CS 425 / ECE 428 Distributed Systems Fall 2014

Indranil Gupta (Indy)

Lecture 24B: Security

Security Threats

- **Leakage**
  - Unauthorized access to service or data
  - E.g., Someone knows your bank balance
- **Tampering**
  - Unauthorized modification of service or data
  - E.g., Someone modifies your bank balance
- **Vandalism**
  - Interference with normal service, without direct gain to attacker
  - E.g., Denial of Service attacks

Common Attacks

- **Eavesdropping**
  - Attacker taps into network
- **Masquerading**
  - Attacker pretends to be someone else, i.e., identity theft
- **Message tampering**
  - Attacker modifies messages
- **Replay attack**
  - Attacker replays old messages
- **Denial of service**: bombard a port

Addressing the Challenges: CIA Properties

- Confidentiality
  - *Protection against disclosure to unauthorized individuals*
  - Addresses Leakage threat
- Integrity
  - *Protection against unauthorized alteration or corruption*
  - Addresses Tampering threat
- Availability
  - *Service/data is always readable/writable*
  - Addresses Vandalism threat

Policies vs. Mechanisms

- Many scientists (e.g., Hansen) have argued for a separation of policy vs. mechanism
- A security policy indicates *what* a secure system accomplishes
- A security mechanism indicates *how* these goals are accomplished
- E.g.,
  - Policy: in a file system, only authorized individuals allowed to access files (i.e., CIA properties)
  - Mechanism: Encryption, capabilities, etc.

Mechanisms: Golden A's

- **Authentication**
  - Is a user (communicating over the network) claiming to be Alice, really Alice?
- **Authorization**
  - Yes, the user is Alice, but is she allowed to perform her requested operation on this object?
- **Auditing**
  - How did Eve manage to attack the system and breach defenses? Usually done by continuously logging all operations.

Designing Secure Systems

- Don’t know how powerful attacker is
- When designing a security protocol need to

1. Specify Attacker Model: Capabilities of attacker

   (Attacker model should be tied to reality)

2. Design security mechanisms to satisfy policy under the attacker model

3. Prove that mechanisms satisfy policy under attacker model

4. Measure effect on overall performance (e.g., throughput) in the common case, i.e., no attacks

Next

- Basic Cryptography

Basic Security Terminology

- **Principals**: processes that carry out actions on behalf of users
  - Alice
  - Bob
  - Carol
  - Dave
  - Eve (typically evil)
  - Mallory (typically malicious)
  - Sara (typically server)

Keys

- Key = sequence of bytes assigned to a user
  - Can be used to “lock” a message, and only this key can be used to “unlock” that locked message

Encryption

- Message (sequence of bytes) + Key →
  (Encryption) →
  Encoded message (sequence of bytes)
- Encoded Message (sequence of bytes) + Key →
  (Decryption) →
  Original message (sequence of bytes)
- No one can decode an encoded message without the key

Two Cryptography Systems

I. Symmetric Key systems:

- K_A = Alice’s key; secret to Alice
- K_{AB} = Key shared only by Alice and Bob
- Same key (K_{AB}) used to both encrypt and decrypt a message

- E.g., DES (Data Encryption Standard): 56 b key operates on 64 b blocks from the message

Two Cryptography Systems (2)

II. Public-Private Key systems:

- K_{Apriv} = Alice's private key; known only to Alice
- K_{Apub} = Alice's public key; known to *everyone*
- Anything encrypted with K_{Apriv} can be decrypted only with K_{Apub}
- Anything encrypted with K_{Apub} can be decrypted only with K_{Apriv}

- RSA and PGP fall into these category
  - RSA = Rivest Shamir Adleman
  - PGP = Pretty Good Privacy
  - Keys are several 100s or 1000s of b long
  - Longer keys => harder for attackers to break
  - Public keys maintained via PKI (Public Key Infrastructure)

Public-Private Key Cryptography

- If Alice wants to send a secret message M that can be read only by Bob
  - Alice encrypts it with Bob's public key
  - $K_{Bpub}(M)$
  - Bob only one able to decrypt it
  - $K_{Bpriv}(K_{Bpub}(M)) = M$
  - Symmetric too, i.e., $K_{Apub}(K_{Apriv}(M)) = M$

