SE-4920: Lecture 6 Introduction to cryptography

- Reading
  - Chapter 2
- Today's Outcomes
  - Define common cryptography terms
  - Discuss the effect of processing power on the effectiveness of cryptography
  - Explain the meaning of and relationship between the 3 basic classes of cryptographic attacks: ciphertext only, known plaintext, chosen plaintext
  - Discuss the similarities and differences among the 3 basic types of cryptographic functions: (0-, 1-, and 2-key): hash, secret key, public key

1

What is cryptography?

- Cryptography = secret writing (Greek)
  - Temporarily (over network, or storage) mangle a message
- Main use = Confidentiality
- Can also provide
  - Integrity checking
  - Authentication

2

Terms and overview

- Plaintext – original message
- Ciphertext – mangled information
- Encryption – producing ciphertext from plaintext
- Decryption – the reverse of encryption
- Key – (usually) secret value that allows the same algorithm to be used securely by millions of people (*e.g.*, combination lock)

plaintext $\xrightarrow{\text{encryption}}$ ciphertext $\xrightarrow{\text{decryption}}$ plaintext

3

Computational difficulty

- Encryption/decryption should be fast for people who know the key
- And unbearably slow for people who don't
- Combination lock
  - Adding a digit
    - Increases unlocking time linearly if you know the code: $\Theta(N)$
    - But exponentially if you don't: $\Theta(2^N)$
- Good news
  - Faster computers make larger N feasible
    - Rapidly increasing the divide between user and attacker

4

Simple secret codes

- Caesar cipher
  - Shift letters in alphabet by secret amount between 0 and 25 (cyclic order, 'a' follows 'z'):
    - E("Code", 2) = "Eqfg"
    - 26 possibilities (keys)
    - Easily broken with enough text by trying all keys (recognizable plaintext attack)
- Monoalphabetic cipher
  - "Mono" – each letter is always replaced with the same ("mono" = "one") letter
  - 26! ≈ 4e26 (a lot of possible keys)
  - Still easy to cryptanalyze
    - *E.g.*, character frequency analysis ('e' vs. 'q', etc.)

5

Basic attacks against cryptographic schemes

- Generally want to be secure against all
  - But security against the more basic attacks may suffice in some cases
- Ciphertext only (most basic)
- Known plaintext
- Chosen plaintext (most sophisticated)

6

Ciphertext only

- Encrypted data captured from network, storage, etc.
  - May accumulate and store over time (need "enough" data)
- Offline attack – try all keys?
  - Stop when the result makes sense?
    - "Recognizable plaintext attack"
  - Limited keyspace problems
    - What if key = f(password)?
    - Password is one of ~10000 ≈ 2^{13} dictionary words
    - Or one of ~60^8 ≈ 2^{47} random strings?

7

Known plaintext

- Capture some <plaintext, ciphertext> pairs
  - Maybe an encrypted message later became public
- Defeats monoalphabetic and Caesar ciphers
  - And some other methods
- Do not use algorithms vulnerable to known plaintext
  - If plaintext might ever become known

8

Chosen plaintext

- When you can get the system to encrypt something for you
- Again, Caesar and monoalphabetic are vulnerable to this
- Also, vulnerable to message guessing
  - Guess possible messages m_1 m_2 m_3
  - Get $E(m_1)$, $E(m_2)$, $E(m_3)$
  - Look for match with intercepted value
- There are solutions to this
  - *E.g.*, make E() a function of something that changes with each message (time, nonce)

9

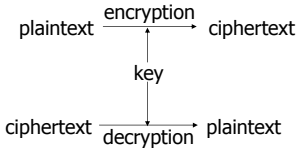
The 3 types of cryptographic functions

- Secret key cryptography
  - ***One***, shared key
- Public key cryptography
  - Keys come in ***pairs***, a private half and a public half
- Hash function
  - ***Zero*** keys (algorithm only)

10

Secret key cryptography

- Encryption and decryption based on a pre-shared secret
- Also called "symmetric" (since the same key is used on both sides) or "conventional" cryptography



11

Using secret key cryptography

- Transmit over an insecure channel (confidentiality)
- Secure storage on insecure media (just another type of channel)
  - Danger of losing the key
- Authentication
  - Prove you can encrypt a random number with the shared secret (and thus you know the secret)
  - Weakness of basic exchange?
- Extending the basic checksum / CRC, which are easily replaced by an attacker
  - Cryptographic checksum / message authentication code (MAC)
  - f(message, secret), typically 48 bits or longer

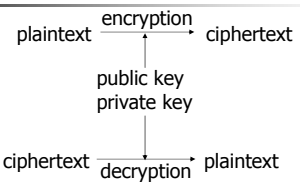
12

Public key cryptography

- Also called asymmetric cryptography, since a different key is used on each side
- Newest form, first published in 1976
- Keys come in mathematically related pairs
  - Public key – e – normally for encrypting
  - Private key – d – normally for decrypting
- Many advantages over private key, including
  - No shared secrets – keep your *d* private, but tell the world what your *e* is

13

Operation of public key cryptography



- Much slower than secret key, solution...
  - Often generate a random secret key
  - Protect it with public/private key
  - Protect data with the random secret

14

Uses of public key cryptography

- Transmit over an insecure channel
  - No pre-shared secret, but need to trust that you have the right public key (Chapter 15)
- Secure storage on insecure media
  - Advantage over secret key: encrypt for someone else (and only them)
- Authentication
  - *E.g.*, prove I know my private d
    - You send me c = $\{r\}_e$
    - Only I can perform D(c, d) = r and tell you your random r
- Digital signatures: $[m]_d = f(m, d)$
  - Proves
    - Who generated message (only ***one*** owner – no ***shared*** secret)
    - That the message was not modified
  - Together: non-repudiation (prove who placed the order, *cf.* secret)

15

Hash algorithms

- h: potentially long message → fixed length (short) number (*e.g.*, 64 or 128 bits)
- Cryptographic hash properties
  - Easy to compute h(m) for all m
  - No better way to find an m that generates a given h(m) than to do a random search
  - Infeasible to find m_1 and m_2 such that $h(m_1) = h(m_2)$
    - Although, many such pairs must exist
      - The messages can be arbitrarily long

16

Uses of hash algorithms

- Password hashing
  - Store the hash for comparison, not the password
- Message integrity
  - Problem: How do you know that an attacker didn't just regenerate the hash when he modified the message?
  - Solution: keyed hash: Transmit h(m|s)|m
- Message fingerprint / message digest
  - Detect changes to file by securely storing (*e.g.*, file cabinet, ROM) and occasionally verifying hash
- Efficient digital signatures
  - Use public key to sign a hash of a message

17