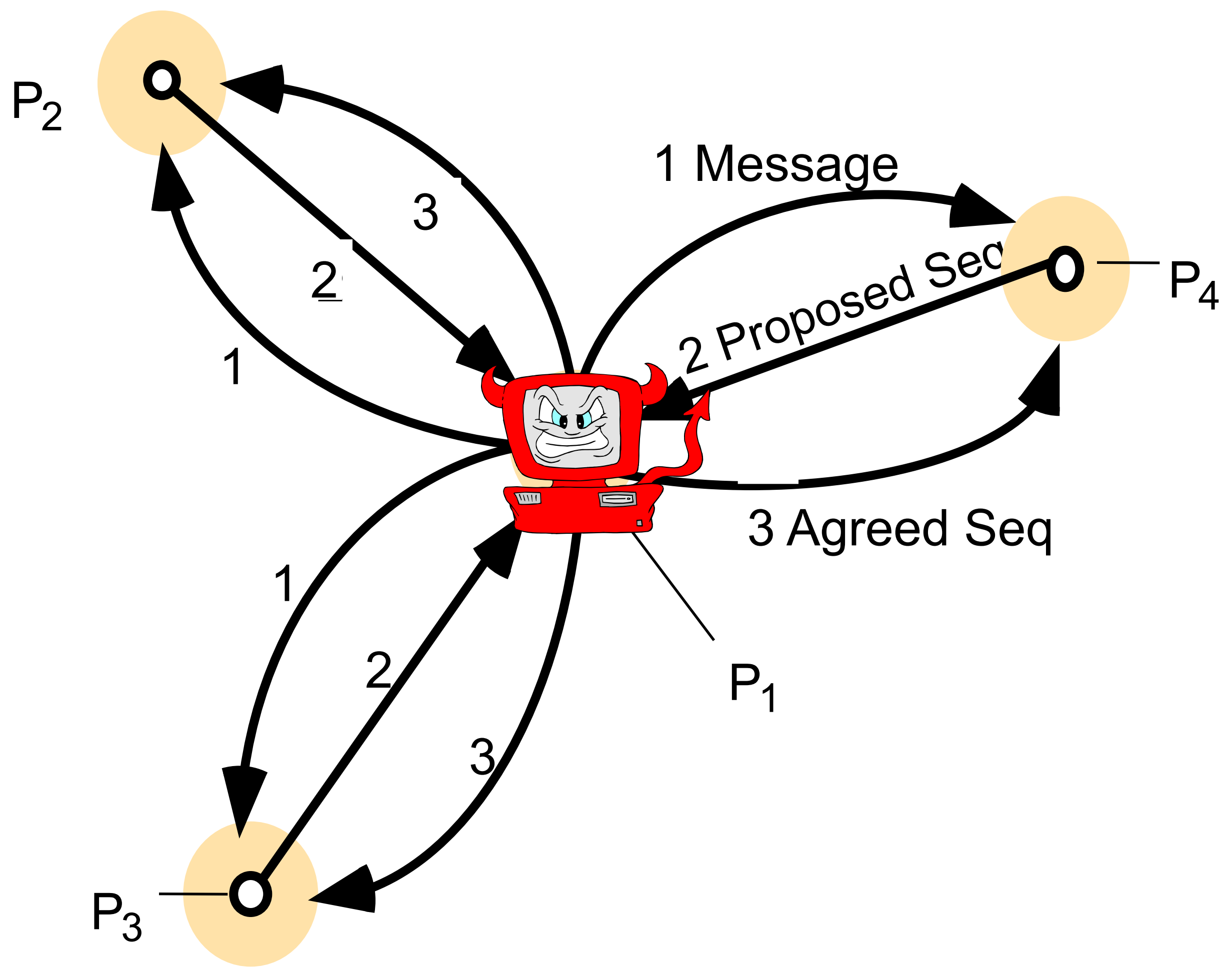
CS425/CSE424/ECE428 – Distributed Systems

Security in Distributed systems

Some material derived from slides by I. Gupta, M. Harandi, J. Hou, S. Mitra, K. Nahrstedt, N. Vaidya

