CS 423
Operating System Design:
OS Security Overview

Professor Adam Bates
Spring 2018

Main Points

- Security theory
  - Access control matrix
  - Passwords
  - Encryption
- Security practice
  - Example successful attacks

Security: Theory

- Principals
  - Users, programs, sysadmins, …
- Authorization
  - Who is permitted to do what?
- Authentication
  - How do we know who the user is?
- Encryption
  - Privacy across an insecure network
  - Authentication across an insecure network
- Auditing
  - Record of who changed what, for post-hoc diagnostics

Authorization

- Access control matrix
  - For every protected resource, list of who is permitted to do what
  - Example: for each file/directory, a list of permissions
    - Owner, group, world: read, write, execute
    - Setuid: program run with permission of principal who installed it
  - Smartphone: list of permissions granted each app

Principle of Least Privilege

- Grant each principal the least permission possible for them to do their assigned work
  - Minimize code running inside kernel
  - Minimize code running as sysadmin
- Practical challenge: hard to know
  - what permissions are needed in advance
  - what permissions should be granted
    - Ex: to smartphone apps
    - Ex: to servers

Authorization w/ Intermediaries

- Trusted computing base: set of software trusted to enforce security policy
- Servers often need to be trusted
  - E.g.: storage server can store/retrieve data, regardless of which user asks
  - Implication: security flaw in server allows attacker to take control of system

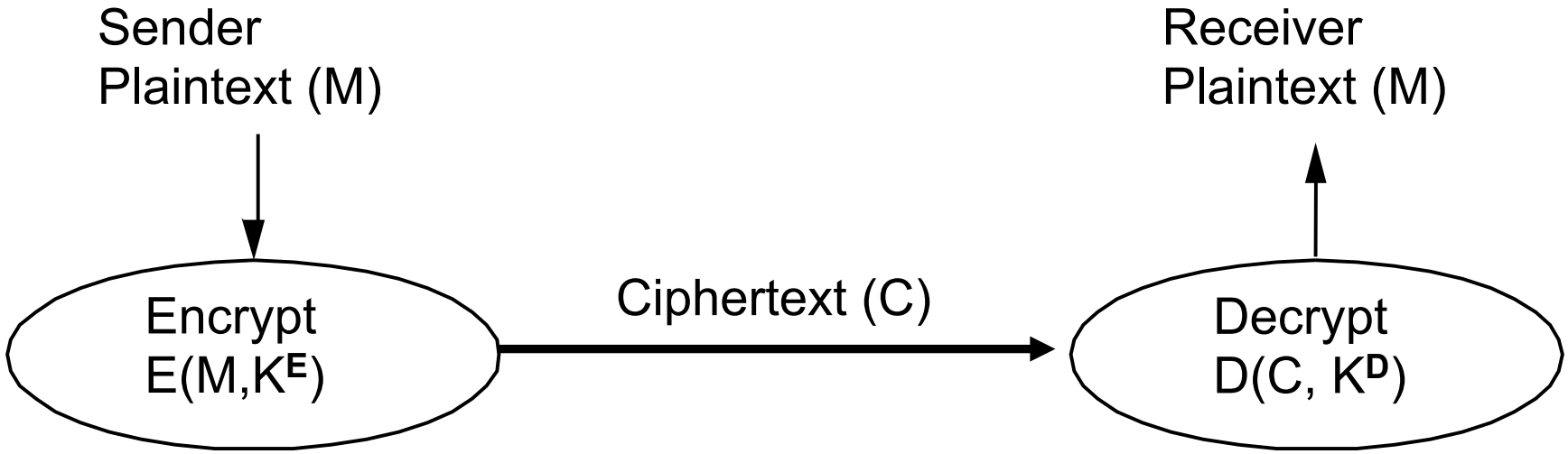
Authentication

- How do we know user is who they say they are?
- Try #1: user types password
  - User needs to remember password!
  - Short passwords: easy to remember, easy to guess
  - Long passwords: hard to remember

