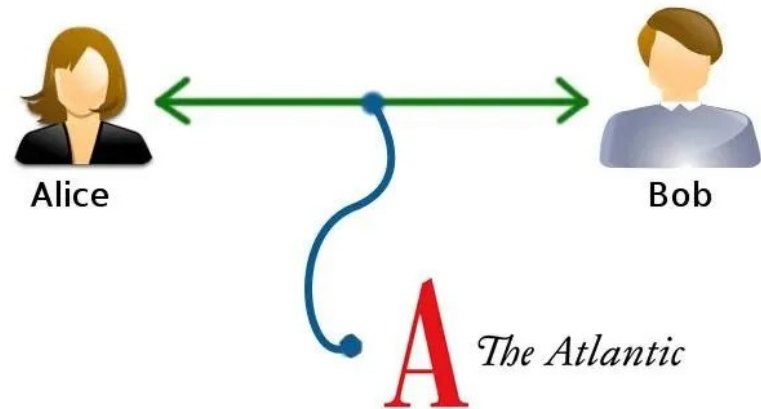


CS 470 Spring 2026

Mike Lam, Professor



Security

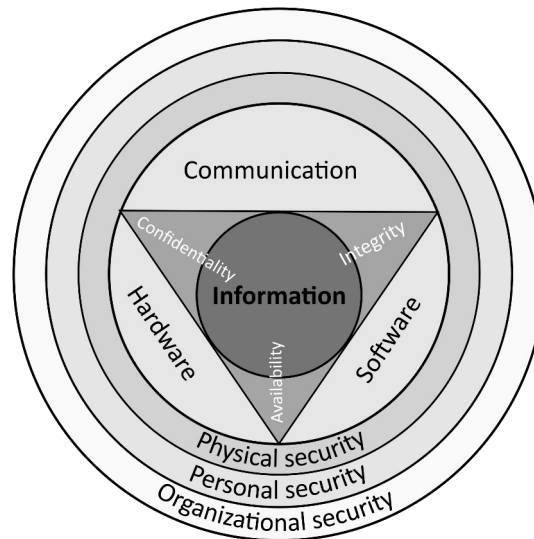
Content taken from the following:

"*Distributed Systems: Principles and Paradigms*" by Andrew S. Tanenbaum and Maarten Van Steen (Chapter 9)

Various online sources

Security Issues (“CIA Triad”)

- **Confidentiality**: data is only disclosed to authorized users
- **Integrity**: changes can only be made by authorized users
- **Availability**: data is consistently accessible to authorized users
- Security threats
 - **Interception**
 - **Interruption**
 - **Modification**
 - **Fabrication**



Threat models

- **Interception:** has data been received by an attacker?
 - Usually reserved for receipt of *unencrypted* data
- **Interruption:** can a service be disrupted by an attacker?
 - Sometimes via multiple sources
- **Modification:** can an attacker change data during transmission?
 - Enables “person-in-the-middle” attacks
- **Fabrication:** can an attacker create legitimate-looking data?
 - Does not require existing communication

Security policies

- Security **policy**: description of actions allowed in a system
 - E.g., *"users in group 'students' may read files located in /shared but cannot write to them"*
- Policy enforcement mechanisms
 - **Encryption**
 - **Authentication**
 - **Authorization**
 - **Auditing**



Security policies

- **Encryption:** are messages secure against eavesdroppers?
 - Variation on end-to-end principle
- **Authentication:** are you connecting to the real recipient?
 - Issue of identity verification
- **Authorization:** do you have permission to perform this action?
 - Intersects with business/policy concerns
- **Auditing:** has the system been compromised?
 - Often bound by legal requirements

Least privilege

- Principle Of “**Least Privilege**” (POLP)
 - Every process or user should only be able to access resources or perform actions that are *strictly necessary*
 - Systems should be designed to *minimize privilege*
 - Limits vulnerability of the system to compromised components
 - Minimizes the need for full trust in participants
 - **Social engineering** can compromise even well-meaning participants

Least privilege

- The principle of "least privilege" often reveals a tension between security and
 - A. scalability
 - B. consistency
 - C. partition tolerance
 - D. convenience
 - E. availability