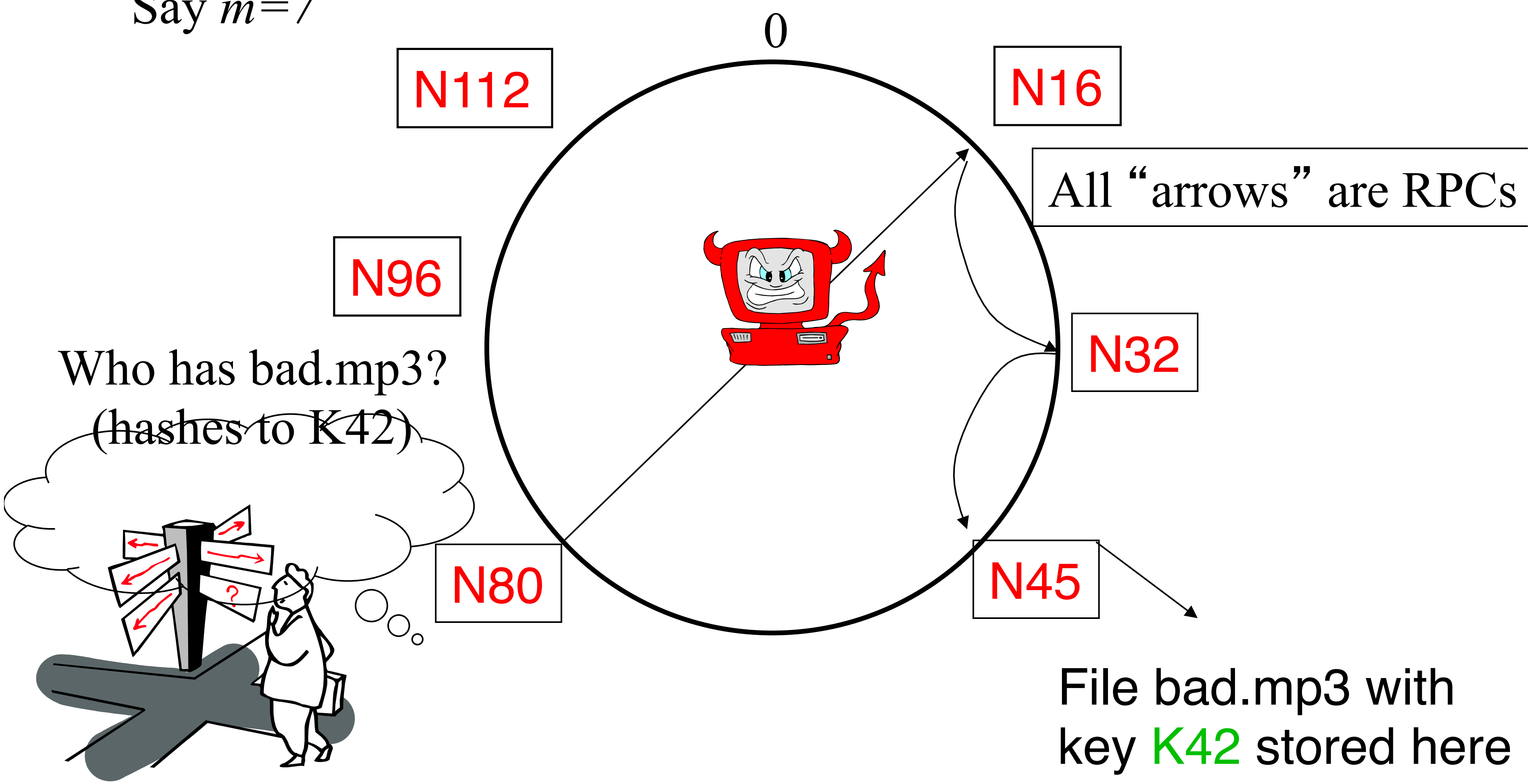
ISIS algorithm for total ordering



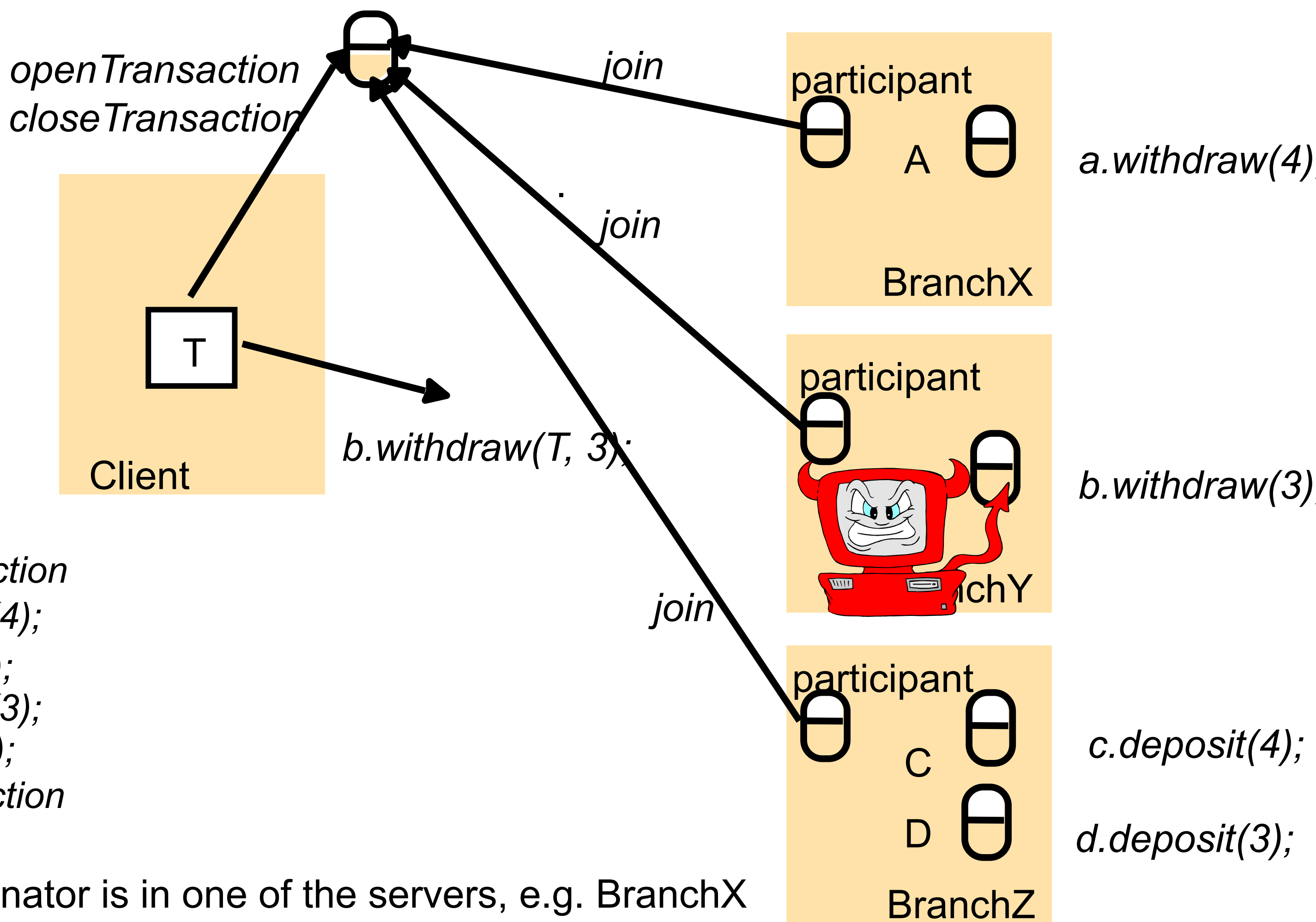
Chord: client to client

At node n, send query for key k to largest successor/finger entry $< k$
if none exist, return *successor(n)* to requestor

Say $m=7$



Distributed banking transaction




```
T = openTransaction
      a.withdraw(4);
      c.deposit(4);
      b.withdraw(3);
      d.deposit(3);
    closeTransaction
```

Note: the coordinator is in one of the servers, e.g. BranchX

Security Threats

- **Leakage:** An unauthorized party gains access to a service or data.
  - Attacker obtains knowledge of a withdrawal or account balance, e.g., via eavesdropping
- **Tampering:** Unauthorized change of data, tampering with a service
  - Attacker changes the variable holding your personal checking $$ total
- **Vandalism:** Interference with proper operation, without gain to the attacker
  - Attacker does not allow any transactions to your account
  - E.g., DOS=denial of service

More Concerns

Attacks on Communication Channel / Network

- **Eavesdropping** – Obtaining copies of messages without authority.
- **Masquerading** – Sending or receiving messages with the identity of another principal (user or corporation).
- **Message tampering** – Intercepting messages and altering their contents before passing them onto the intended recipient.
- **Replaying** – Intercepting messages and sending them at a later time.
- **Denial of Service Attack** – flooding a channel or other resources (e.g., port) with messages.