Question

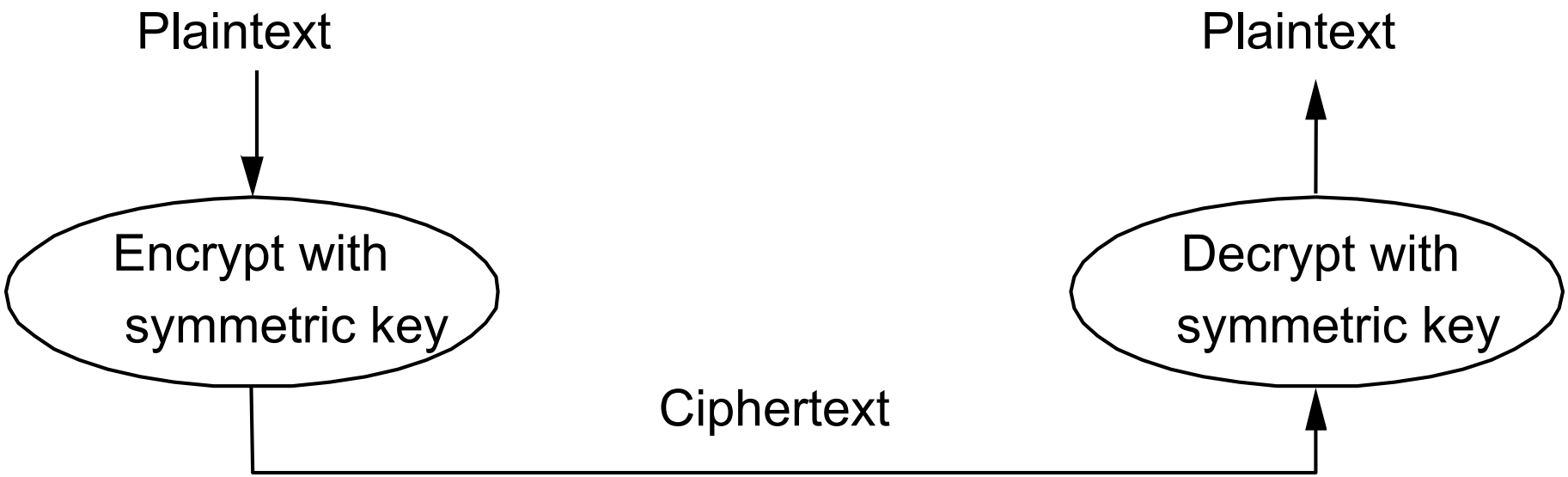
- Where are passwords stored?
  - Password is a per-user secret
  - In a file?
    - Anyone with sysadmin permission can read file
  - Encrypted in a file?
    - If gain access to file, can check passwords offline
    - If user reuses password, easy to check against other systems
- Encrypted in a file with a random salt?
    - Hash password and salt before encryption, foils precomputed password table lookup

Encryption



- Cryptographer chooses functions E, D and keys K^E, K^D
  - Suppose everything is known (E, D, M and C), should not be able to determine keys K^E, K^D and/or modify msg
  - provides basis for authentication, privacy and integrity

Symmetric Key (DES, IDEA)



- Single key (symmetric) is shared between parties, kept secret from everyone else
  - Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
  - if K kept secret, then both parties know M is authentic and secret

Public Key (RSA, PGP)