Trust

- How much of your computer do you *trust*?
 - (and what does that even mean?)
- "*Reflections on Trusting Trust*"
 - A compiler virus that inserts a backdoor into `login()`
 - It also re-inserts itself to any further compilers
 - Ken Thompson Turing Award lecture (1984)
<https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf>
- **Trusted Computing Base (TCB)**
 - Minimal component of a system trusted to enforce security policies
 - Sometimes a physically-separate ROM-based processor
 - Hidden encryption key inaccessible to the rest of the system
 - Trusted Computing Group's **Trusted Platform Module (TPM)**

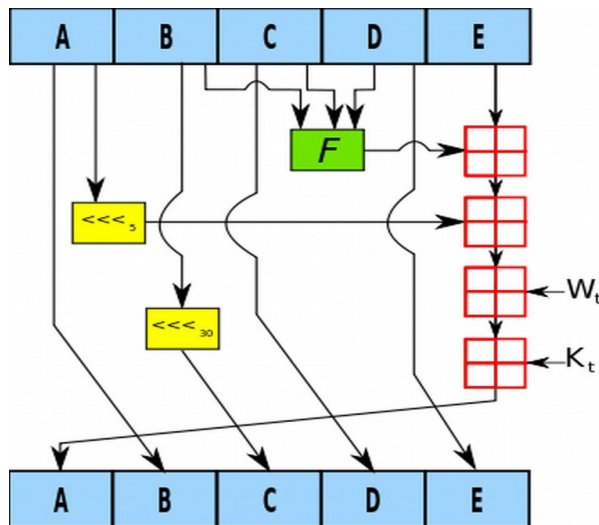
```
compile(s)
char *s;
|
|   if(match(s, "pattern1")) {
|       compile ("bug1");
|       return;
|   }
|   if(match(s, "pattern 2")) {
|       compile ("bug 2");
|       return;
|   }
|   ...
|
```

Security policy enforcement

Encryption

Hash functions

- One-way hash functions w/ collision resistance
 - Computationally infeasible to reverse
 - MD5: 128-bit fixed-length message digest
 - SHA-1 / SHA-2 / SHA-256 / SHA-512



One iteration of SHA-1

SHA1("The quick brown fox jumps over the lazy dog")
= 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

SHA1("The quick brown fox jumps over the lazy cog")
= de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

A, B, C, D and E are 32-bit words of the state;
F is a nonlinear function that varies;
 \lll_n denotes a left bit rotation by n places;
n varies for each operation;
 W_t is the expanded message word of round t;
 K_t is the round constant of round t;
+ denotes addition modulo 2^{32}

Cryptography

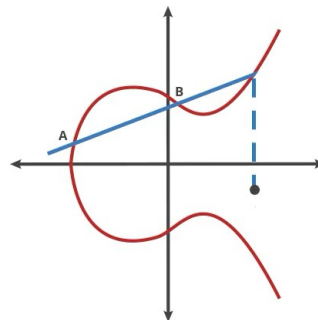
- Terminology
 - **Plaintext**: original message
 - **Ciphertext**: encrypted plaintext
 - **Nonce**: random number that is only used once
 - **Encrypt**: turn plaintext into ciphertext
 - $C = E_K(P)$
 - Usually based on a one-way hash function
 - **Decrypt**: turn ciphertext into plaintext
 - $P = D_K(C)$
 - Alternatively: $P = D_K(E_K(P))$
 - **Cryptographic system**: pair of $D()$ and $E()$ functions

Cryptography

- **Symmetric** ($P = D_K(E_K(P))$) vs. **asymmetric** ($P = D_{KD}(E_{KE}(P))$)
 - Same key vs. key pair
 - **Private key** vs. **public/private keys**
- Symmetric (e.g., **Advanced Encryption Standard (AES)**)
 - Various bitwise operations with different key values
 - Fast to encrypt/decrypt, relies on robust secret keys
 - Relatively secure against quantum computing attacks
- Asymmetric (e.g., **Rivest, Shamir, Adleman (RSA)**)
 - Multiplication and modulus operations with large prime keys
 - Signing (encrypt w/ private) and secure messaging (encrypt w/ public)
 - Slow to encrypt/decrypt
 - Relies on difficulty of **prime factorization** or **elliptic curve discrete logarithms**