Addressing the Challenges: Security

- **Leakage:** An unauthorized party gains access to a service or data.
  - Confidentiality : protection against disclosure to unauthorized individuals.
- **Tampering:** Unauthorized change of data, tampering with a service
  - Integrity : protection against alteration or corruption.
- **Vandalism:** Interference with proper operation, without gain to the attacker
  - Availability : protection against interference with the means to access the resources.

Security Policies & Mechanisms

- A Security ***Policy*** indicates which actions each entity (user, data, service) is allowed or prohibited to take.
  - E.g., Only an owner is allowed to make transactions to his account. CIA properties.
- A Security ***Mechanism*** enforces the policy
  - *Encryption and decryption:* transform data to a form only understandable by authorized users, and vice-versa.
  - *Authentication:* verify the claimed identity of a user, client, service, process, etc.
  - *Authorization:* verify access rights for an authenticated entity.
  - *Auditing*: make record of and check access to data and resources. Mainly an offline analysis tool, often after the fact.

Security Tenets

- Make worst-case assumptions
  - Network compromised
  - Code / mechanism is known
  - Nothing remains secret forever
- Separate policy from mechanism
  - Cryptography for secure channels
  - Identity management (PKI, passwords, etc.) for authentication
  - Access control lists, capabilities for authorization

Cryptography

- Science of enciphering data
  - Cryptology (algorithm design) + cryptanalysis (breaking algorithms)
- History
  - First algorithms thousands of years old
  - Encryption driven by military, intelligence, and financial uses
  - Since 1970's, subject of much open research
  - Backbone of most Internet security mechanisms

Encryption (symmetric)

- Block cipher:
  - $E_K(P) = C$
  - $D_K(C) = P$
  - P: Plaintext
  - C: Ciphertext
  - K: Shared key
- Example: AES
  - Result of design competition by NIST
  - AES-128: key, block size are 128 bits
  - Also, AES-192, AES-256

Encryption (symmetric)

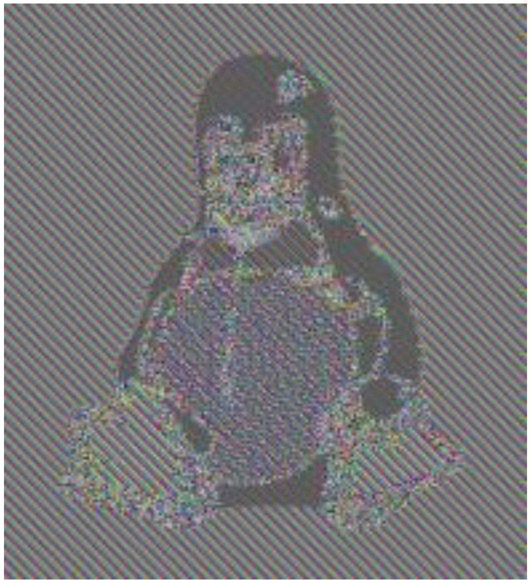
- Stream cipher:
  - Keystream(K)
    - Produce infinite, unpredictable key stream from key K
  - C = P xor Keystream(K)
  - P = C xor Keystream(K)
- Example: RC4
  - Used in older version of 802.11, SSL
  - Some security vulnerabilities

Security Properties

- Indistinguishability
  - Adversary queries encryption, decryption oracles
    - $E_K(.)$, $D_K(.)$
    - Polynomial # of times
  - Adversary provides M_1, M_2
  - Challenger provides $E_K(M_b)$ for $b = 0$ or 1
  - Adversary queries oracles again
  - Outputs guess for b
- Security
  - Adversary can't win with probability (non-negligibly) more than 1/2

Encryption mode

- Basic encryption primitives insecure
  - Block cipher: C = C′ => P = P′
  - Stream cipher: C xor C′ = P xor P′
- Must use operation mode
  - E.g., CBC
  - C_1 = IV (random)
  - $C_2 = E_K(P_1 \text{ xor } C_1)$
  - $C_3 = E_K(P_2 \text{ xor } C_2)$
  - …



Secure channel

- Alice, Bob share key K
  - Each sends $E_K(M)$ to send M over secure channel
- Security properties?
  - Confidentiality
    - Guaranteed by security of E
  - Integrity
    - Not guaranteed
  - Availability
    - Cannot be guaranteed by cryptography

