CSCI 3330 Cybersecurity

Word of the day

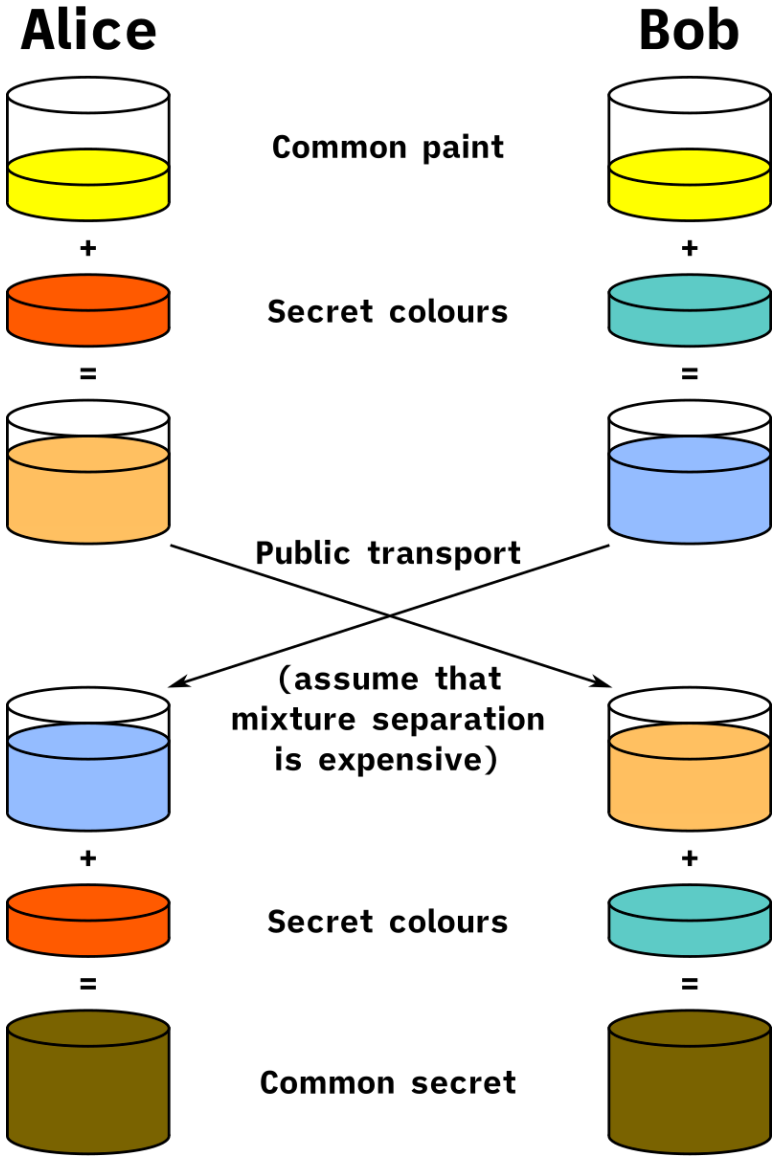
QUALITY BAR n. The remarkably flexible level of acceptability in a product. Tends inexorably to drop as the pressure of an impending milestone, content freeze, or other deadline builds.

Agenda for today

- Diffie-Hellman
- Quantum cryptography
- TLS

Diffie-Hellman

- Key exchange algorithm
- Assumes difficulty of the discrete logarithm problem
- Agree on a symmetric key without ever transmitting it



