Security Threats

- Leakage: An unauthorized party gains access to a service or data.
  - Attacker obtains knowledge of a withdrawal or account balance
- Tampering: Unauthorized change of data, tampering with a service
  - Attacker changes the variable holding your personal checking $5 total
- Vandalism: Interference with proper operation, without gain to the attacker
  - Attacker does not allow any transactions to your account

Security Properties

- Confidentiality: Concealment of information or resources
- Authenticity: Identification and assurance of origin of info
- Integrity: Trustworthiness of data or resources in terms of preventing improper and unauthorized changes
- Availability: Ability to use desired info or resource
- Non-repudiation: Offer of evidence that a party indeed is sender or a receiver of certain information
- Access control: Facilities to determine and enforce who is allowed access to what resources (host, software, network, …)

Attack on Confidentiality

- Eavesdropping
  - Unauthorized access to information
  - Packet sniffers and wiretappers (e.g. tcpdump)
  - Illicit copying of files and programs

Attack on Integrity

- Tampering
  - Stop the flow of the message
  - Delay and optionally modify the message
  - Release the message again

Attack on Authenticity

- Fabrication
  - Unauthorized assumption of other’s identity
  - Generate and distribute objects under identity
### Attack on Availability
- Destroy hardware (cutting fiber) or software
- Modify software in a subtle way
- Corrupt packets in transit
- Blatant denial of service (DoS):
  - Crashing the server
  - Overwhelm the server (use up its resource)

### Designing Secure Systems
- Your system is only as secure as your weakest component!
- Need to make worst-case assumptions about attackers:
  - exposed interfaces, insecure networks, algorithms and program code available to attackers, attackers may be computationally very powerful
  - Tradeoff between security and performance impact/difficulty
  - Typically design system to withstand a known set of attacks (Attack Model or Attacker Model)
- It is not easy to design a secure system.
- And it’s an arms race!

### CSE 486/586 Administrivia
- CSE 622 Advanced Computer Systems
  - Probably on Android platform
  - Will be open for registration either today or tomorrow
- More practice problems & example final posted
- PhoneLab hiring
  - Testbed developer/administrator
- Anonymous feedback form still available.
- Please come talk to me!

### Cryptography
- Comes from Greek word meaning “secret”
  - Primitives also can provide integrity, authentication
- Cryptographers invent secret codes to attempt to hide messages from unauthorized observers
  - encryption plaintext ciphertext decryption plaintext
- Modern encryption:
  - Algorithm public, key secret and provides security
  - May be symmetric (secret) or asymmetric (public)
- Cryptographic algorithms goal
  - Given key, relatively easy to compute
  - Without key, hard to compute (invert)
  - “Level” of security often based on “length” of key

### Three Types of Functions
- Cryptographic hash Functions
  - Zero keys
- Secret-key functions
  - One key
- Public-key functions
  - Two keys

### Cryptographic Hash Functions
- Take message, \( m \), of arbitrary length and produces a smaller (short) number, \( h(m) \)
- Properties
  - Easy to compute \( h(m) \)
  - Pre-image resistance: Hard to find an \( n \), given \( h(m) \)
    - “One-way function”
  - Second pre-image resistance: Hard to find two values that hash to the same \( h(m) \)
    - E.g. discover collision: \( h(m) = h(m') \) for \( m \neq m' \)
  - Often assumed: output of hash fn’s “looks” random
How Hard to Find Collisions?

- **Birthday paradox**
  - In a set of \( n \) random people, what's the probability of two people having the same birthday?
  - **Calculation**
    - Compute probability of different birthdays
    - Random sample of \( n \) people taken from \( k = 365 \) days

Birthday Paradox

- Probability of no repetition:
  - \( P = 1 - \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\ldots\left(1 - \frac{n-1}{365}\right) \)
  - \( (k = \# \) of slots, e.g., 365 \( ) \) \( P \approx 1 - e^{-\frac{n(n-1)}{2k}} \)
  - For \( p \), it takes roughly \( \sqrt{2k \ln(1/(1-p))} \) people to find two people with the same birthday.
  - With \( p = 50\% \),

How Many Bits for Hash?

