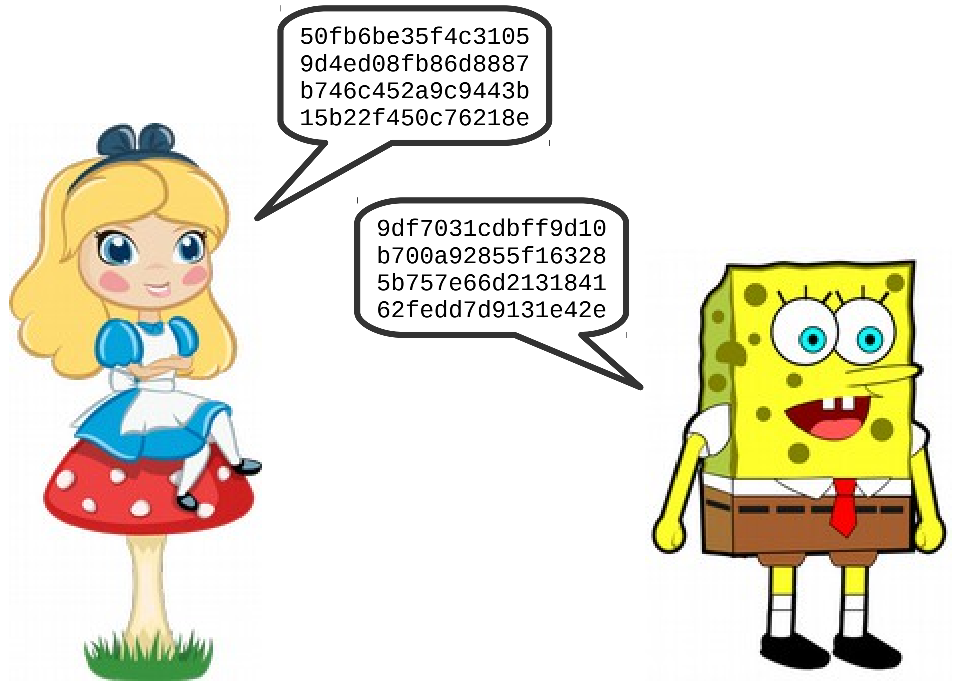
CS 470 Spring 2018

Mike Lam, Professor


50fb6be35f4c3105 9d4ed08fb86d8887 b746c452a9c9443b 15b22f450c76218e
9df7031cdbff9d10 b700a92855f16328 5b757e66d2131841 62fedd7d9131e42e

Security

a.k.a. “Why on earth do Alice and Bob need to share so many secrets?!?”

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

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

Security Solutions

- 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

Distributed security

- **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*

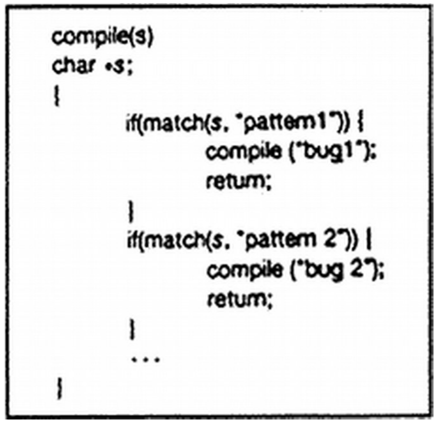
Least privilege

- 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
  - Tradeoff vs. convenience

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)

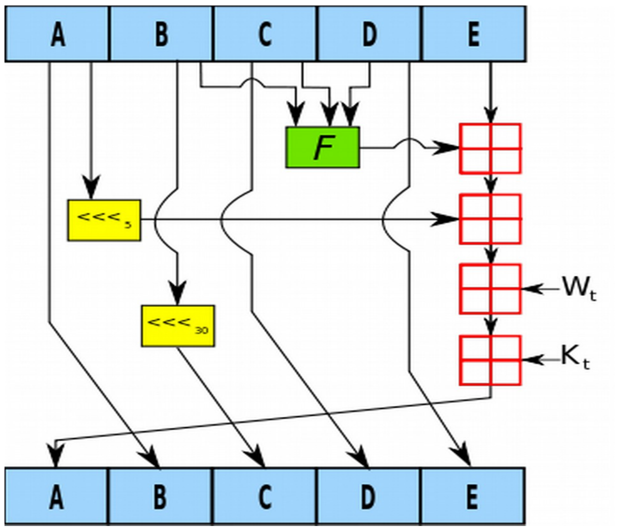
    `https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf`
- Trusted Computing Base (TCB)
  - Minimal component of a system trusted to enforce security policies
  - Sometimes a physically-separate ROM-based processor
  - Hidden encryption key inaccessible to the rest of the system
  - Trusted 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;
        }
        ...