Integrity Protection

- Message Authentication Code (MAC)
  - aka Message Integrity Code (MIC)
- $MAC_K(M) = x$
- Security: unforgeability
  - Adversary queries MAC oracle
    - $MAC_K(.)$
  - Adversary produces (M,x) where M has never been queried
  - Wins if $MAC_K(M) = x$
  - Secure if adversary cannot win with probability non-negligibly more than 0
- Examples: HMAC, CBC-MAC

Secure Channel

- Encryption key EK, MAC key MK
- Send(M) = $E_{EK}(M) \| MAC_{MK}(M)$
- Secure?
  - Replay
  - Reflection
- Solution:
  - Sequence numbers
  - Different keys in different directions

Public-key cryptography

- Must establish symmetric key with everyone
  - $O(N^2)$ keys total
  - Must be exchanged over secure channel!
- Public key cryptography
  - Two keys: PK – public, SK – secret
  - $C = E_{PK}(P)$
  - $P = D_{SK}(C)$
  - $O(N)$ keys total

RSA

- Example: RSA
  - Rivest, Shamir, Adleman, 1977
- Key generation
  - N = p*q, for two large primes p
  - $e = 3$, $d = e^{-1}$ in Z_N^*
    - d can be computed with knowledge of p, q
  - PK = (N, e), SK = d
    - Factoring N into p,q currently infeasible if p,q > ~1024 bits
- Encryption
  - $C = M^e \pmod{N}$
  - $P = C^d \pmod{N}$
- Note: insecure in this form
  - Must use randomization, padding to ensure indistinguishability

Key exchange

- RSA-based key exchange
  - (roughly what's used in TLS)
- Parties: Client, Server
- Steps:
  - S->C: PK_S, N_S
  - C->S: $E_{PKS}(N_C)$
  - $K = H(N_S||N_C)$
    - Encryption, MAC keys derived from K
- Properties:
  - Nonces protect from replay
  - One-way authentication
  - No PFS

Perfect Forward Secrecy

- Goal: if (long-term) keys uncompromised at end of session, session remains secure forever
- E.g., Diffie-Hellman
  - S: pick random x, send g^x
  - C: pick random y, send g^y
  - Use $(g^x)^y = (g^y)^x = g^{xy}$ to derive shared key
  - Securely forget secrets (incl. x,y, g^{xy}) after session
- Security relies on discrete logarithm problem

Digital Signatures

- Public-key algorithm
  - Secret signing key SK
  - Public verification key VK
- Operation
  - sig = $\text{Sign}_{SK}(M)$
  - $\text{Verify}_{VK}(M,\text{sig})$ = True or False
- Example: RSA
  - N,e = verification key, d = signature key
  - $\text{Sign}(M) = H(M)^d \pmod N$

Authenticated Key Exchange

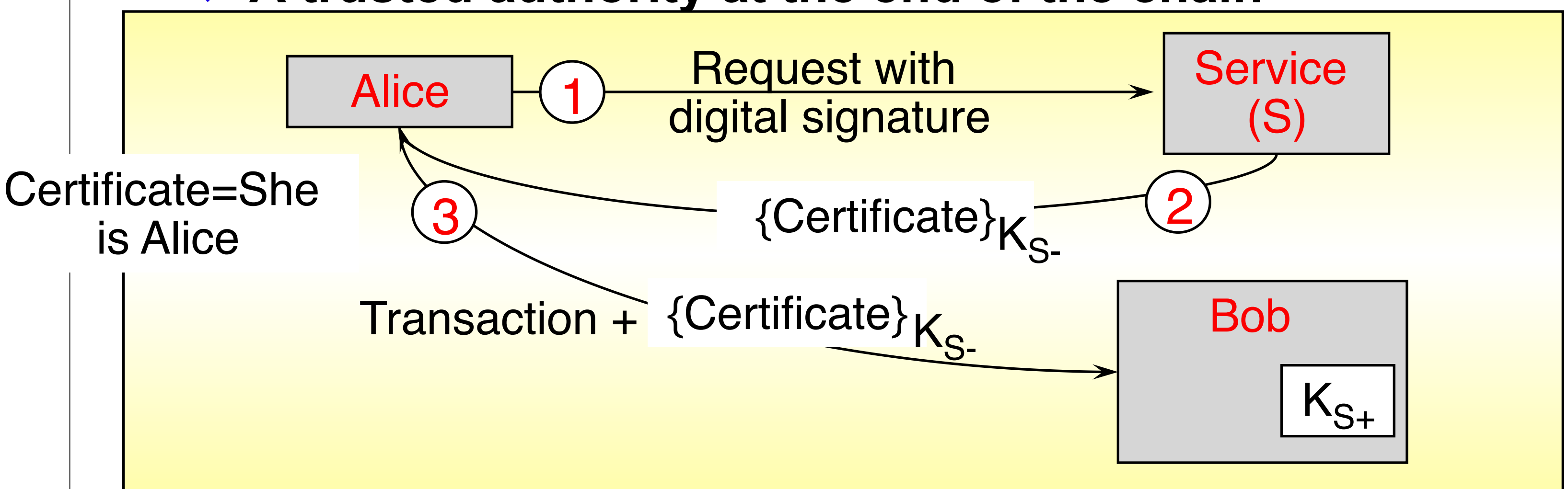
- Putting things together:
  - A->B: A, g^x, Sign(g^x)
  - B->A: B, g^y, Sign(g^y)
- Problems?