Shared/Symmetric vs. Public/Private

- Shared keys reveal too much information
  - Hard to *revoke* permissions from principals
  - E.g., group of principals shares one key

    → want to remove one principal from group

      → need everyone in group to change key
- Public/private keys involve costly encryption or decryption
  - At least one of these 2 operations is costly
- Many systems use public/private key system to generate shared key, and use latter on messages

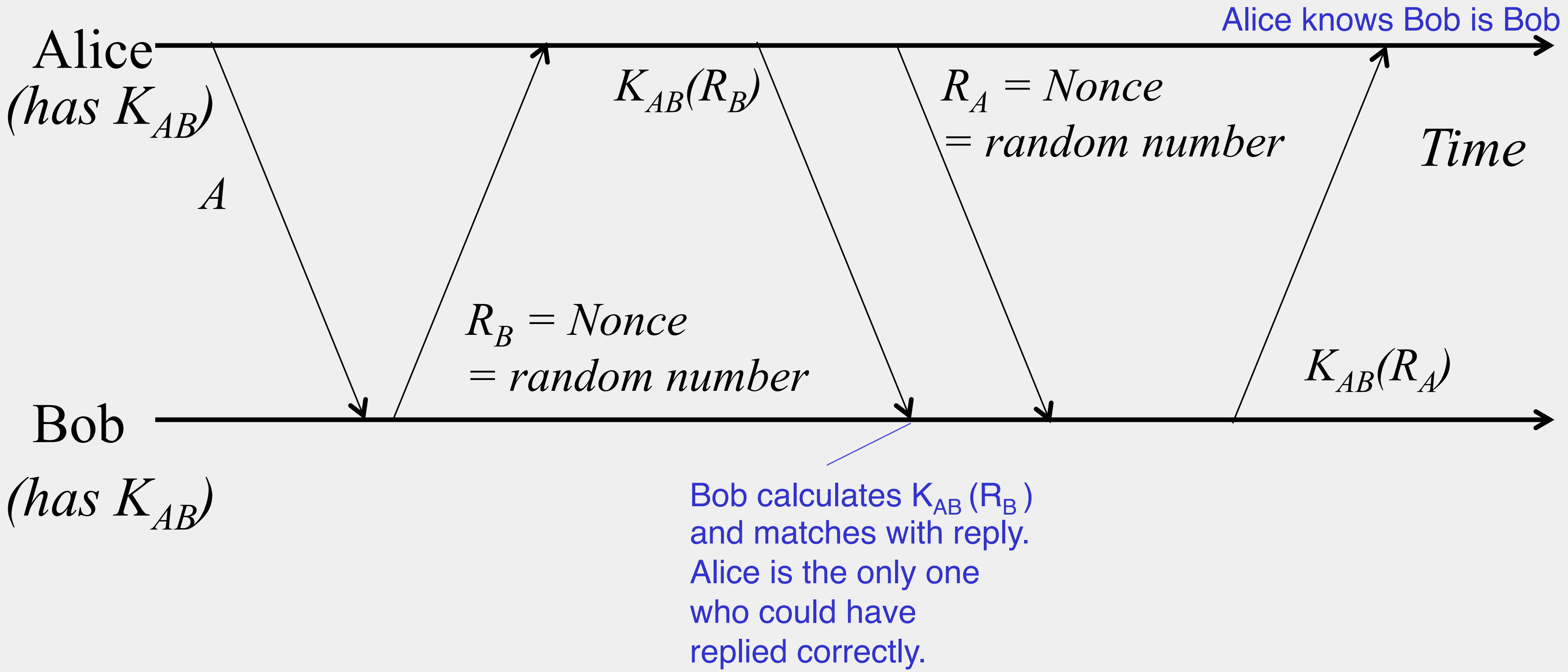
Next

- How to use cryptography to implement
    I. Authentication
    II. Digital Signatures
    III. Digital Certificates