}
```

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;
$<<<_n$ 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}

from `https://en.wikipedia.org/wiki/SHA-1`

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

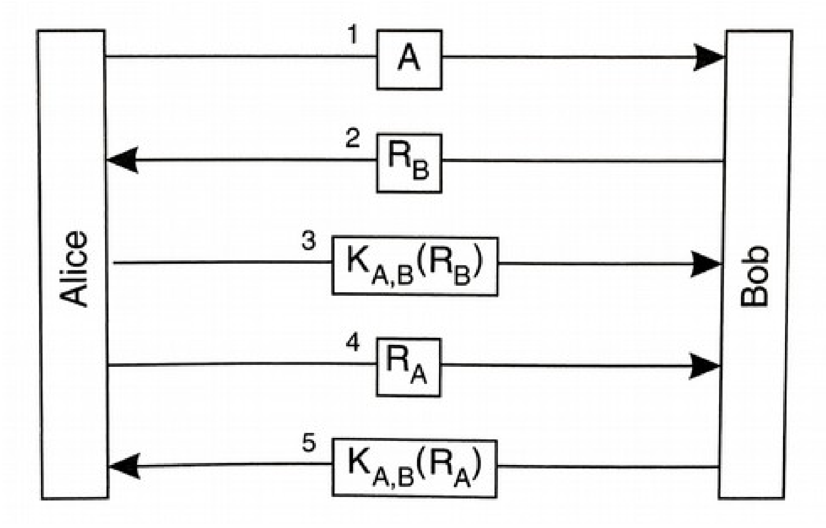
- Symmetric ($P = D_K(E_K(P))$) vs. asymmetric ($P = D_{KD}(E_{KE}(P))$)
  - Same key vs. key pair
  - Private key vs. public/private keys
- Symmetric: Data Encryption Standard (DES)
  - XOR-based operations with different 48-bit keys
  - Fast to encrypt/decrypt, relies on robust secret keys
- Asymmetric: 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

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



from Tanenbaum and Van Steen (Ch. 9)

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