- If \( m \) bits, how many numbers do we need to find (weak) collision?
  - It's not \( 2^m \)
  - It takes \( 2^m \) to find weak collision
  - Still takes \( 2^m \) to find strong (pre-image) collision
  - 64 bits, takes \( 2^{64} \) messages to search
  - MD5 (128 bits) considered too little
  - SHA-1 (160 bits) getting old

Example: Password

- Password hashing
  - Can't store passwords in a file that could be read
  - Concerned with insider attacks!
  - Must compare typed passwords to stored passwords
    - Does hash (typed) \( = \) hash (password)?
  - Actually, a salt is often used: hash (input \( || \) salt)
  - Avoids precomputation of all possible hashes in "rainbow tables" (available for download from file-sharing systems)

Symmetric (Secret) Key Crypto

- Also: "conventional / private-key / single-key"
  - Sender and recipient share a common key
    - All classical encryption algorithms are private-key
    - Dual use: confidentiality (encryption) or authentication/integrity (message authentication code)
  - Was only type of encryption prior to invention of public-key in 1970's
    - Most widely used
    - More computationally efficient than "public key"

Symmetric Cipher Model
Requirements

• Two requirements
  – Strong encryption algorithm
  – Secret key known only to sender/receiver
• Goal: Given key, generate 1-to-1 mapping to ciphertext that looks random if key unknown
  – Assume algorithm is known (no security by obscurity)
  – Implies secure channel to distribute key

Uses

• Encryption
  – For confidentiality
    – Sender: Compute $C = AES_K(M)$ & Send $C$
    – Receiver: Recover $M = AES_K^{-1}(C)$
• Message Authentication Code (MAC)
  – For integrity
    – Sender: Compute $H = AES_K(SHA1(M))$ & Send $<M, H>$
    – Receiver: Compute $H' = AES_K(SHA1(M))$ & Check $H' == H$

Public (Asymmetric) Key Crypto

• Developed to address two key issues
  – Key distribution: secure communication without having to trust a key distribution center with your key
  – Digital signature: verifying that a message comes from the claimed sender without prior establishment
• Public invention Diffie & Hellman in 1976
  – Known earlier to classified community

Public (Asymmetric) Key Crypto

• Involves two keys
  – Public key: can be known to anybody, used to encrypt and verify signatures
  – Private key: should be known only to the recipient, used to decrypt and sign signatures
• Asymmetric
  – Can encrypt messages or verify signatures w/o ability to decrypt msgs or create signatures
  – If “one-way function” goes $c \leftarrow F(m)$, then public-key encryption is a “trap-door” function:
    • Easy to compute $c \leftarrow F(m)$
    • Hard to compute $m \leftarrow F^{-1}(c)$ without knowing $k$
    • Easy to compute $m \leftarrow F^{-1}(c,k)$ by knowing $k$

Security of Public Key Schemes

• Like private key schemes, brute force search possible
  – But keys used are too large (e.g., >= 1024 bits)
• Security relies on a difference in computational difficulty b/w easy and hard problems
  – RSA: exponentiation in composite group vs. factoring
  – ElGamal/DH: exponentiation vs. discrete logarithm in prime group
  – Hard problems are known, but computationally expensive
• Requires use of very large numbers
  – Hence is slow compared to private key schemes
    – RSA-1024: 80 us / encryption; 1460 us / decryption [cryptopp.com]
    – AES-128: 109 MB / sec = 1.2us / 1024 bits
(Simple) RSA Algorithm

- **Security** due to cost of factoring large numbers
  - Factorization takes $O(e^{\frac{1}{3} (log n)^{2} log log n})$ operations (hard)
  - Exponentiation takes $O((log n)^{3})$ operations (easy)
- To encrypt a message $M$ the sender:
  - Obtain public key $(e, n)$; compute $C = M^e \mod n$
- To decrypt the ciphertext $C$ the owner:
  - Use private key $(d, n)$; computes $M = C^d \mod n$
- Note that msg $M$ must be smaller than the modulus $n$
- Otherwise, hybrid encryption:
  - Generate random symmetric key $r$
  - Use public key encryption to encrypt $r$
  - Use symmetric key encryption under $r$ to encrypt $M$

Typical Applications

- **Secure digest**
  - A fixed-length that characterizes an arbitrary-length message
  - Typically produced by cryptographic hash functions, e.g., SHA-1 or MD5.
- **Digital signature**
  - Verifies a message or a document is an unaltered copy of one produced by the signer
  - Signer: compute $H = RSA_{d}(SHA1(M))$ and send $<M, H>$
  - Verifier: compute $H' = SHA1(M)$ and verify $RSA_{d}(H) == H'$
- **MAC (Message Authentication Code)**
  - Digital signatures with secret keys
  - Verifies the authenticity of a message
  - Sender: compute $H = AES_{r}(SHA1 (M))$ and send $<M, H>$$$
  - Receiver: compute $H' = AES_{r}(SHA1 (M))$ and check $H' == H$

Summary

- **Security properties**
  - Confidentiality, authenticity, integrity, availability, non-repudiation, access control
- **Three types of functions**
  - Cryptographic hash, symmetric key crypto, asymmetric key crypto
- **Applications**
  - Secure digest, digital signature, MAC, digital certificate

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Jennifer Rexford (Princeton) and Michael Freedman (Princeton).