Elliptic curve cryptography

- **Elliptic curves** (e.g., $y^2 = x^3 + ax + b$)
 - Horizontal symmetry, and any non-vertical line will intersect the curve in at most three places
 - “Dot” operation: given two points, find third and then reflect
 - Very difficult to undo! (essentially a one-way hash)
 - **ECDSA** is a variant of DSA that uses elliptic curves



Security policy enforcement

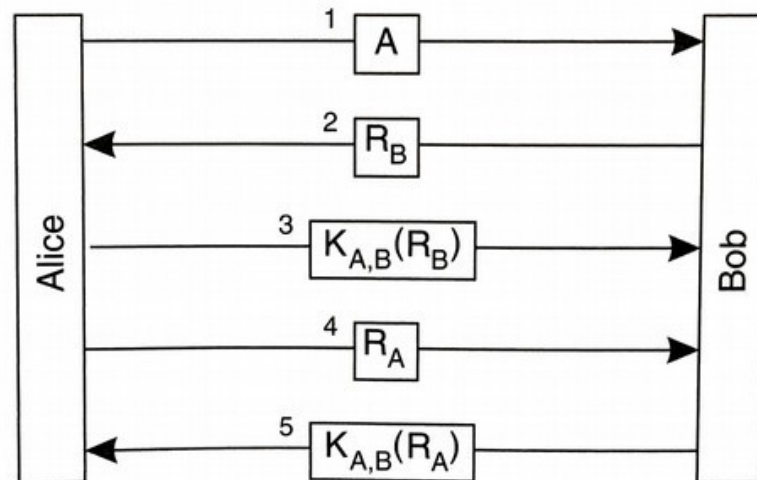
Authentication

Authentication

- A **secure channel** provides security on an unsecured network
 - Requires some kind of setup first
 - Protects against interception, modification, and fabrication
 - Cannot prevent interruption (recall CAP theorem)
 - Issue: authentication (verifying the identity of the recipient)
 - Issue: establishing shared secrets (after verifying identity)
- Security protocols
 - Shared-key authentication (requires pairwise secrets)
 - **Needham-Schroeder** authentication (uses central server)
 - Key signing parties (physical exchange of keys)
 - **Diffie-Helman** key exchange (uses public messaging)

Shared-key authentication

- Basic **challenge-response** protocol
 - Alice contacts Bob (“A”)
 - Bob issues a challenge (“ R_B ”) and receives a response (R_B encrypted using shared key “ $K_{A,B}$ ”)
 - Alice also issues a challenge (“ R_A ”) and receives a similar response
 - Issue: requires shared key



Needham-Schroeder authentication

- Uses a central **Key Distribution Center (KDC)**
 - Alice sends a nonce to the KDC to request communication with Bob
 - The nonce prevents a **replay attack** using an old (compromised) $K_{B,KDC}$
 - Alice receives a new shared key ($K_{A,B}$) as well as an encrypted copy to send to Bob
 - Bob and Alice then exchange challenges and responses using this shared key

