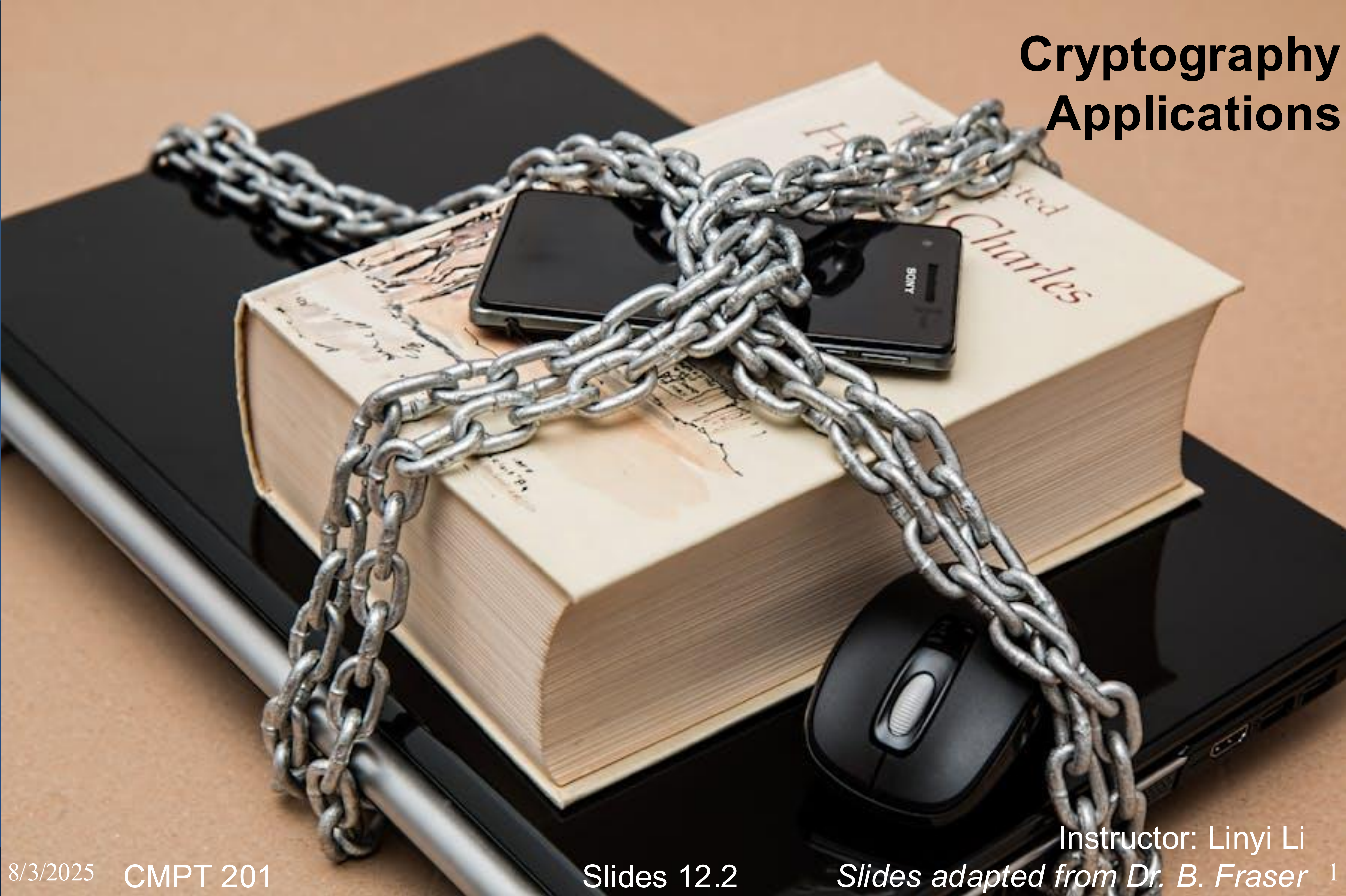


Cryptography Applications

Instructor: Linyi Li

Slides adapted from Dr. B. Fraser

Topics

- How can we safely store passwords?
- How do we verify a document is authentic?
- How can we trust websites?