SIGMA protocol

- SIGn-and-MAc, due to Hugo Krawczyk
- Used in IKE, part of IPSec
  - A->B: g^x
  - B->A: g^y, Sign(g^x,g^y), MAC_{MK}(B)
  - A->B: A, Sign(g^y,g^x),MAC_{MK}(A)

Digital Certificates

- A **digital certificate** is a statement signed by a third party principal, and can be reused
  - e.g., Verisign Certification Authority (CA)
- To be useful, certificates must have:
  - A standard format, for construction and interpretation
  - A protocol for constructing <u>chains</u> of certificates
  - A trusted authority at the end of the chain



Alice's Bank Account Certificate

1. *Certificate type*	Account number
2. *Name*	Alice
3. *Account*	6262626
4. *Certifying authority*	Bob's Bank
5. *Signature*	$\{Digest(field\ 2 + field\ 3)\}_{K_{Bpriv}}$

Public-Key Certificate for Bob's Bank

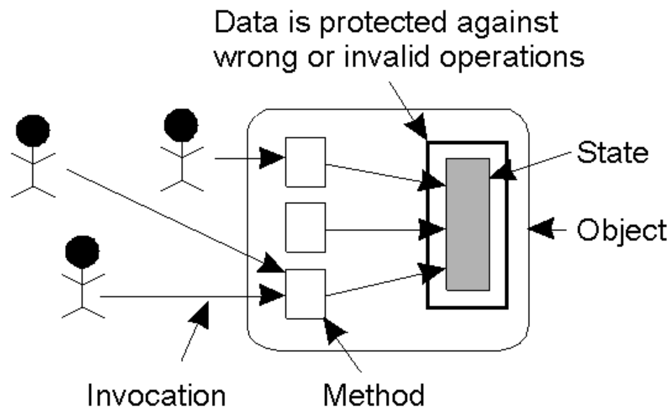
1. *Certificate type*	Public key
2. *Name*	Bob's Bank
3. *Public key*	K_{Bpub}
4. *Certifying authority*	Fred – The Bankers Federation
5. *Signature*	$\{Digest(field\ 2 + field\ 3)\}\ K_{Fpriv}$

Eventually K_F-, K_F+ have to be obtained reliably.

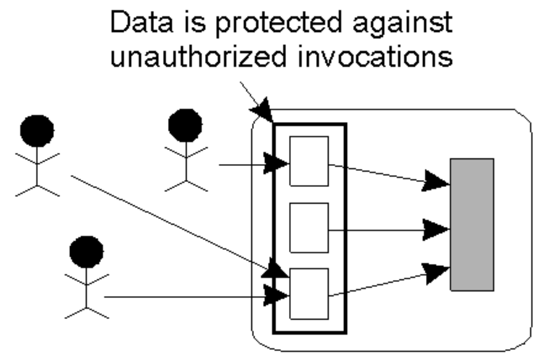
Authorization: Access Control

- Control of access to resources of a server.
- A basic form of access control checks <principal, op, resource> requests for:
  - Authenticates the principal.
  - Authorization check for desired op, resource.
- Access control matrix M (e.g., maintained at server)
  - Each principal is represented by a row, and each resource object is represented by a column.
  - M[s,o] lists precisely what operations principal s can request to be carried out on resource o.
  - Check this before carrying out a requested operation.
  - M may be sparse.
- Access control list (ACL)
  - Each object maintains a list of access rights of principals, i.e., an ACL is some column in M with the empty entries left out.
- Capability List = row in access control matrix, i.e., per-principal list.