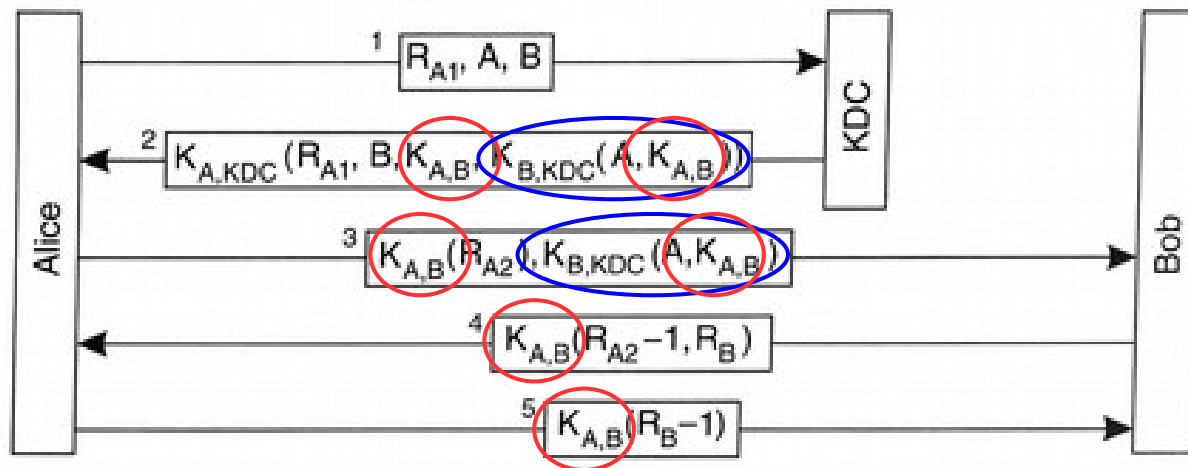


Figure 9-17. The Needham-Schroeder authentication protocol.

shared secret

Bob's copy of shared secret

Needham-Schroeder authentication

- Kerberos is similar, but uses two servers:
 - Authentication Server (AS) to establish identity (authentication)
 - Ticket Granting Server (TGS) to verify permissions (authorization) and set up shared key

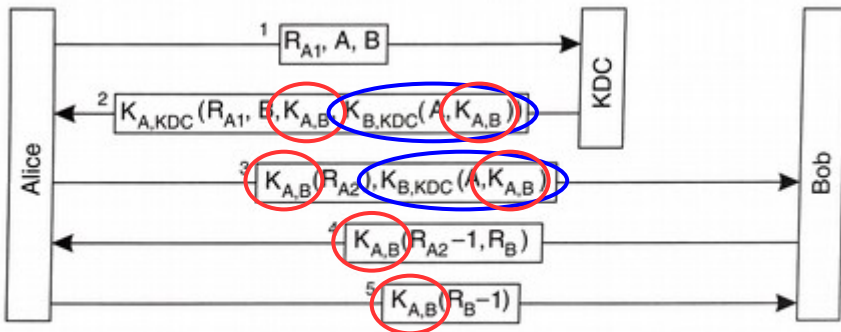


Figure 9-17. The Needham-Schroeder authentication protocol.

shared secret (red oval) Bob's copy of shared secret (blue oval)

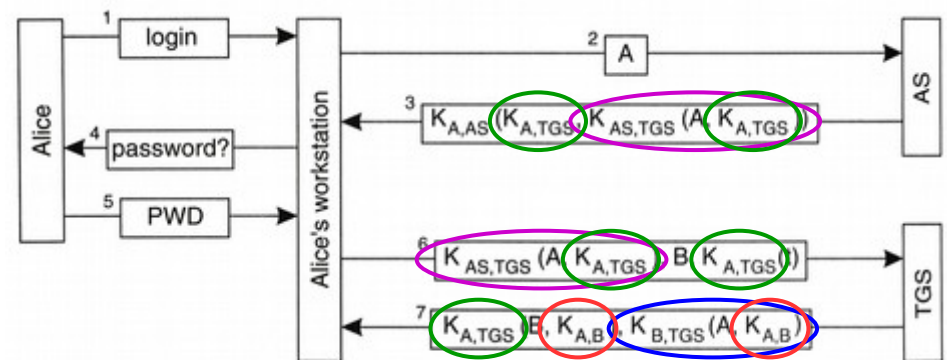


Figure 9-23. Authentication in Kerberos.

session key (green oval) ticket (purple oval) from Tanenbaum and Van Steen (Ch. 9)