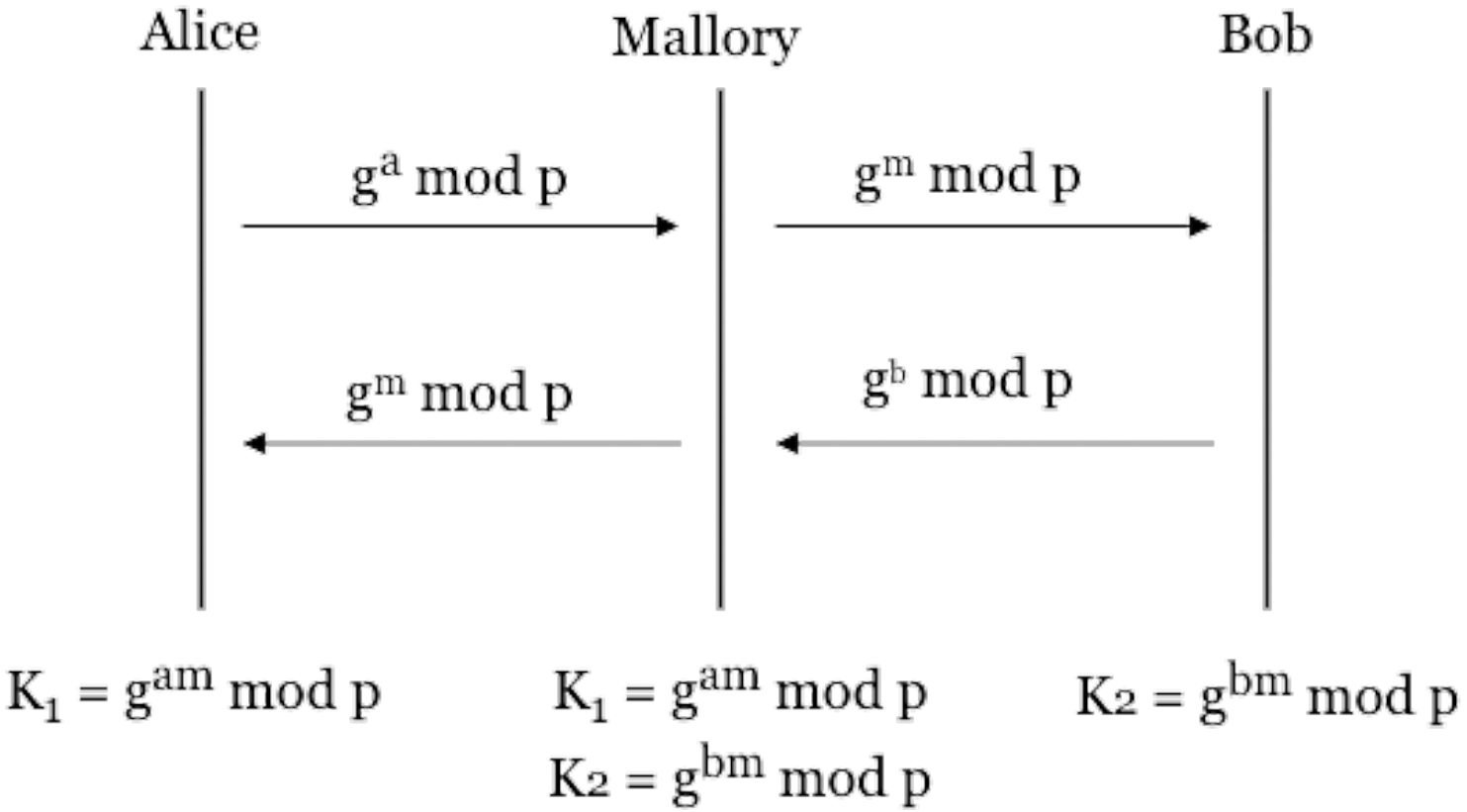
Exchange between Alice and Bob

- Let p be prime and g be a primitive root mod p
- Assume $p = 23$, $g = 5$
- Alice picks secret $a = 12$ and sends $g^a \% p$ to Bob
- Bob picks secret $b = 4$ and sends $g^b \% p$ to Alice
- Alice computes $(g^b \% p)^a \% p = g^{ab} \% p$
- Bob computes $(g^a \% p)^b \% p = g^{ab} \% p$