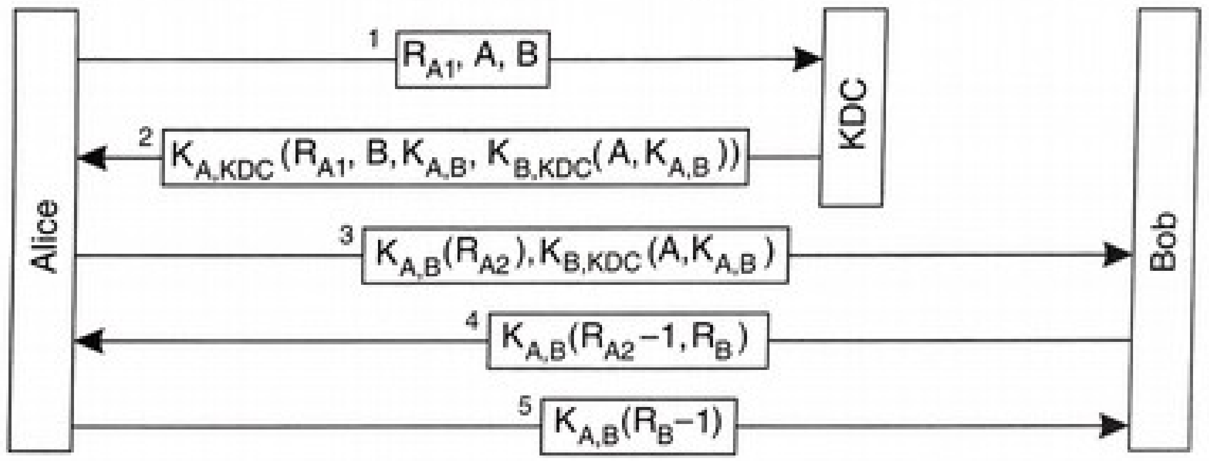

Alice
KDC
Bob
1 R_{A1}, A, B
2 $K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}))$
3 $K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$
4 $K_{A,B}(R_{A2}-1, R_B)$
5 $K_{A,B}(R_B-1)$

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

from Tanenbaum and Van Steen (Ch. 9)

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

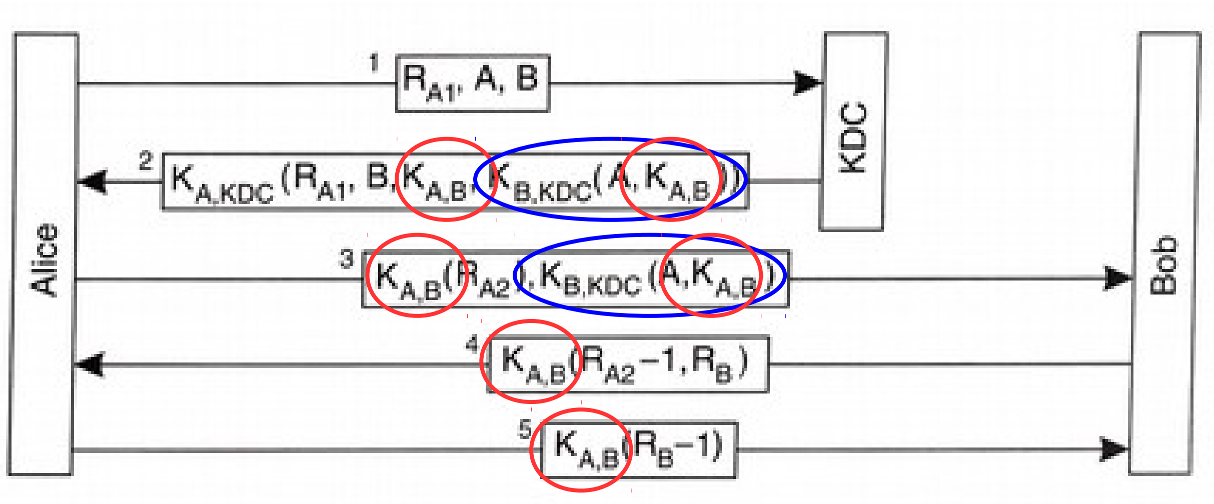

Alice
KDC
Bob
1 R_{A1}, A, B
2 $K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}))$
3 $K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$
4 $K_{A,B}(R_{A2}-1, R_B)$
5 $K_{A,B}(R_B-1)$
shared secret
Bob's copy of shared secret

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

from Tanenbaum and Van Steen (Ch. 9)