Kerberos

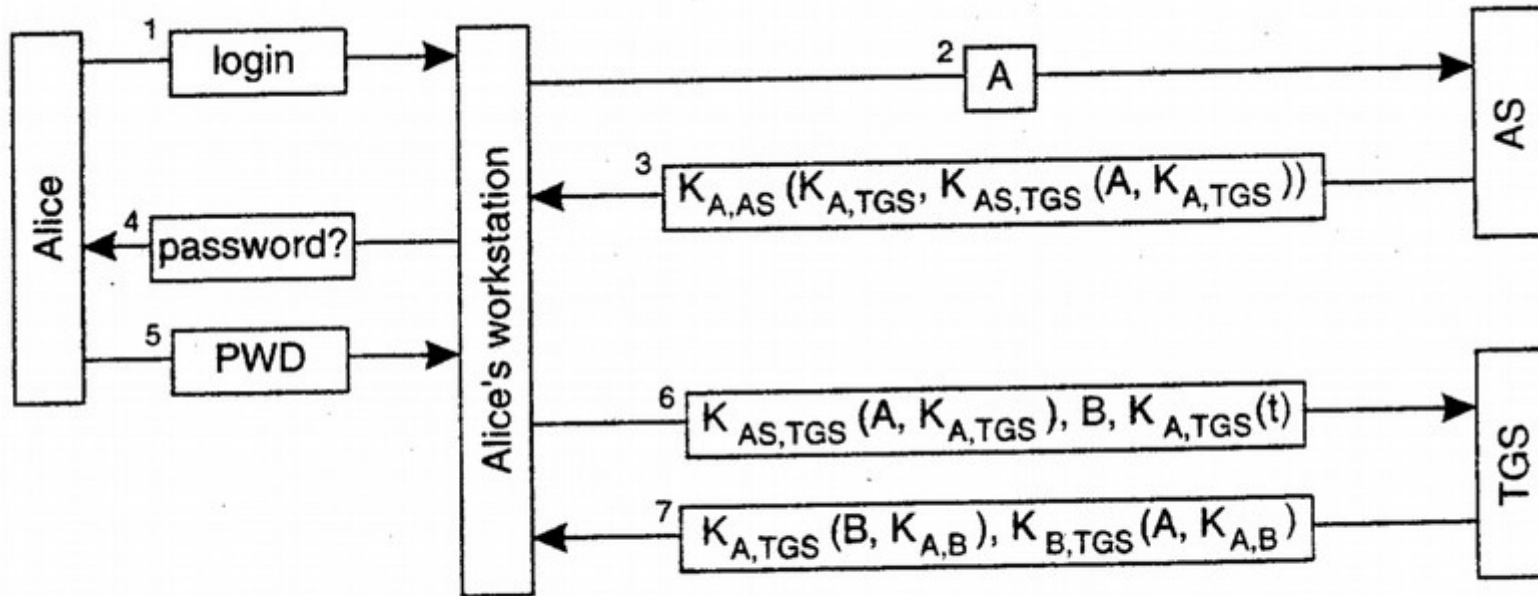


Figure 9-23. Authentication in Kerberos.

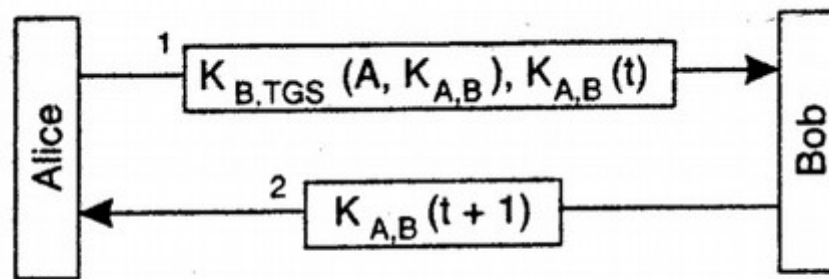
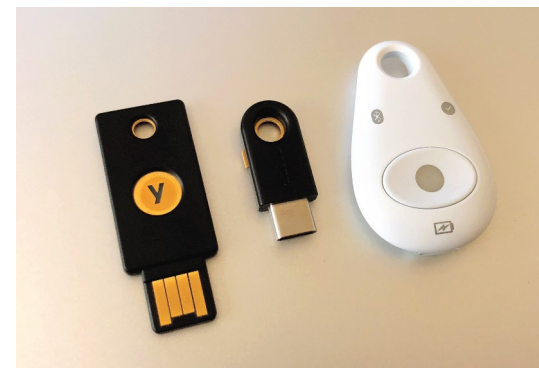


Figure 9-24. Setting up a secure channel in Kerberos.

