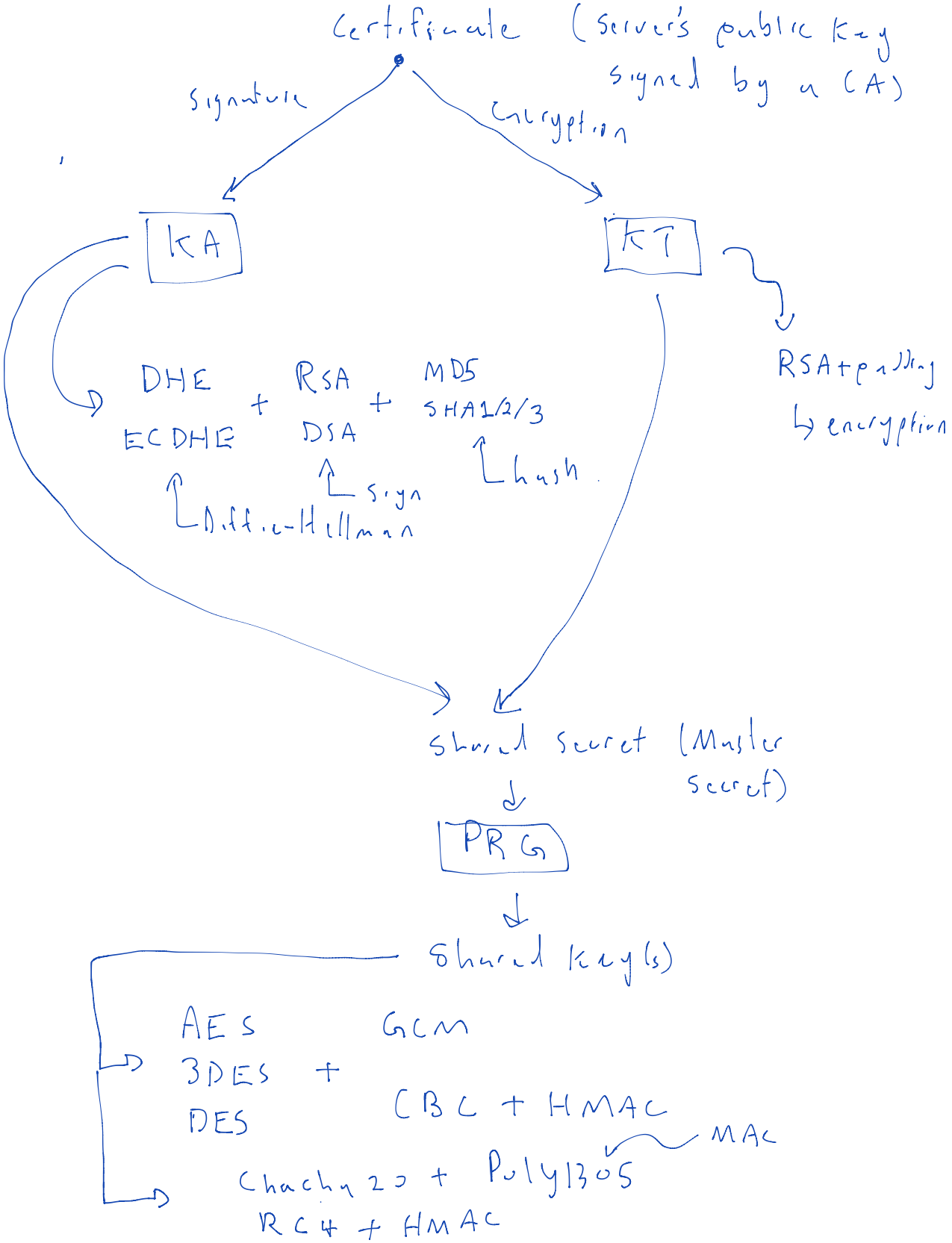
Web protocol

① Confidentiality
② Message Integrity
} 6110

③ Server Authentication } 6150

↳ How Alice knows Bob's public key.




Certificate (server's public key signed by a CA)
Signature
Encryption
KA
KT
DHE
ECDHE
+
RSA
DSA
+
MD5
SHA1/2/3
sign
hash
Diff.-Hellman
RSA + padding
↳ encryption
shared secret (Master secret)
PRG
shared key(s)
AES
3DES +
DES
GCM
CBC + HMAC
MAC
Chacha20 + Poly1305
RC4 + HMAC

Qualys. SSL Labs

Home Projects Qualys.com Contact

You are here: Home > Projects > SSL Server Test > www.concordia.ca

SSL Report: www.concordia.ca (132.205.244.70)

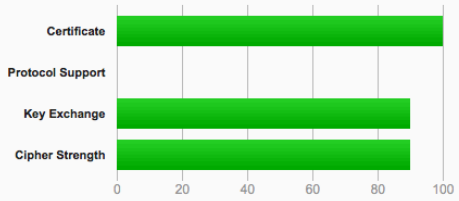
Assessed on: Tue, 05 Dec 2017 00:49:50 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

F



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server is vulnerable to the POODLE TLS attack. Patching required. Grade set to F. MORE INFO »

Cipher Suites

 # TLS 1.2 (suites in server-preferred order)

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	256

 # TLS 1.1 (we could not determine if the server has a preference)

 # TLS 1.0 (we could not determine if the server has a preference)