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
  - Kerberos is similar, but uses an Authentication Server (AS) to establish identity and a Ticket Granting Server (TGS) to set up shared keys

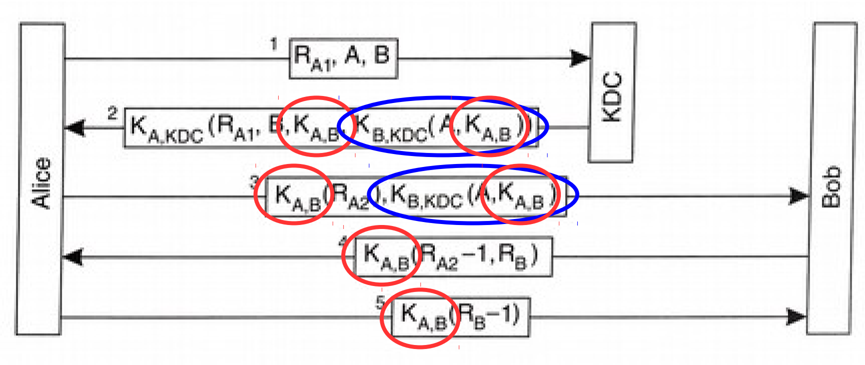

Alice
KDC
Bob
1 R_A1, A, B
2 K_A,KDC(R_A1, B, K_A,B, K_B,KDC(A, K_A,B))
3 K_A,B(R_A2), K_B,KDC(A, K_A,B)
4 K_A,B(R_A2−1, R_B)
5 K_A,B(R_B−1)

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

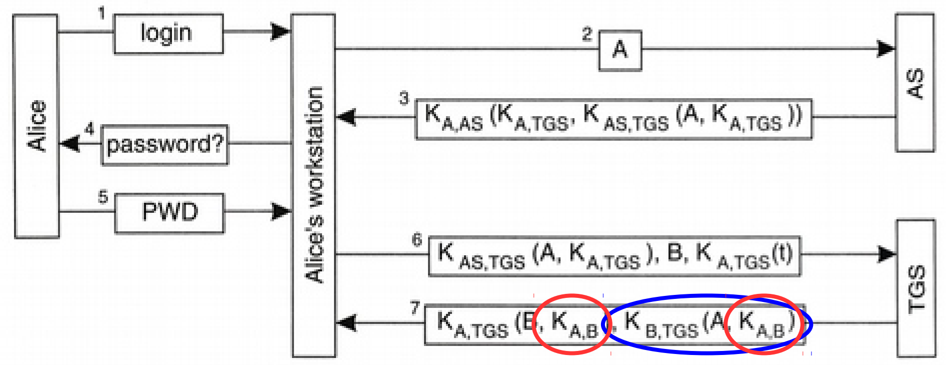
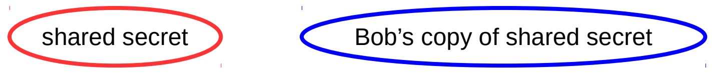

Alice
Alice's workstation
AS
TGS
1 login
2 A
3 K_A,AS(K_A,TGS, K_AS,TGS(A, K_A,TGS))
4 password?
5 PWD
6 K_AS,TGS(A, K_A,TGS), B, K_A,TGS(t)
7 K_A,TGS(B, K_A,B), K_B,TGS(A, K_A,B)

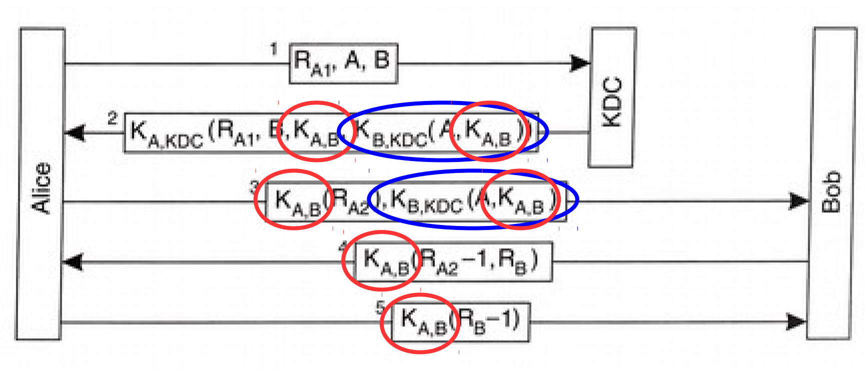
Figure 9-23. Authentication in Kerberos.