Keys come in pairs: public and private

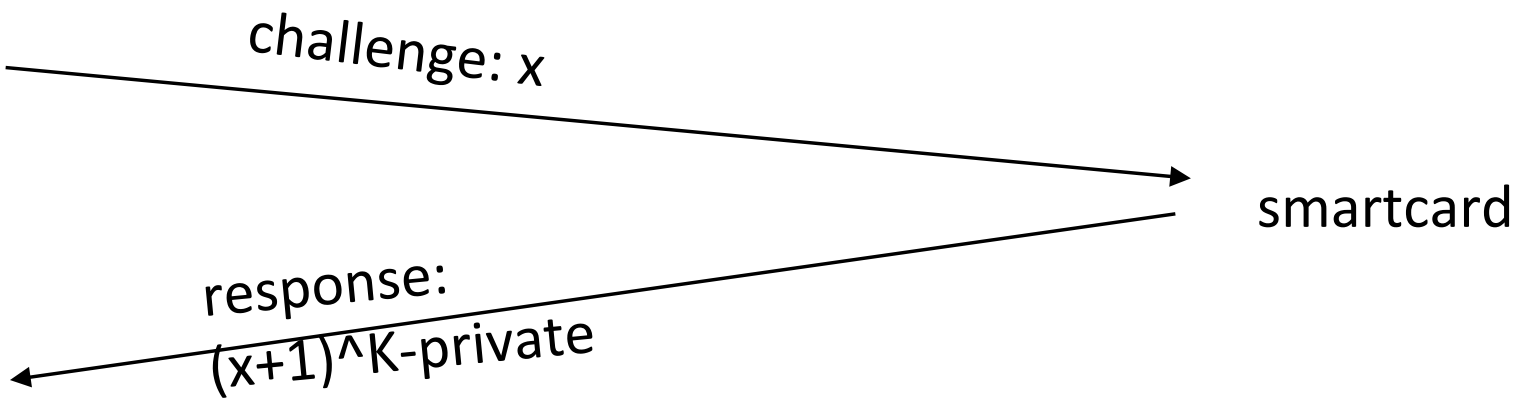
- M = ((M)^K-public)^K-private
- Ensures secrecy: can only be read by receiver

Encryption Summary

- Symmetric key encryption
  - Single key (symmetric) is shared between parties, kept secret from everyone else
  - Ciphertext = (M)^K
- Public Key encryption
  - Keys come in pairs, public and private
  - Secret: (M)^K-public
  - Authentic: (M)^K-private

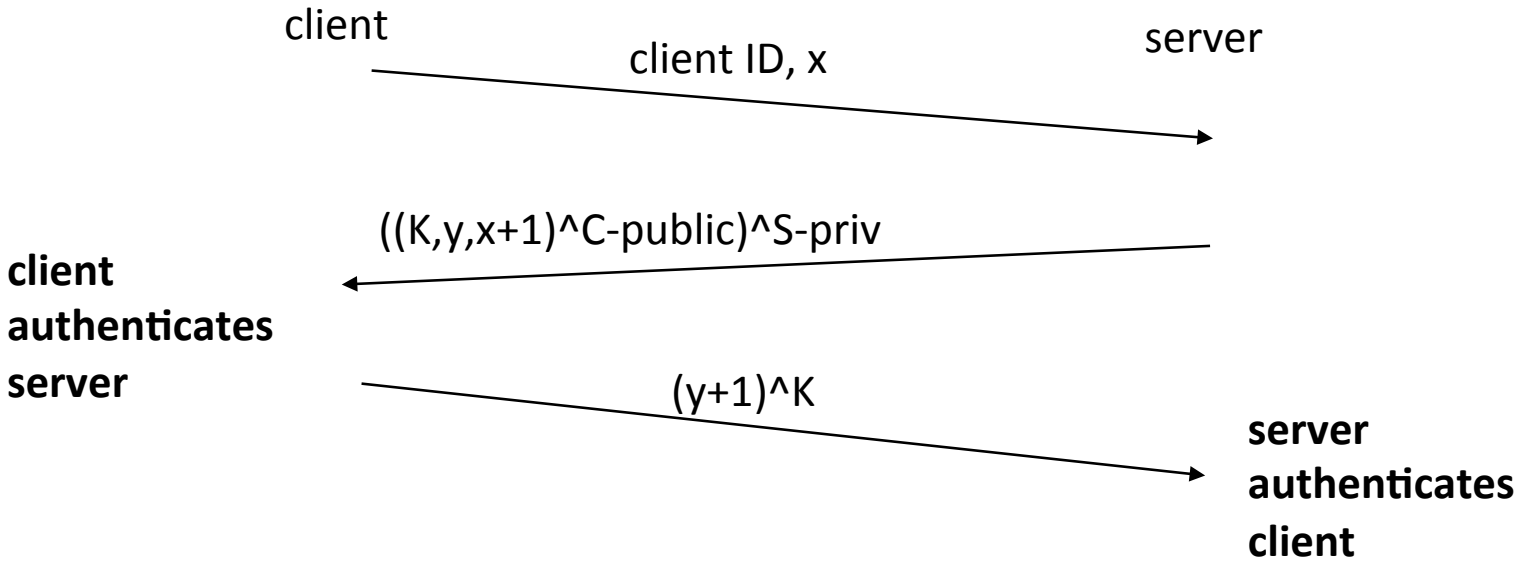
2-Factor Authentication

- Can be difficult for people to remember encryption keys and passwords
- Instead, store K-private inside a chip
  - use challenge-response to authenticate smartcard
  - Use PIN to prove user has smartcard



