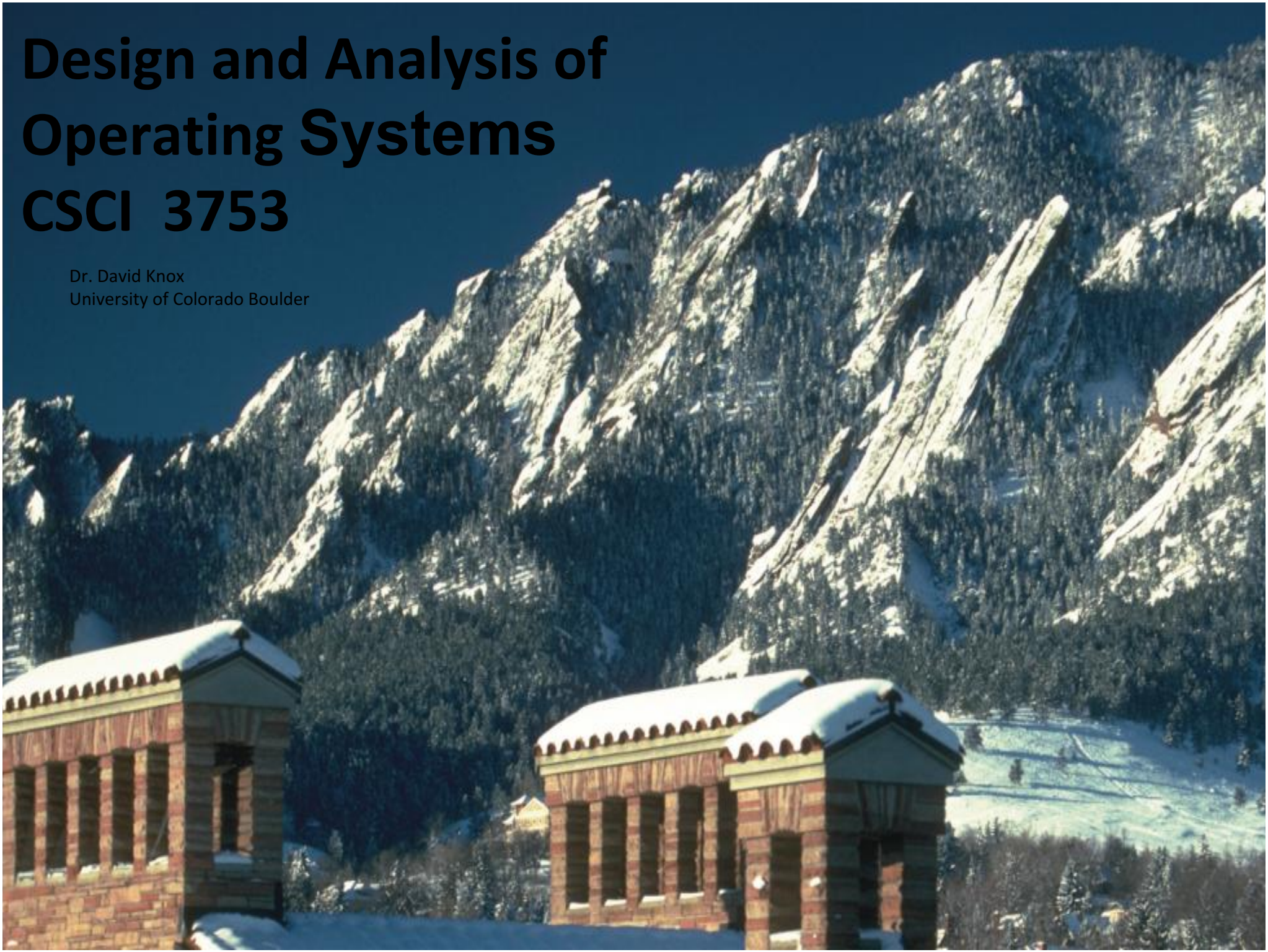
Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder



Design and Analysis of Operating Systems CSCI 3753

Security in Operating Systems

Dr. David Knox
University of
Colorado Boulder

Material adapted from: Operating Systems: A Modern Perspective : Copyright © 2004 Pearson Education, Inc.

Security in Operating Systems

Authentication

Data Integrity

6 Main Areas of Security

1. ***Authorization*** **– managing access to resources**
2. ***Confidentiality*** **– only allow authorized viewing of data - encrypting files and communication**
3. ***Authentication*** **– proving you are who you say you are**
4. ***Data Integrity*** **– detecting tampering with digital data**
5. ***Non-repudiation*** **– proving an event happened**
6. ***Availability*** **– ensuring a service is available (despite denial of service attacks)**

Authentication

- **Prove you are who you say you are**
  - e.g. Logging into your laptop or smartphone
- **Password is a form of authentication**
  - Providing the correct password is seen as authenticating the user to the OS
- **New: biometric authentication on smartphones**
- **For text-based authentication:**
  - Attacker can try to guess your password, using common words, etc.
  - OS can block or slow down access after too many login attempts

Remote Authentication

- **Harder than logging into your own laptop/mobile phone**
- **Now must communicate login messages over the network**
  - Early *rlogin* sent password unencrypted!
  - An attacker can employ a *replay attack*: just replay the password to login as the intercepted user
- **Should encrypt the password!**
  - *But even an encrypted password can be replayed!*

Remote Authentication

- **Solution: change/perturb the encrypted message containing the password on each login**
  - Approach 1: Add a different # to the message containing the password on each login attempt
    - e.g. a timestamp
    - Then encrypt the perturbed message
  - Approach 2: change the encryption key each login
- **For both approaches, the login message will look different on every login**
  - Hence, an attacker cannot replay an old encrypted message

SSH and SSL/TLS

- **SSH (Secure Shell) provides secure remote login**
  - is used to securely encrypt your password login and terminal traffic during a shell session (can be used to tunnel other data too)
- **SSL/TLS (Secure Sockets Layer/Transport Layer Security) is used to securely encrypt http Web traffic**
  - Any URL beginning with https:// is encrypted using SSL/TLS, e.g. for financial and product-ordering Web sites like Amazon

SSH

User/Client

Server

Server's Keys: K_{PUBLIC}, $K_{PRIVATE}$

Login?