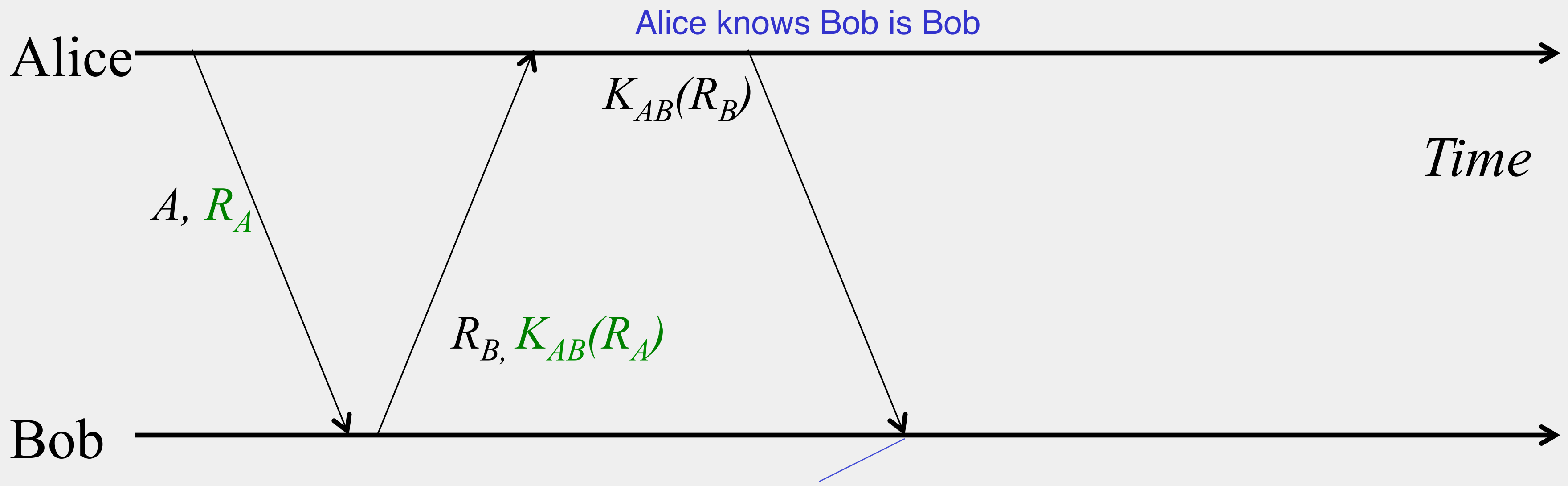
I. Authentication

- Two principals verify each others' identities
- Two flavors
  - **Direct authentication**: directly between two parties
  - **Indirect authentication**: uses a trusted third-party server
    - Called authentication server
    - E.g., A Verisign server