Public Key to Session Key

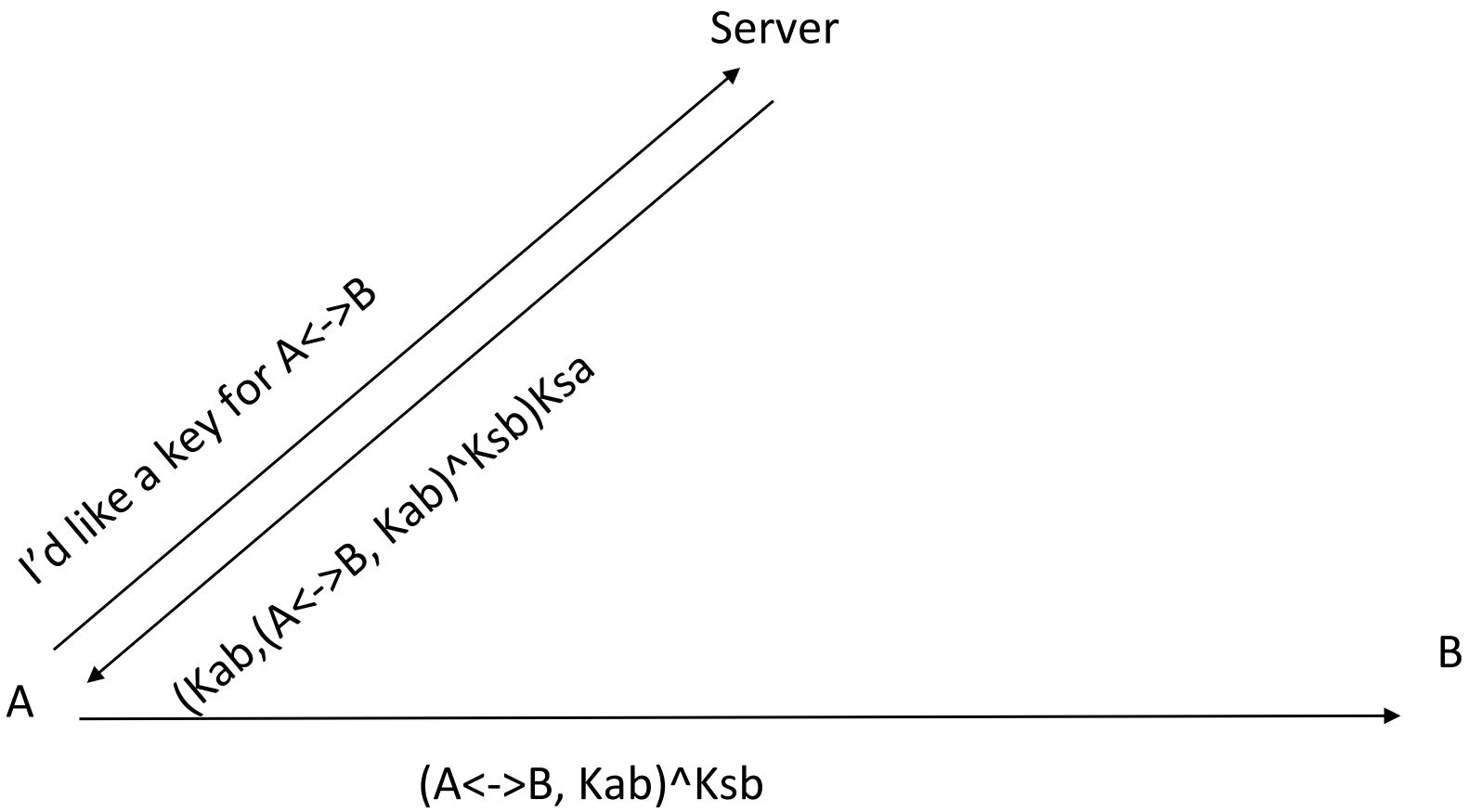
- Public key encryption/decryption is slow; so can use public key to establish (shared) session key
  - assume both sides know each other's public key