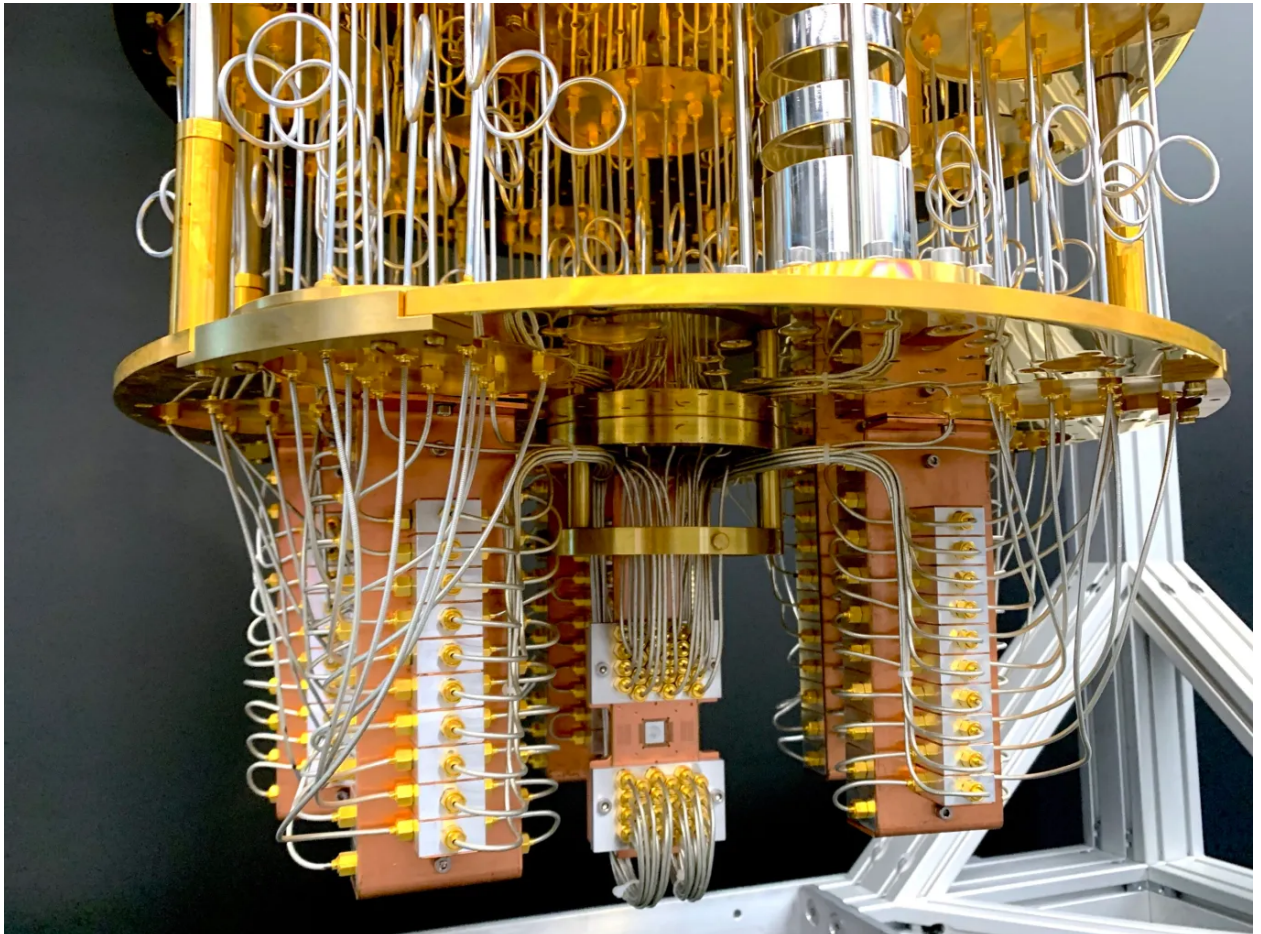
So we don’t need RSA?