from Tanenbaum and Van Steen (Ch. 9)

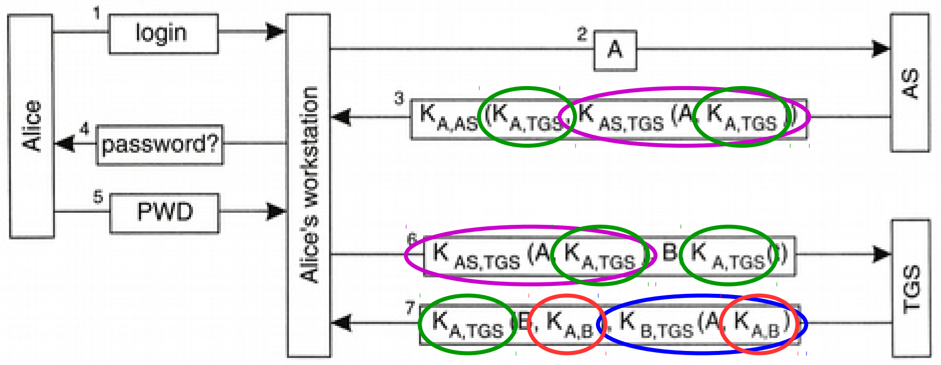
Needham-Schroeder authentication

- 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
  - Kerberos is similar, but uses an Authentication Server (AS) to establish identity and a Ticket Granting Server (TGS) to set up shared keys



Alice
KDC
Bob
1 R_{A1}, A, B
2 $K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}))$
3 $K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$
4 $K_{A,B}(R_{A2}-1, R_B)$
5 $K_{A,B}(R_B-1)$

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



Alice
Alice's workstation
AS
TGS
1 login
2 A
3 $K_{A,AS}(K_{A,TGS}, K_{AS,TGS}(A, K_{A,TGS}))$