Direct Authentication Using Shared Key

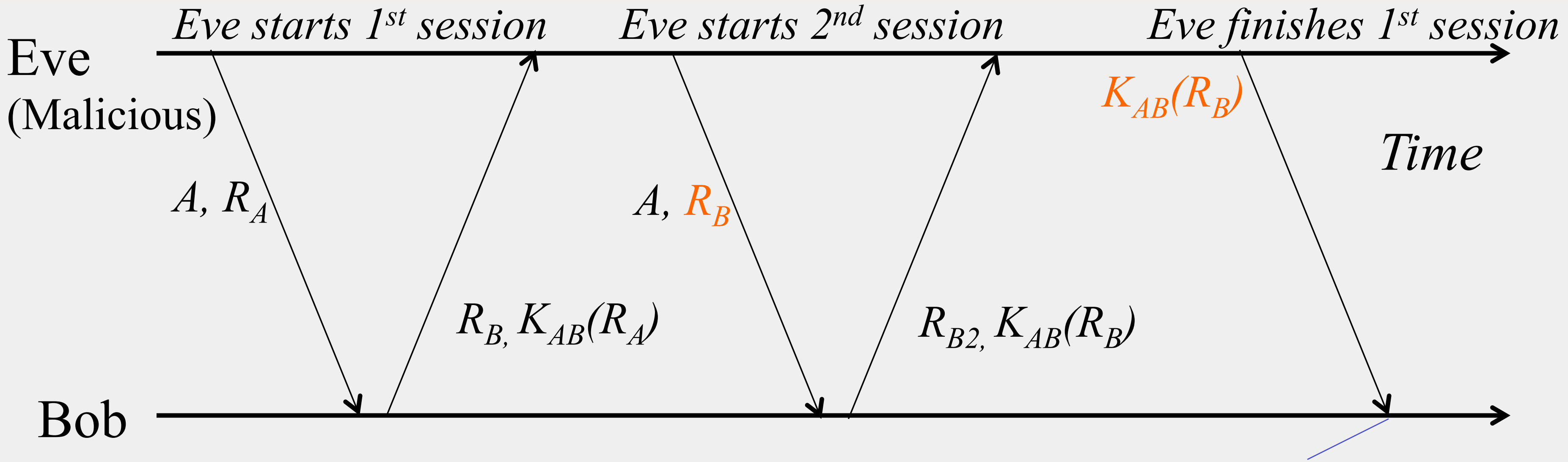


Why Not Optimize Number of Messages?



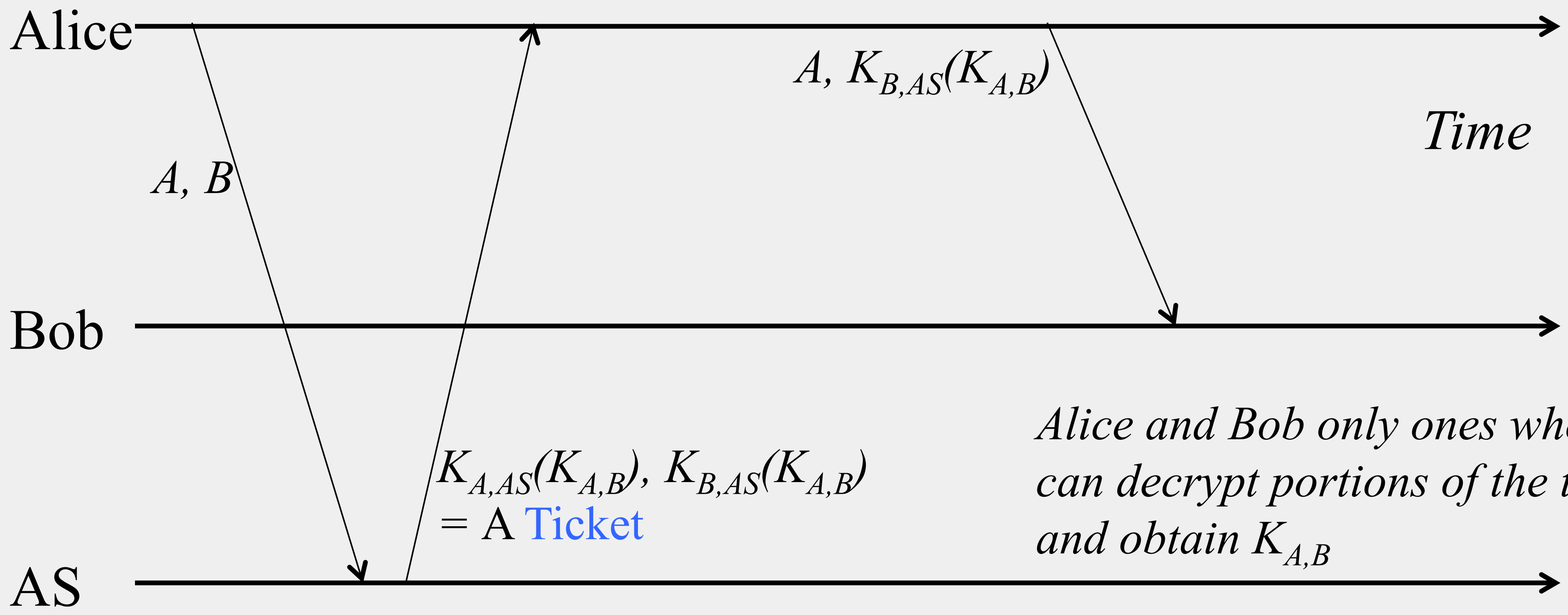
Bob calculates $K_{AB}(R_B)$ and matches with reply. Alice is the only one who could have replied correctly.

