Basics of Data Security, Privacy, and Encryption

Kai Shen

Data Security in the Big Data Era

- Mis-uses of big data is a big concern, for examples
  - information of a person's online activities may reveal every aspect of the person's life
  - medical/genetic information of individuals can be improperly exploited by health insurers
  - large datasets may be stolen by governments or unfriendly groups
- Not a comprehensive study on data security, but
  - to understand the value and consequences of misuses of big data
  - to understand basic techniques and practical issues in data security
  - to know, as a user, how to protect yourself in the big data era
  - to stimulate your interests to read more and possibly do more (final course project)

Data Security Model

- What can adversary do to your data?
  - Steal and understand the content of sensitive data
  - Actively changing data
- Data security protection:
  - Confidentiality: only the proper persons/groups can possess and "understand" data
  - Data Integrity: data is preserved in an integral way without unauthorized tampering

Cryptography

- Encryption: uses a key to turn original data into encrypted data
- Decryption: uses a key to turn encrypted data back to the original
  - It is (computationally) infeasible to know the original data from the encrypted data without the decryption key
- Symmetric key crypto: encryption and decryption keys are identical. (both are secret)
- Public key crypto: encryption key is public, decryption key is secret.

Symmetric Key Cryptography: Monoalphabetic Cipher

Monoalphabetic cipher: substitute one letter for another.

```
plaintext:  abcdefghijklmnopqrstuvwxyz
            |                       |
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

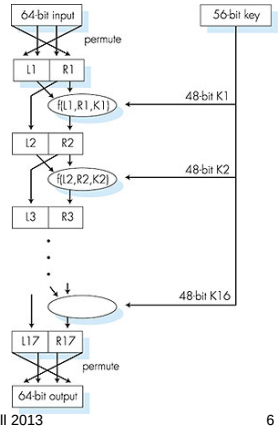
Example: `Plaintext: bob. i love you. alice`
`ciphertext: nkn. s gktc wky. mgsbc`

Q1: How hard to break this simple cipher?
- brute force?
- other?

Q2: How to make it more difficult to break?

Symmetric Key Cryptography: DES

- DES: Data Encryption Standard
  - US encryption standard [NIST 1993]
  - 56-bit symmetric key, 64-bit plaintext input
  - encryption: initial permutation ⇒ 16 "rounds", each using different 48 bits of key ⇒ final permutation
  - decryption: reverse operation using the same key
- How secure is DES?
  - DES Challenge (1999): 56-bit-key-encrypted phrase decrypted (brute force) in 22 hours 15 minutes
- Making DES more secure:
  - use three keys sequentially (3-DES)
  - use more bits



AES: Advanced Encryption Standard

- Newer (Nov. 2001) symmetric-key NIST standard, replacing DES
- Processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- Brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for 128-bit AES

Stream Cypher

- Fixed-size key K expanded to an infinite random stream C(K)
- For data X, it uses an portion of the key stream (equal length to data X), then xor the data to produce encrypted data
- Same key cannot be used twice, otherwise

  Encrypted(X) = X xor C(K)

  Encrypted(Y) = Y xor C(K)

  ⇨ Encrypted(X) xor Encrypted(Y) = X xor Y

  ⇨ If you know X (or Y), then you know the other

  ⇨ Even if you know neither, you can guess X and Y quite well from X xor Y if X and Y are in a natural language

  Sources: http://en.wikipedia.org/wiki/Stream_cipher_attack

- Used in the WEP wireless encryption
  - Employ an initialization vector of 24-bit, but insufficient

Public Key Cryptography

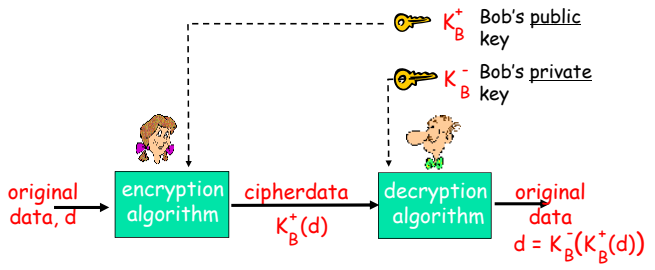
Symmetric key cryptography