4 password?
5 PWD
6 $K_{AS,TGS}(A, K_{A,TGS}), B, K_{A,TGS}(t)$
7 $K_{A,TGS}(B, K_{A,B}), K_{B,TGS}(A, K_{A,B})$

Figure 9-23. Authentication in Kerberos.

shared secret

Bob's copy of shared secret

session key

ticket

from Tanenbaum and Van Steen (Ch. 9)

Kerberos

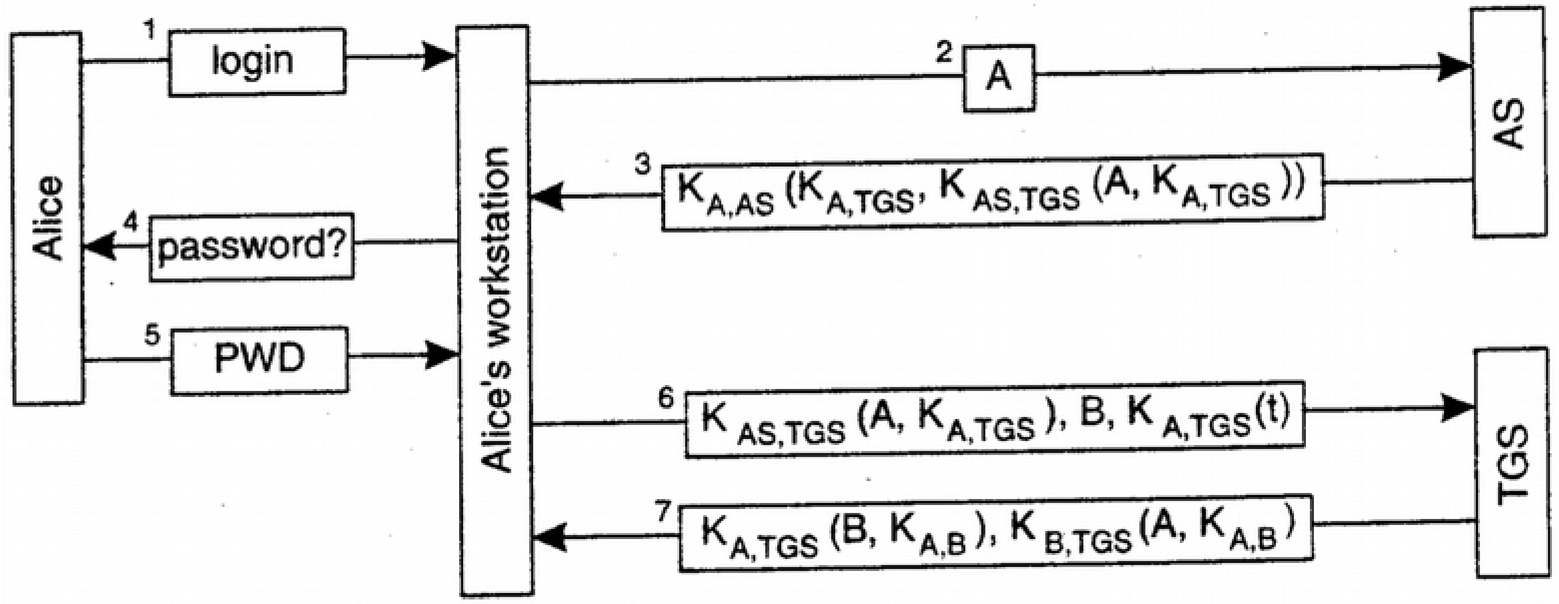

Alice
Alice's workstation
AS
TGS
1 login
2 A
3 $K_{A,AS}(K_{A,TGS}, K_{AS,TGS}(A, K_{A,TGS}))$
4 password?
5 PWD
6 $K_{AS,TGS}(A, K_{A,TGS}), B, K_{A,TGS}(t)$
7 $K_{A,TGS}(B, K_{A,B}), K_{B,TGS}(A, K_{A,B})$

Figure 9-23. Authentication in Kerberos.

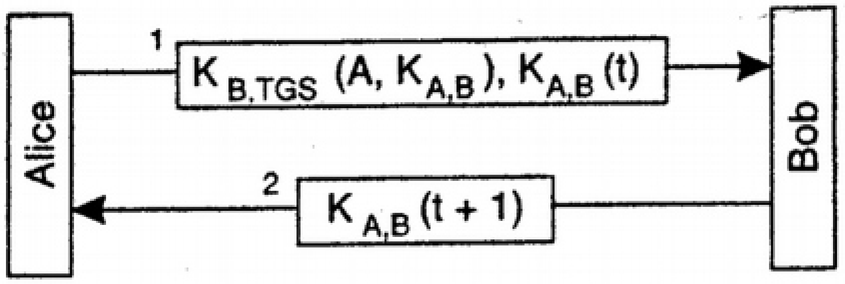


Figure 9-24. Setting up a secure channel in Kerberos.