Symmetric Key to Session Key

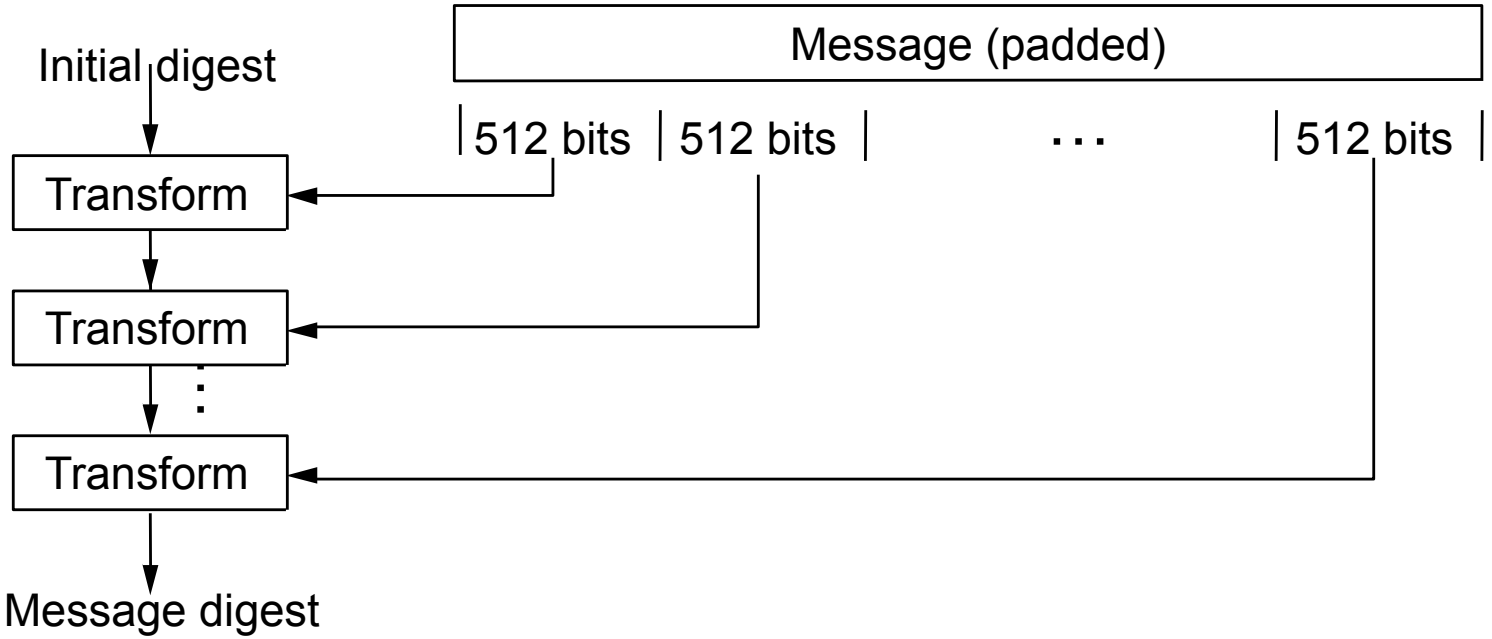
- In symmetric key systems, how do we gain a session key with other side?
  - infeasible for everyone to share a secret with everyone else
  - solution: "authentication server" (Kerberos)
    - everyone shares (a separate) secret with server
    - server provides shared session key for A <-> B
  - everyone trusts authentication server
    - if compromise server, can do anything!

Kerberos Example



Message Digest (MD5, SHA)

- Cryptographic checksum: message integrity
  - Typically small compared to message (MD5 128 bits)
  - "One-way": infeasible to find two messages with same digest



Security Practice

- In practice, systems are not that secure
  - hackers can go after weakest link
    - any system with bugs is vulnerable
  - vulnerability often not anticipated
    - usually not a brute force attack against encryption system
  - often can't tell if system is compromised
    - hackers can hide their tracks
  - can be hard to resecure systems after a breakin
    - hackers can leave unknown backdoors