Multi-factor authentication

- **Multi-factor authentication (“MFA”)**
 - Most common: **two-factor authentication (“2FA”)**
 - Requires you to **have** something in addition to **knowing** something (i.e., a password)
 - Push/SMS notifications, one-time codes, hardware keys/tokens, biometrics
 - **Duo Mobile, Okta Verify**



Public keys

- Private keys are used to **sign** documents by encrypting them
 - Public key can also be used to encrypt a document for a single recipient (the one who holds the private key)
- A **certificate** is a signed document claiming to own a public key
 - Only the public key can decrypt the document, proving it was encrypted using the corresponding private key
- At a **key signing party**, participants exchange public keys
 - This allows others to later sign a certificate containing a known public key (thus vouching for its authenticity)
 - Purely peer-to-peer; no central server required

Public keys

- Issues: **scaling** and **certificate revocation**
 - Revocation lists and certificate lifetime limits
- In a large distributed system, a **Public-Key Infrastructure (PKI)** provides scalable certificate management
 - Usually implemented using trusted third-party **certificate authorities (CAs)**
 - CAs issue certifications, handle authorization requests, and revoke certificates when necessary
 - **Domain validation (DV)** vs. **organization/extended validation (OV/EV)**

Let's Encrypt

- Open source and free certificate authority
 - Goal: make HTTPS (encrypted HTTP) ubiquitous
 - **Automated Certificate Management Environment (ACME)** protocol for certificate issuing

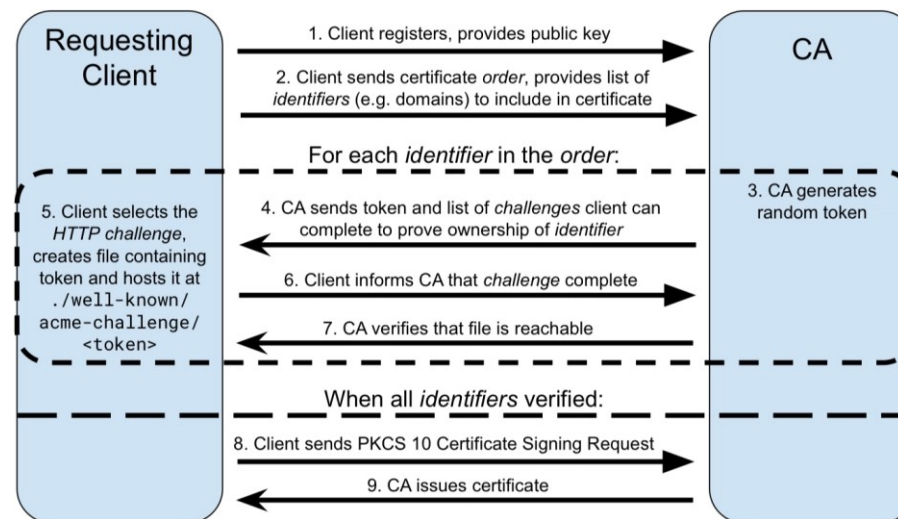


Figure 2: ACME protocol. This diagram illustrates how an ACME client can obtain a certificate without human interaction. In the dashed region, the client proves ownership of the domain using an HTTP-based challenge.

Diffie-Hellman key exchange

- Allows distributed entities to establish a shared secret via unsecured channels
- Can be extended to more than two entities
- Resists **person-in-the-middle** attacks
 - Third party pretends to be other conversant