from Tanenbaum and Van Steen (Ch. 9)

Public keys

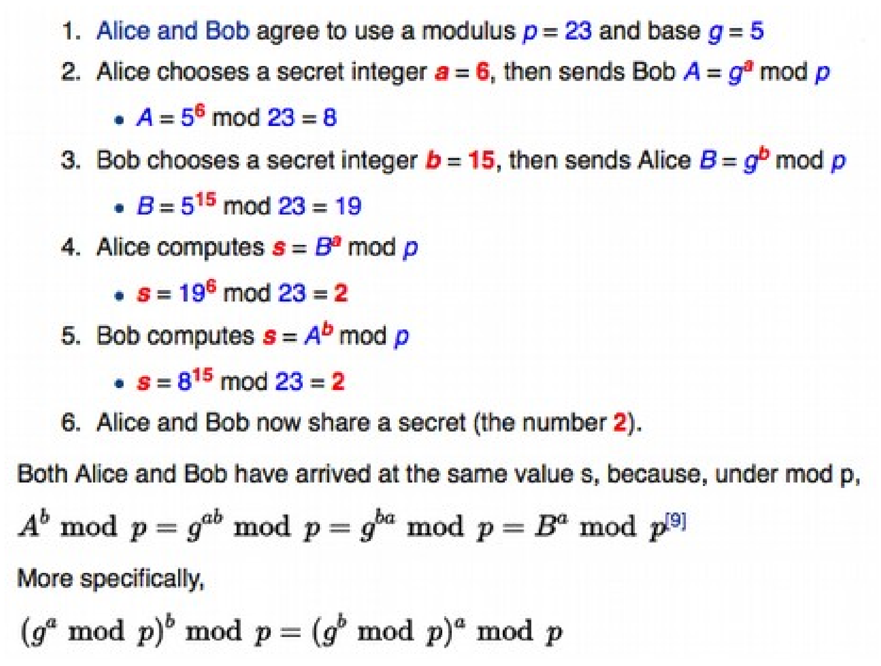
- Private keys are used to sign documents by encrypting them
- A certificate is a signed document claiming to own a public key
  - Only the public key can decrypt the document, proving it was encrypted using the corresponding private key
- At a key signing party, participants exchange public keys
  - 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

Diffie-Helman key exchange

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



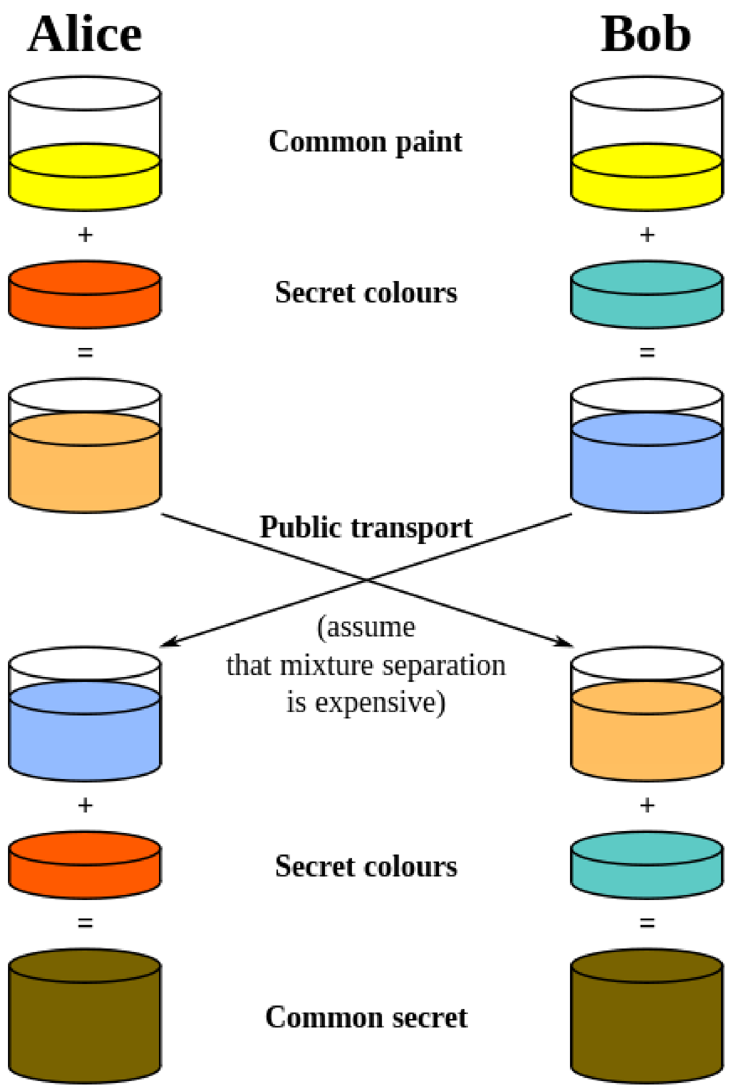
1. Alice and Bob agree to use a modulus $p = 23$ and base $g = 5$
2. Alice chooses a secret integer $a = 6$, then sends Bob $A = g^a \bmod p$
   - $A = 5^6 \bmod 23 = 8$
3. Bob chooses a secret integer $b = 15$, then sends Alice $B = g^b \bmod p$
   - $B = 5^{15} \bmod 23 = 19$
4. Alice computes $s =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

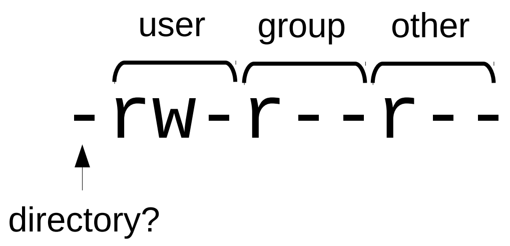
