CSE 451: Operating Systems
Winter 2001

Lecture 18
A Bit of Cryptography

Steve Gribble
gribble@cs.washington.edu
323B Sieg Hall

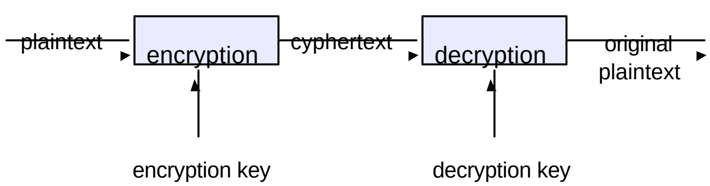
Brief Intro to Cryptography

- Much of this material taken from “Applied Cryptography” by Bruce Schneier
  - a highly recommended book; very approachable, few errors
- Cryptography:
  - the art and science of keeping messages secure
  - communicating in the presence of adversaries
    - practiced by cryptographers
    - in its essence, art of disguising a message (“plaintext”) in such a way as to hide its substance (turn it into “cyphertext”)
  - cryptanalysis
    - art and science of breaking cyphertext
    - cryptology = union(cyryptography, cryptanalysis)
- The basic idea
  - trust mathematics instead of people

Roles of Cryptography

- Confidentiality
  - hiding the content of messages
- Authentication
  - ascertain the origin of a message
- Integrity
  - verify that a message hasn't been modified in transit
- Nonrepudiation
  - prevent a sender from falsely denying sending a message

The Basic Idea



- $E_{k1}(M) = C$, $D_{k2}(C) = M$
  - $D_{k2}(E_{k1}(M)) = M$
- Symmetric algorithms (aka secret-key algorithms):
  - given k1, can deduce k2, and vice-versa
- Public-Key Algorithms
  - decryption key (k2) cannot be calculated from encryption key
  - encryption key can be made public!
    - encryption key = "public key", decryption key = "private key"

Communications using Symmetric Crypto

- Protocol for Alice and Bob to communicate securely:
  - 1. Alice and Bob agree on a cryptosystem (e.g., DES)
  - 2. Alice and Bob agree on a key. (how?)
  - 3. Alice encrypts plaintext with algo+key
  - 4. Alice sends the ciphertext to Bob
  - 5. Bob decrypts the ciphertext with same algo+key
- Claims:
  - this gives 3 of 4 desired properties (confidentiality, authentication, integrity). why not nonrepudiation?
- Weaknesses:
  - which steps must Alice and Bob protect from Eve[sdropper]?
  - what could Mallory (man in the middle) to do harm A&B?
  - what happens if Alice is subverted?

Public-Key Cryptography

- Symmetric-key crypto is like a safe
  - public-key crypto is like a mailbox
- Protocol to communicate with public-key crypto
  - 1. Alice/Bob agree on a public-key cryptosystem (e.g. RSA)
  - 2. Bob sends Alice his public key
  - 3. Alice encrypts her message using Bob's public key
  - 4. Alice sends ciphertext to Bob
  - 5. Bob decrypts Alice's message using his private key
- Which properties does this give Alice and Bob?
- Weaknesses?
  - what must Alice/Bob protect from Eve?
  - what could Mallory do?

Hybrid Cryptosystem

- But, public key cryptography is 1000 times as expensive as symmetric key crypto
  - also, susceptible to chosen-plaintext attacks
    - e.g., if only 1000 possible messages, attacker could simply encrypt all 1000 with public key to get a dictionary of ciphertext
- Hybrid cryptosystem
  - 1. Bob sends Alice his public key.
  - 2. Alice generates a random session key K, encrypts it using Bob's public key, and sends it to Bob. $E_b(K)$
  - 3. Bob decrypt's Alice's message using private key to recover session key. $D_b(E_b(K)) = K$
  - 4. Alice and Bob communicate using symmetric cryptography, using key K.

Beware Randomness

- Is your random number truly random?
  - Donald Knuth quotes John von Neumann:
    - Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
  - typical random number generation libraries aren't that random (e.g., rand() or rand48())
    - emit a deterministic sequence of numbers
    - sequence isn't that random, e.g., rand() flips low-order bit
- Netscape crack:
  - used hybrid cryptosystem (SSL)
  - session key chosen by random number generation
    - RNG was seeded using time-of-day, process IDS, etc.
    - very guessable!
      - my Berkeley officemates reverse-engineered Netscape binary, and wrote code to do online attack of SSL connections
  - better: use physical process (noise on electrical conductor, geiger counter), or very low-level stuff (interrupt arrival times, timing of keystrokes)

One-Way Hash Functions

- Hash function
  - takes a variable-lengthed input string (pre-image) and produces a fixed-length, smaller output string (hash value)
  - essentially fingerprints the pre-image
    - e.g.: return byte consisting of XOR of all preimage bytes
- One-way hash function (e.g., MD5)
  - easy to compute hash from preimage, but hard to generate preimage that hashes to particular value
    - how hard? brute force
      - birthday paradox…
  - useful in many places (as we'll see)
    - e.g.: to confirm somebody has a file intact, ask her to compute a one-way hash on the file, and verify that. (integrity)