K_{PUBLIC}

Ask user if they will accept K_{PUBLIC}. If so, choose symmetric session key K[1]

$E(K_{PUBLIC},K)$[2]

Decrypt session key K with $K_{PRIVATE}$:

$K = D(K_{PRIVATE},E(K_{PUBLIC},K))$

Encrypt password and messages

E(K,message)

Decrypt message with session key K

[1]In SSH-2, user and server negotiate a session key, e.g. using Diffie-Hellman, rather than the user just choosing a session key

[2]Note E(k,d) means some data d is encrypted by key k

SSH

- **Observation #1: SSH is a *hybrid* public key + symmetric key scheme. Why?**
  - Public key (RSA) cryptography is computationally expensive because of exponentiation at both encryptor and decryptor
  - In comparison, symmetric key cryptography is much faster
  - But symmetric key crypto suffers from the key distribution problem
  - So combine them: use public key to distribute the symmetric key!

SSH

- **Observation #1: SSH is a *hybrid* public key + symmetric key scheme. Why? (continued)**
  - Public key only encrypts the symmetric key, which is a small amount of data, so computational expense is minimized
  - Afterwards, all future data messages are encrypted using fast symmetric key cryptography
  - Get the best of both worlds:
    - solve the key distribution problem and
    - get fast encryption/decryption

SSH

- **Observation #2: The symmetric key changes on each login.**
  - This prevents a replay attack
- **Observation #3: The initial public key is not certified! SSH is subject to an MIM attack!**
  - When you SSH for the first time into a new server, it will prompt you to ask if you trust the advertised public key. This is called “trust on first use” (TOFU).
  - If you say yes, SSH caches the public key and proceeds. Hope no MIM on first access.

SSH

- There is a provision for certificates in SSH-2 (OpenSSH has a patch), but it is not widely used yet.
- Servers probably don't provide such certificates because of the effort to obtain certification from a trusted 3rd party

SSL/TLS

- **Secure Sockets Layer (SSL), renamed to Transport Layer Security (TLS), is the basis of the secure Web protocol *https***
- **Similar to SSH, except uses certified public keys:**
  1. Requestee's *certified* public key is initially passed to requestor, who verifies certificate
  2. SSL requestor negotiates a symmetric session key K with requestee
     - Negotiation is protected by encrypting it with certified public key
  3. All subsequent messages (e.g. secure Web page contents) are encrypted with symmetric key K

SSL/TLS

User/Client

Server

Server's Keys: Certificate(K_{PUBLIC}), $K_{PRIVATE}$

Login?

certificate(K_{PUBLIC})

Verify K_{PUBLIC} with certificate. If OK, generate random premaster secret.
K = f(premaster secret)

E(K_{PUBLIC},premaster secret)

Decrypt premaster secret with $K_{PRIVATE}$:
session key K = f(premaster secret)

Send messages

E(K,message)

Decrypt message with session key K

SSL/TLS

- **Key difference from SSH**
  - SSL/TLS uses certified public key

- **SSL/TLS's approach solves:**
  - confidentiality: login & data are encrypted
  - key distribution: public keys are used to distribute symmetric keys
  - speed: most data is encrypted with fast symmetric keys
  - replay attack: the encrypted login password looks different every time because the symmetric key changes every session
  - man-in-the-middle attack: because the initial public key is certified (improvement over SSH)

Other Authentication Approaches

- **Challenge-response protocol:**
  - X and Y share a secret symmetric key. X wants to authenticate a node N that says it is Y.
  - X sends a challenge to N, i.e. a random number used only once (nonce)
  - N sends to X nonce encrypted w/ N's symmetric key
  - X decrypts N's message with X's sym key. If decrypted # matches nonce, X knows responder N is Y.
- **S/KEY has 1-time password vs replay attacks**
  - a list of one-time passwords is generated a priori and then consulted during login at both ends
  - list could be generated using a one-way function

6 Main Areas of Security

1. ***Authorization*** **– managing access to resources**
2. ***Confidentiality*** **– only allow authorized viewing of data - encrypting files and communication**
3. ***Authentication*** **– proving you are who you say you are**
4. ***Data Integrity*** **– detecting tampering with digital data**
5. ***Non-repudiation*** **– proving an event happened**
6. ***Availability*** **– ensuring a service is available (despite denial of service attacks)**

Data Integrity

- **Refers to the overall completeness, accuracy and consistency of the data**

- **Physical integrity**
  - challenges of correctly storing and retrieving the data
  - hardware faults

- **Logical integrity**
  - software bugs (corrupting data)
  - human errors

Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of
Colorado Boulder

Material adapted from: Operating Systems: A Modern Perspective : Copyright © 2004 Pearson Education, Inc.