Storing Passwords

Storing Passwords

- Password verification systems don't store plain text passwords
  - .. Passwords are hashed and it stores only the hashes.
  - Adds security: attacker getting a copy of the password file gives them the hash which cannot be used to log in.
  - To check a password, system checks
    .. hash(user-input) == stored hash.
- Linux stores passwords in /etc/shadow (accessible by root)

```
pwdis_1234:$y$j9T$LirzYIOxlHbbu6Wi/9zCl.$I9AZk5jAaY0uRPQfPnTEu.x5UeMVROrhP.i9gI96DD7:20159:0:99999:7:::
pwdis_1235:$y$j9T$Rnadt7C63/7BI1s/3Gx8v0$fvKI5ljHrj2hhnIhb6SkvjPvDQb7Awb3wUKU5YNmKK.:20159:0:99999:7:::
pwdis_abcd:$y$j9T$/TAalyA61/lpbWM0vB5wA1$jofzhQHkwXjtpqfZ6cXIEwepD1L1V75gNriCM5C3pS7:20159:0:99999:7:::
```

User name

Hash alg. y=yescript

Salt

Hash

Password expiry info

Rainbow Table Attack

- Rainbow table attack
  - .. Attacker pre-computes the hash of common passwords.
  - Can then quickly search password file of hashes for known passwords.
- Defence: Salt the password.
  - Salt: .. A random number or string
  - Store salt and hash(user-password || salt) (|| means concatenation)
  - Verify password:
  - ·· hash(user-input || salt) == stored-hash
- Attacker cannot reasonably compute hash of all possible passwords along with all possible salts.

Verifying Documents

Secure digest

- Secure digest for summary of document
  - Often used to verify a downloaded file is not corrupted.
  - A secure digest is a summary of a message:
    - A fixed-length string that characterizes an arbitrary-length message.
  - Typically produced by a cryptographic hash function e.g., SHA-256.
- Example

```
$ sha256sum ./README.md
e293cdc4f5c4686772fba8159be9e9636654fed7ce72bfd2e75add8a6833c5ab  ./README.md
```

Digital Signature

- Digital signatures combine public key crypto and hashing.
  - Goal: .. Verify a message (or document) is an unaltered copy of the one produced by the signer.
  - The message can be public; we just want to prove who sent it and that it's unaltered.
- Two parties: signer and verifier.
  - The Signer:
    - Sends a message
    - Wants to prove they sent the message.
  - The Verifier:
    - Receives message
    - Wants to verify the message was sent by the signer and is unaltered.

Signer

- The Signer will:
  - Writes a document: m
  - Computes a message digest: h(m) (e.g., using SHA-256)
    - Not good enough yet: Adversary could write document z, computes h(z) and plant both on the server.
  - .. Encrypts the digest with own private key: enc(h(m)) (e.g., using RSA public key crypto)
    - This is called signing.
    - Only the signer has the private key, so only the signer can encode with it.
  - Sends the message & the signature: <m, enc(h(m))>

Verifier

- The Verifier will:
  - Receives the message and the signature:
    <m, enc(h(m))>
  - Decrypts the signature with.. the signer's public key:
    dec(enc(h(m))) == h(m)
  - Computes a message digest: h(m). Let's call it h'.
  - .. Verifies that h(m) == h'
    - If yes, then the message is authentic.
- Since only signer knows their private key,
  ..if the signature can be decrypted with their public key
  then they must have signed the document.

Trusting Unknown Companies

Digital Certificate

- Digital certificates use digital signatures.
- Scenario
  - Imagine sending password to website (e.g., Instagram).
  - You encrypt your password with Instagram's public key.
  - Only Instagram can decrypt the message, so password is safe.
- Questions
  - .. How do you get the public key of Instagram?
  - One way:

  Instagram sends you their public key when you first go site.
  - How do you know if the public key really belongs to Instagram?

  But a rogue website could disguise as Instagram and send you a wrong key.

Secure Browsing

- HTTP has no encryption.
- HTTPS uses encryption:
  - Instagram sends you its public key in a digital certificate.
  - Digital Certificate:
    - proves that the public key indeed belongs to Instagram.
  - Your OS verifies the authenticity of the digital certificate.