- Just because we have asymmetric encryption doesn’t mean we don’t still need symmetric
- Similarly, we still need RSA (or another asymmetric algorithm) to exchange keys
- It just becomes *another* part of the process
- Diffie-Hellman by itself **provides no authentication**
- Server uses RSA to sign $g^x \% p$

Active attack possible



Quantum computers



Quantum computers

- Quantum computers can compute some problems faster than classical (asymptotically)
- Can they be used to break crypto?
- And if so, how do we make our algorithms robust?

Grover’s algorithm

- You can search an *n* element space in O(sqrt(*n*)) time.
- Classical computer: O(*n*) time.

Grover’s algorithm

- Cracking 128-bit AES key
- Conventional computer: 2^{128} guesses
- Quantum computer: sqrt(2^{128}) guesses
- Effectively halves key size
- How do we remedy this?

Grover’s algorithm

- Remediation: double key size!
- Use a 256-bit AES key

Shor’s algorithm

- Integer factorization time complexity?
- Classical computer: sub-exponential (best known)
- Quantum computer: polynomial time…
- What algorithm can we break?

Shor’s algorithm

- Recall: RSA public key is (*n*, *e*)
- If we can factor *n* into *p* and *q*, we can derive *d*, the private key: `pow(e, -1, lcm(p-1, q-1))`

Shor’s algorithm

- Recall: RSA public key is (*n*, *e*)
- If we can factor *n* into *p* and *q*, we can derive *d*, the private key: `pow(e, -1, lcm(p-1, q-1))`
- How do we remedy this?
- With Grover’s, exponential was still exponential

Shor’s algorithm

- Quantum-resistant (or post-quantum) algorithms
- These are algorithms designed to use mathematical problems difficult on both classical and quantum computers

Quantum prognosis

- Quantum computers capable of solving nontrival problems may be physically impossible
  - Quantum computers have factored 15 (but not 21)
- Forward secrecy protects historical communication, so we have time to migrate

Algorithm summary

- Grover’s algorithm: threatens symmetric
- Shor’s algorithm: threatens asymmetric

TLS

- Encrypts almost all encrypted traffic on the Internet
- Powers HTTPS, DoT, SMTPS, IMAPS, FTPS, etc.

TLS

- Since the Edward Snowden disclosures, there has been a large push to encrypt all traffic on the Internet
- Almost every web site supports HTTPS these days
- Before, not the case
- Even banks would often only encrypt logins

History

- SSL 1.0 (1994)
- SSL 2.0 (1995)
- SSL 3.0 (1996)
- TLS 1.0 (1999)
- TLS 1.1 (2006)
- TLS 1.2 (2008) (still secure, still most popular…)
- TLS 1.3 (2018) (important but controversial features)

TLS ciphers (examples)

- TLS_RSA_WITH_DES_CBC_MD5
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS handshake (assuming DH)

- Client sends “Client Hello”:
  - Supported TLS versions and cipher suites
- Server sends:
  - A “Server Hello”:
    - A TLS version and cipher suite from the list
  - An SSL certificate
  - A *signed* “Server Key Exchange” (DH credentials)
- Client *verifies* the certificate

Certificate verification

- Certificate = a signed public key
- Signing a public key: a little recursive
- Signed by certificate authority
- Most browsers / operating systems agree on trusted authorities

TLS handshake (assuming DH)

- Client replies with:
  - A “Client Key Exchange” encrypted with the public key in the SSL certificate
- Server and client compute a shared secret using Diffie-Hellman and “server random” and “client random”
- Client *verifies* the certificate

Assignment 6

- Deadline: November 6, 2025, 10:00pm
- CBC padding oracle attacks
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment6.pdf

Assignment 7

- Deadline: November 18, 2025, 10:00pm
- Textbook RSA attacks
  https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment7.pdf