Focus of Access Control



(a)



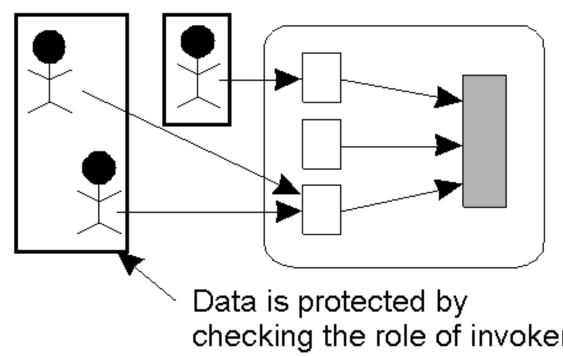
(b)

- **Three approaches for protection against security threats**

a) Protection against invalid operations

b) Protection against unauthorized invocations

c) Protection against unauthorized users



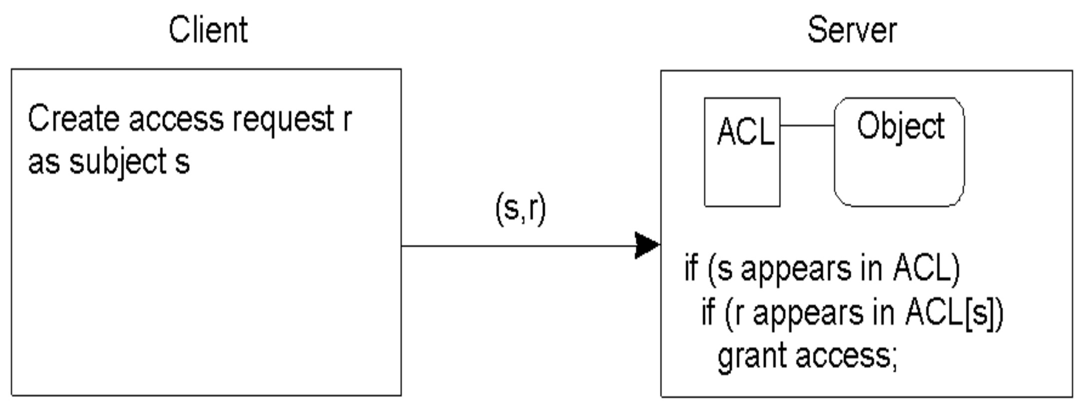
(c)

ACL and Capability Usage

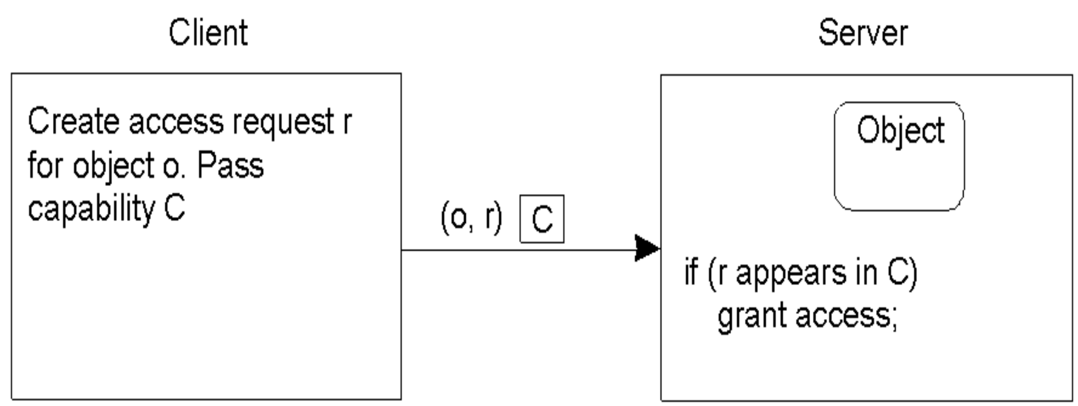
Comparison between ACLs and capabilities for protecting objects.

a) Using an ACL

b) Using capabilities.



(a)



(b)