  OS has some built-in.. trusted signing authorities.
  - Your browser then uses Instagram's public key to encrypt messages to Instagram.
- Only Instagram can decrypt messages encrypted with their public key.

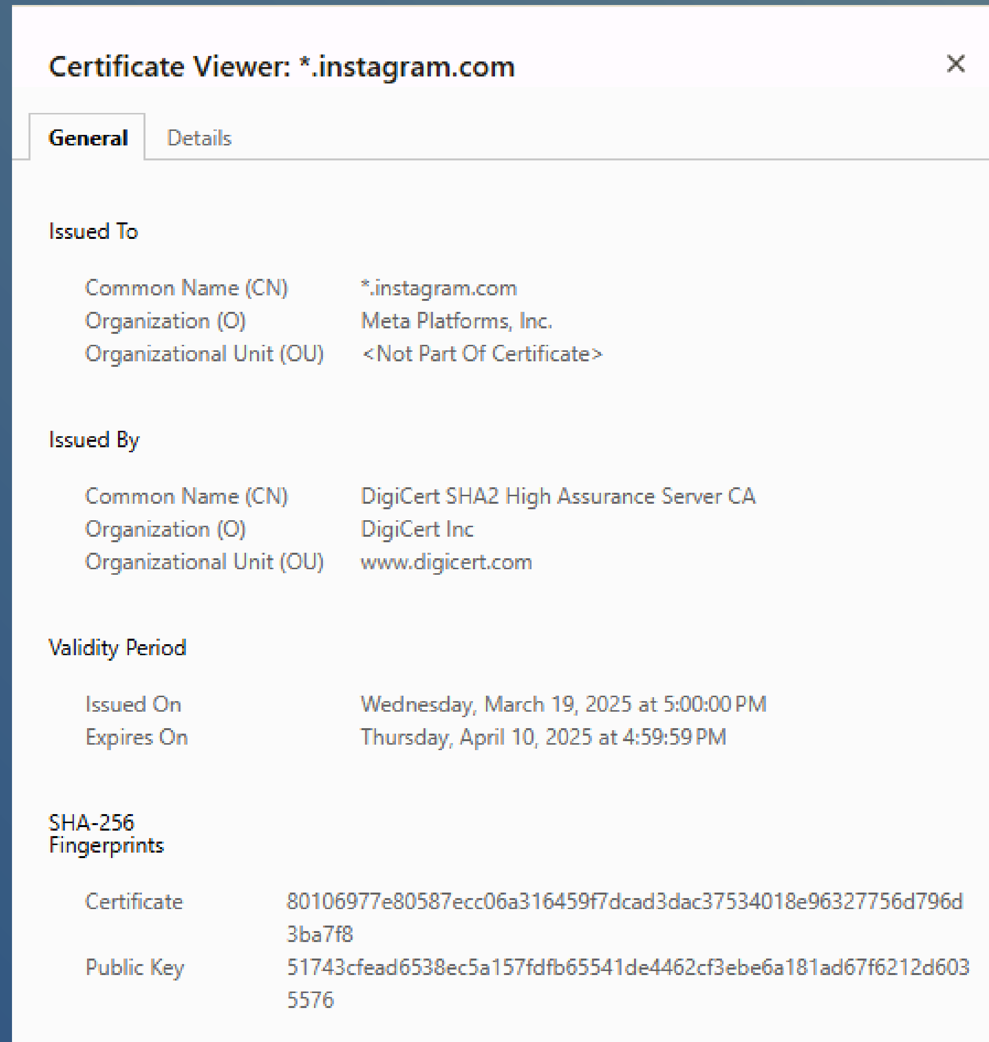
Digital Cert Operation

- How digital certificates work:
  - A digital certificate is signed by a digital certificate authority
    - E.g., VeriSign, DigiCert
  - OS vendor ships OS with public keys for some trusted digital certificate authorities like DigiCert.
    - This establishes the base level of trust:

    .. root of trust.

Digital Certificate Example

- Instagram uses DigiCert:
  - Instagram goes to DigiCert, gives its public key, and requests a digital certificate.
- DigiCert creates a digital certificate:
  - It says "this public key belongs to Instagram"
  - ..DigiCert signs it with DigiCert's own private key.