$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$



from `https://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange`

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



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

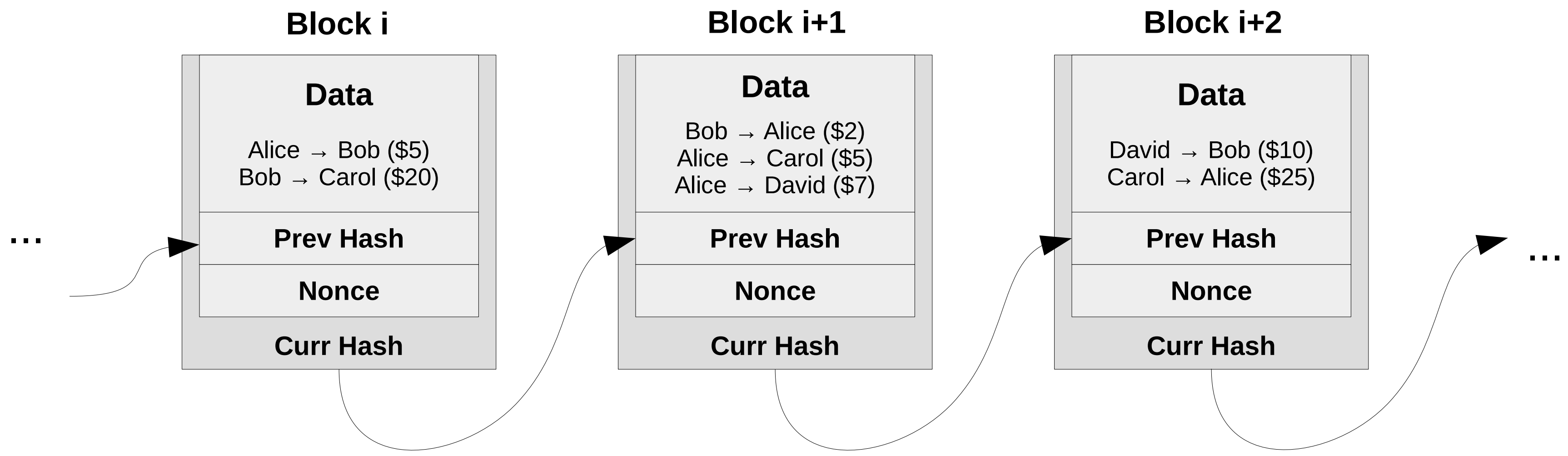
Authorization

- A directory service provides internal distributed authorization and access control
  - Handles user management/permissions 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 LDAP and/or Kerberos
  - Examples: Facebook Connect, OAuth, OpenID, Shibboleth

Auditing

- Access logs provide an audit trail for a system
  - Who can access the logs? Who can modify them?
  - 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://anders.com/blockchain/blockchain.html`



Block i
Data
Alice → Bob ($5)
Bob → Carol ($20)
Prev Hash
Nonce
Curr Hash
Block i+1
Data
Bob → Alice ($2)
Alice → Carol ($5)
Alice → David ($7)
Prev Hash
Nonce
Curr Hash
Block i+2
Data
David → Bob ($10)
Carol → Alice ($25)
Prev Hash
Nonce
Curr Hash
…
…