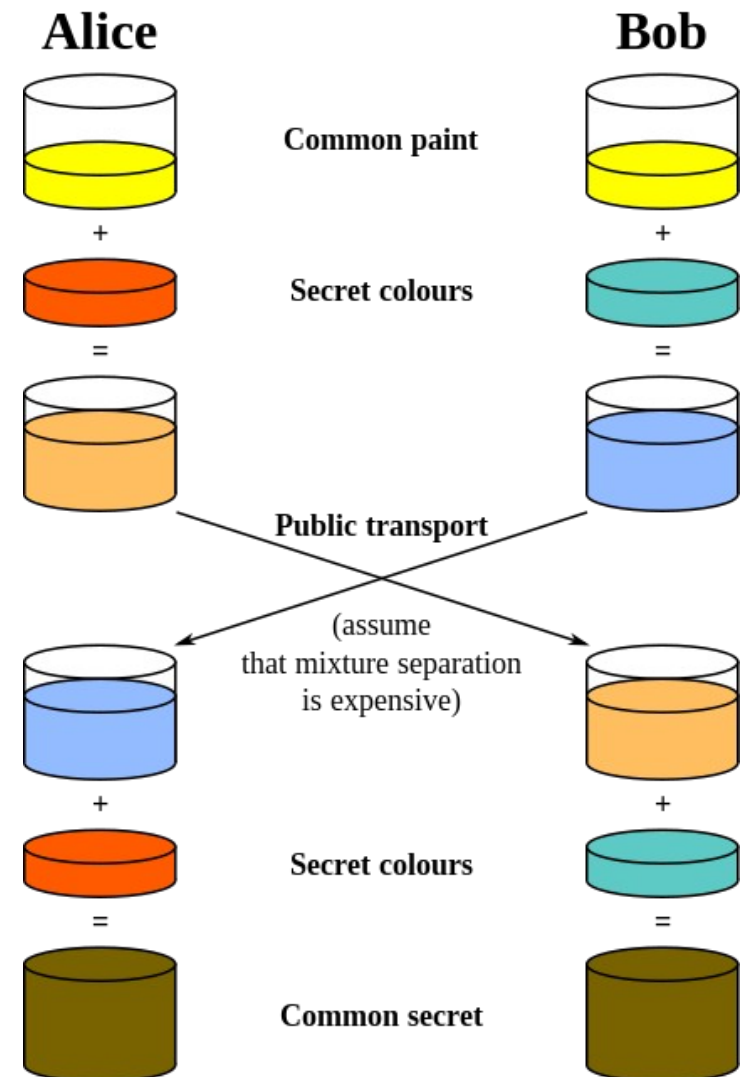
1. Alice and Bob agree to use a modulus $p = 23$ and base $g = 5$
2. Alice chooses a secret integer $a = 6$, then sends Bob $A = g^a \bmod p$
 - $A = 5^6 \bmod 23 = 8$
3. Bob chooses a secret integer $b = 15$, then sends Alice $B = g^b \bmod p$
 - $B = 5^{15} \bmod 23 = 19$
4. Alice computes $s = B^a \bmod p$
 - $s = 19^6 \bmod 23 = 2$
5. Bob computes $s = A^b \bmod p$
 - $s = 8^{15} \bmod 23 = 2$
6. Alice and Bob now share a secret (the number 2).

Both Alice and Bob have arrived at the same value s , because, under mod p ,

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p^{[9]}$$

More specifically,

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$



Security policy enforcement

Authorization

Authorization

- **Access control** mechanisms enforce authorization constraints
 - Internal vs. external access control
 - **Firewalls** prevent external access to a host or internal network
 - Defends against **Denial-of-Service** (DoS) or **distributed DoS** (DDoS) attacks
 - **Access control lists/matrices** track user permissions

user group other
┌───┬───┬───┐
- r w - r - - r - -
↑
directory?

Unix file permissions

```
# file: .  
# owner: studentid  
# group: csmajor  
user:instructorid:rwx  
user:graderid:rwx  
user:studentid:rwx  
group:faculty:r-x  
group:csmajor:---
```

Access control list on stu

Authorization

- A **directory service** provides internal distributed authorization and access control
 - Handles user management, group membership, and password storage
 - Often distributed and/or replicated among multiple servers
 - Lightweight Directory Access Protocol (**LDAP**) for communication
 - Authentication provided by protocols like Kerberos
 - Example: **Active Directory**
- A **single sign-on service** provides authorization for multiple applications or systems
 - Often provides seamless hand-off of an authentication ticket
 - May also use a directory service
 - Examples: **Facebook Connect**, **OAuth**, **OpenID**, **Shibboleth**

Security policy enforcement

Auditing

Auditing

- **Access logs** provide an audit trail for a system
 - Who can access the logs? Who can modify them?
 - Encryption is useful here
 - **Append-only logs** provide guarantees against tampering using checksums and/or cryptographic signing
 - **Bitcoin** (and other **cryptocurrencies**) uses an append-only **blockchain** of cryptographically-signed transactions to preserve financial integrity
 - Demo: <https://andersbrownworth.com/blockchain/blockchain>