Digital Certificate Example (cont)

- Instagram Digital Certificate
  - When browser connects to Instagram, Instagram sends the digital certificate.
    - Browser uses its trusted public key for DigiCert to verify the digital certificate from Instagram.
  - If your OS is not compromised, this whole process is secure based on the first level of trust.
    - If the OS is compromised, there is no 1st level of trust and this whole process is not secure.
- Encryption Use
  - .. Public key encryption is used to exchange random secret key (public key is slow and generates lots of data)
  - Rest of communication.. encrypted with secret key encryption (faster, smaller)

Chain of Trust

- Digital certificates rely on the chain of trust
  - To trust the public key sent by Instagram,
    we need to trust DigiCert's signature.
  - To trust DigiCert's signature,
    we need to trust DigiCert's public key.
  - In order to trust DigiCert's public key (shipped with OS),
    we need to trust that our OS is not compromised.
- Chain of trust relies on the root of trust being trustworthy.
  - Our root of trust is the OS.

Activity: SSH

- [Opt] Spin up new container: docker run -it ghcr.io/sfu-cmpt-201/base
- Generate public/private key with ed25519 & passphrase

  $ ssh-keygen -t ed25519 -C "your email address"

  – Look at files in ~./ssh
- SSH SFU

  – ssh <yourID>@csil-cpu01.csil.sfu.ca –p 24

  Asks user name & password; use VPN if off campus.
- SSH Keys

  – SSH SFU; manually add pub key to end of ~/.ssh/authorized_keys

  – Log-out, log-in (asks passphrase)
- SSH Agent: Avoids passphrase; Stores key in memory.

  eval ssh-agent

  ssh-add

  kill $SSH_AGENT_PID

Hash Collisions

Birthday Match = Hash Collision

- Birthday Match
  - In a class of 30 people, probability of two students having the same birthday is.. ~70%!
https://en.wikipedia.org/wiki/Birthday_attack

- Hash Collisions
  - Given enough messages,
..we can find a hash collision between two messages.
  - i.e., can we show that a hash function
.. does not achieve strong collision resistance?

- (Recall) Strong collision resistance:
It should be difficult to find two messages x and x'
where h(x) == h(x').
  - i.e., given a hash function, it should be difficult to find two values that produce the same hash.

Birthday Attack

- Attacker use a contract the customer is expected to sign (say agreement to buy company for $100,000).
- Attacker then:

– .. Creates benign altered copies of the contract (adding a space, adding commas, adding typos, ...)

– Creates malicious altered copies (sale price $100,000,000)

– Goal: .. finds malicious contract with same hash as benign contract

– Customer given benign copy to sign using their private key and hash of document.

– Attacker then.. substitutes benign contract with malicious one.

– Since the contracts have same hash, attacker can claim customer signed malicious contract using their private key!

Demo: Hash Collision

- Demo: Collision in Crypto Hash Functions
  - MD5 was a widely used crypto hash function but was found to be insecure by 2005.
  - No longer in use.
- Get images & Compare Hashes

```
$ wget https://s3-eu-west-1.amazonaws.com/md5collisions/ship.jpg
$ wget https://s3-eu-west-1.amazonaws.com/md5collisions/plane.jpg
$ openssl dgst -md5 ship.jpg
$ openssl dgst -md5 plane.jpg
```

  - Algorithm exists to manipulate a pair of images into having the same MD5 hash.
- SHA256 is not yet known to be insecure.

```
$ openssl dgst -sha256 ship.jpg
$ openssl dgst -sha256 plane.jpg
```

ABCD: Birthday

- A birthday attack is successful when attackers find:

(a) Two images that look the same but have different binary data.
(b) Two students in CMPT 201 who have the same birthday.
(c) A second document which matches the hash of a single given document.
(d) Hash collision of a benign and malicious document.

Summary

- Passwords
  - Store salted and hashed passwords to avoid rainbow tables.
- Digest
  - A hash of a document.
- Digital Signatures
  - Sign a hash with a private key.
- Digital Certificates
  - Sign document to show who really owns a public/private key.
  - Chain of trust for distributing certificates.
  - Root of trust built into OS.
- Hash Collisions
  - Duplicate hash (digital signature) is a security issue.
  - Birthday attack to find duplicates.