- requires the knowledge of secret key
- Q: in network communication, how sender/receiver agree on key in the first place? (particularly difficult if adversary is eavesdropping on all communication)

Public key cryptography

- encryption key is different from decryption key
- encryption key is public, known to everyone, also called public key
- decryption key is secret, known only to receiver, also called private key

Public Key Cryptography



Principle for choosing the public/private key pair:
One should not be able to derive the private key from the public key.

Public Key Cryptography: RSA (Ron Rivest, Adi Shamir and Len Adleman)

- Choosing keys:
  - Choose two large prime numbers p, q. (e.g., 1024 bits each)
  - Compute n = pq, z = (p-1)(q-1)
  - Choose e (with e<n) that has no common factors with z
  - Choose f such that ef-1 is exactly divisible by z
  - Public key is (n,e). Private key is (n,f)
- To encrypt a dataset, d (<n): do $c = d^e \bmod n$
- To decrypt a encrypted cipherdata, c: do $d = c^f \bmod n$
- Reason: for any d (relatively prime with n)
  - $d^z \bmod n = 1$; therefore $d^{ef-1} \bmod n = 1$
- Another property: $(d^f \bmod n)^e \bmod n = d$

Public Key Cryptography: RSA

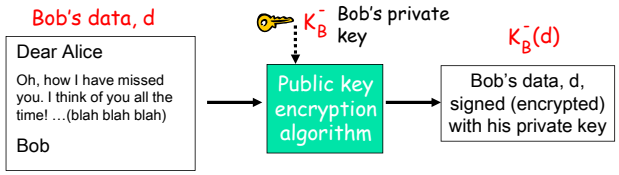
- RSA is much slower than the symmetric key cryptos
- In practice, you never use RSA to encrypt large datasets
  - Use symmetric key to encrypt large datasets
  - Then use RSA to protect the secret transfer of symmetric key

Data Integrity

- Digital Signatures:
  - cryptographic technique to ensure data integrity
  - analogous to hand-written signatures
- Data is attached with a digital signature which ensures that the data is
  - nonforgeable: data hasn't been changed since the signing
  - verifiable: data was signed by the right person

Digital Signatures

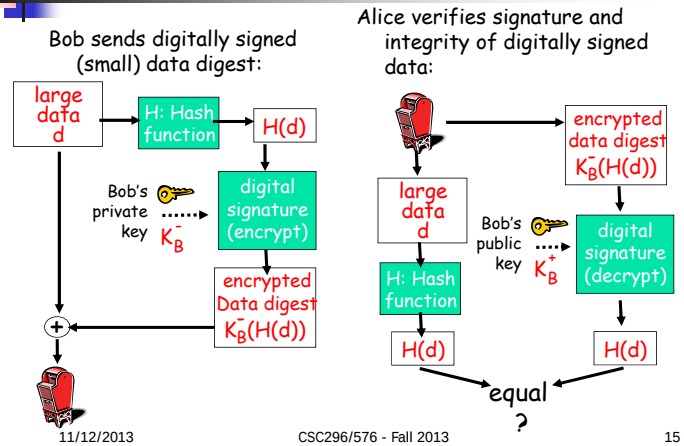
- Bob signs data by encrypting with his private key, creating a digital signature $K_B^-(d)$



- Suppose Alice receives d and its digital signature $K_B^-(d)$
- Alice applies Bob's public key K_B^+ to $K_B^-(d)$ then checks whether $K_B^+(K_B^-(d)) = d$.
- If so, whoever signed d must have used Bob's private key.

Problem: computationally expensive to public-key-encrypt large data.

Signed Data Digest



Data Digests

- Apply a hash function H to d, get a much smaller data digest H(d).
- Public-key-encrypt the data digest to generate the digital signature $K_B^-(H(d))$.

Good/bad hash functions?

- Hint: given a hash function, it is possible for many data instances sharing the same digest.
- Hash function property: given digest x for data d, computationally infeasible to find another data instance d' that shares the same digest.

Good Hash Functions for Generating Data Digests

- MD5
  - computes 128-bit data digest in 4-step process.
  - appears difficult to construct data d whose MD5 hash is equal to x.
- SHA-1
  - [NIST, FIPS PUB 180-1]
  - 160-bit data digest

Security Overhead for Big Data

- Tests on a 3.1GHz Intel Xeon processor (by Zhuan Chen) show that:
  - 128-bit AES encryption of 4KB data takes 41us; decryption takes 55us
  - SHA-1 hashing takes 10us
- 1TB data
  - Encryption takes 3 hours 3 minutes
  - Decryption takes 4 hours 6 minutes
  - Hashing takes 45 minutes
  - Parallelization can help
- Public-key-crypto (RSA) is much slow, but never directly use to protect large datasets

Disclaimer

- Preparation of this class was helped by materials in the book “Computer Networking: A Top-Down Approach” by Kurose and Ross.