Example Use of Hash Functions

- UNIX passwords
  - when you log in, how is your password verified?
    - naïve strategy: OS keeps a file with everybody's password in it
      - what happens if bad-guy gets that file? all passwords are lost!
    - better strategy: OS keeps a file with hash(password) in it
      - one-way hash protects bad-guy from getting passwords
    - /etc/passwd file entries look like:

```
gribble:AEFEF.hKbYNEU:10046:116:Steve Gribble:/home/gribble:/bin/bash
```

- Weakness: dictionary attack
  - compile list of 1,000,000 common passwords
    - amongst most popular: sex, secret, password, gandalf, god
  - run them each through one-way hash
    - compile a dictionary of popular hashed passwords
  - solution? salt:
    - passwd file entry = hash(salt+password)
    - store salt in clear on host
    - bad-guy can't use pregenerated dictionary file
    - UNIX: uses 12-bit salt (only makes it 4096 times harder to crack)

Digital Signatures

- Why are handwritten signatures good?
  - authentic: convinces document recipient that signer deliberately signed the document
  - unforgeable: proof that the signer, and no-one else, signed the document
  - not reusable: signature is part of document, can't be moved to another document
  - unalterable: after document is signed, it can't be altered
  - non-repudiable: signer cannot later claim didn't sign it
- Of course, none of these things are true for analog handwritten signatures
  - can we do better with digital signatures?

Public-Key based Digital Signatures

- Protocol is really simple:
  - 1. Alice encrypts document with her private key.
  - 2. Alice sends encrypted document to Bob.
  - 3. Bob decrypts the document with Alice's public key, thereby verifying the signature.
- Properties?
  - authentic: Bob verifies message with Alice's public key.
  - unforgeable: only Alice knows her private key.
  - not reusable: signature is function of document, can't be transferred.
  - unalterable: altering the document will alter the produced plaintext, producing gibberish
  - non-repudiation: only Alice knows her private key, thus only she could sign document. Plus, Bob doesn't need Alice's help to verify the signature.

Problems?

- Timestamps:
  - Bob can reuse document+signature
    - what if it is a signed cheque?
  - for this reason, digital signatures often contain timestamps
    - signed document = E_a(doc+timestamp)
      - bank stores timestamp, can detect copied cheque
- Public key crypto is expensive!
  - what if want to sign the human genome sequence?
    - one-way hashes to the rescue…
  - new protocol:
    - 1. Alice produces a one-way hash of a document
    - 2. Alice encrypts the hash value with private key
    - 3. Alice sends plaintext document + signed hash to Bob
    - 4. Bob produces one-way hash of document. He also decrypts signed hash with Alice's public key, and compares with his hash. If match, signature is valid.

Combining Signatures and Encryption

- What if you want to securely send a secret file to somebody, and let them validate it came from you?
  - letter from Mom: signature as proof of authorship, letter to provide privacy from prying eyes
- Protocol:
  - 1. Alice signs message with her private key
    - $S_A(M)$
  - 2. Alice encrypts the signed message with Bob's public key, and sends it to Bob
    - $E_A(S_A(M))$
  - 3. Bob decrypts message with his private key
    - $D_b(E_b(S_A(M))) = S_A(M)$
  - 4. Bob verifies with Alice's public key and recovers message
    - $V_A(S_A(M)) = M$

Bit-commitment

- Imagine:
  - Bob: pick 5 stocks for me, if you're good on all 5, I'll hire you as my stock broker.
  - Alice: if I tell you 5 good stocks, you don't need me! How about I predict 5, and tell you them in a month?
  - Bob: but, how do I know you won't change your answer?
- Answer: bit commitment
  - Alice wants to commit to a prediction (a sequence of bits), but doesn't want to reveal her prediction until later.
    - Bob wants to make sure Alice doesn't change her mind after committing to the bits.

Bit-commitment algorithms

- Using symmetric crypto:
  - 1. Bob generates a random-bit string R, sends to Alice
    - R
  - 2. Alice creates message containg bit she wishes to commit concatenated with Bob's random string. She encrypts message with random key K, sends result to Bob.
    - $E_K(R,b)$
    - the bit is now committed..
  - 3. When it's time to reveal the bit, Alice sends Bob the key
    - K
  - 4. Bob decrypts the message to reveal the bit. He also checks his random string to verify the bit's validity.
    - $R,b = D_K(E_K(R,b))$
- Why does Bob need to provide the random string?

Bit-Commitment Using One-Way Functions

- 1. Alice generates two random -bit strings, R1 and R2
  - R1, R2
- 2. Alice creates message containing her random strings and the bit she wishes to commit
  - (R1, R2, b)
- 3. Alice computes one-way hash on message, sends it and R1 to Bob
  - H(R1,R2,b), R1
  - commitment is now done.
- 4. When time to reveal, Alice sends Bob full message
  - (R1,R2,b)
- 5. Bob computes one-way hash of message, compares it and R1 with the hash and value he received in step 3. If match, bit is valid.
- Note that Bob didn't need to send Alice any messages at all!
  - a strong benefit over symmetric crypto algorithm
  - could imagine posting committed bit on random newsgroup
  - Bob retains anonymity

3/4/2001 17

Many other cool crypto tricks

- Zero-knowledge proofs
  - prove know information without revealing it
- Secure elections and auctions
- Anonymous broadcasts
- Digital cash
- Anonymous mail (messaging)
- Pseudonymity
  - persistent, untraceable identities
- …many many more!

3/4/2001 18