Unfortunately, This Subject to Replay Attack



Bob calculates $K_{AB}(R_B)$ and matches with reply. Bob thinks Eve is Alice.

Indirect Authentication Using Authentication Server and Shared Keys



Alice and Bob only ones who can decrypt portions of the ticket and obtain $K_{A,B}$

II. Digital Signatures

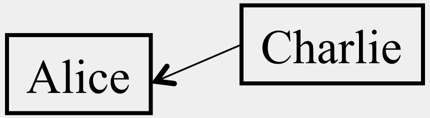
- Just like “real” signatures
  - Authentic, Unforgeable
  - Verifiable, Non-repudiable
- To sign a message M, Alice encrypts message with her own private key
  - Signed message: [M, K_{Apriv}(M)]
  - Anyone can verify, using Alice’s public key, that Alice signed it
- To make it more efficient, use a one-way hash function, e.g., SHA-1, MD-5, etc.
  - Signed message: [M, K_{Apriv}(Hash(M))]
  - Efficient since hash is fast and small; don’t need to encrypt/ decrypt full message

III. Digital Certificates

- Just like “real” certificates
- Implemented using digital signatures
- Digital Certificates have
  - Standard format
  - Transitivity property, i.e., chains of certificates
  - Tracing chain backwards must end at trusted authority (at root)

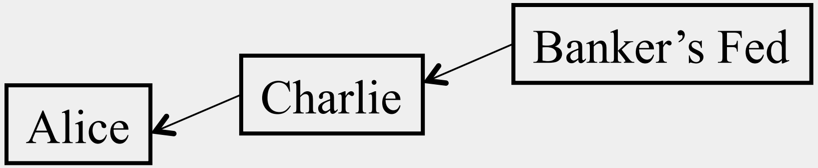
Example: Alice's Bank Account

1. Certificate Type: Account
2. Name: Alice
3. Account number: 12345
4. Certifying Authority: Charlie's Bank
5. Signature
   - K_{Cpriv}(Hash(Name+Account number))



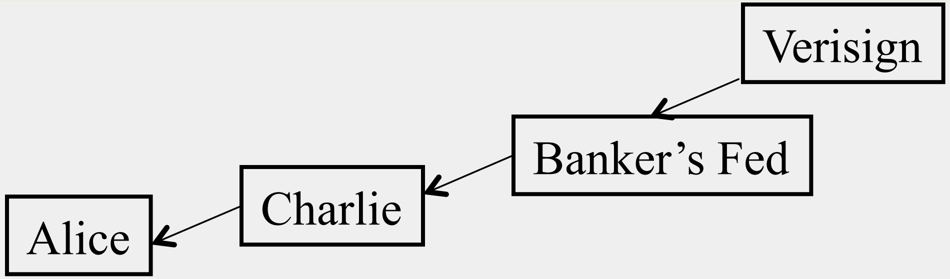
Charlie's Bank, in Turn has another Certificate

1. Certificate Type: Public Key
2. Name: Charlie’s Bank
3. Public Key: K_{Cpub}
4. Certifying Authority: Banker’s Federation
5. Signature
   – K_{Fpriv}(Hash(Name+Public key))



Banker's Federation, Has Another Certificate From the Root Server

1. Certificate Type: Public Key
2. Name: Banker’s Federation
3. Public Key: K_{Fpub}
4. Certifying Authority: Verisign
5. Signature
   - $K_{verisign\ priv}$(Hash(Name+Public key))



IV. Authorization

- **Access Control Matrix**
  - For every combination of (principal,object) say what mode of access is allowed
  - May be very large (1000s of principals, millions of objects)
  - May be sparse (most entries are “no access”)
- **Access Control Lists (ACLs)** = per object, list of allowed principals and access allowed to each
- **Capability Lists** = per principal, list of files allowed to access and type of access allowed
  - Could split it up into capabilities, each for a different (principal,file)

Security: Summary

- Security Challenges Abound
  - Lots of threats and attacks
- CIA Properties are desirable policies
- Encryption and decryption
- Shared key vs Public/private key systems
- Implementing authentication, signatures, certificates
- Authorization