Lecture 2

network protocol issues
secret key crypto issues
Introduction to “layers”

• OSI Reference Model
  – 1: physical
  – 2: (data link) neighbor-neighbor
  – 3: (network) create entire path
  – 4: (transport) end-to-end
  – 5-7: application, data representation
Service of layer

• Each layer provides some API to that above it
  – reliable: bit stream or packets, arrive in same order at other end
  – datagram: send chunk of data, maybe it gets there, possibly lost, duplicated, out-of-order. Usually detects nonmalicious errors
Nonmalicious integrity checks

- send it multiple times
- VRC (vertical redundancy check) parity within byte
- LRC (parity over a bit of all the bytes in a block)
- LRC+VRC catches what bit errors?
- two’s complement add (ordinary add)
- one’s complement add (add in carries). Why is this better than 2’s complement?
- CRC (make message + checksum divisible by the “CRC polynomial”)
Intuition behind CRC

- Pretend message is long number
- Suppose “CRC polynomial” is some number, say 17
- Suppose message is 5283
- Multiply it by 100=528300
- 528300 is 8 mod 17
- Subtract 8 from 528300=528292. Send that
- Rcvr checks divisibility by 17. If not, error. If so, round to nearest 100, then truncate to get message
Cryptographic Integrity Checks

- Requires knowledge of secret/private key to generate
- With secret key: requires knowledge of secret to verify
- With public key: requires knowledge of public key to verify
Two-way reliable protocols

- Used sometimes in layer 2, sometimes in layer 4 (e.g., TCP, but not UDP)
- Was more popular in layer 2 when lossy links
- Basic idea the same, sequence number on data, sequence number on ack, pipelining
- If $n$ sequence numbers (and wrap around), how many can you send?
Reliable 2 vs 4

- reordered messages
- variable and unpredictable delay
- unpredictable packet size
- different multiplexing (layer 3 is small number of permanent processes, users of layer 5 are often dynamic processes)
DNS

- distributed database for looking up names and getting IP addresses (etc)
- control is delegated. If you control foo.com, you can assign names of the form *.foo.com, or delegate a subdomain
- each client must know at least one server IP address (maybe learned with dhcp)
- each server knows addresses of subdomain servers, and at least one root
DNS lookups

• Sequence might be: your local server, the root, child, child, until target
• Server can answer your request (recursive) or tell you who to ask next (iterative)
• Caching at all places
What happens in a network

- User types “talk to FOO”, a user-friendly name
- DNS looks it up to find an IP address
- network infrastructure (routers) cooperate to calculate paths to IP addresses
- one side needs to be easily findable, and listens for calls
Lower Layers

• Layer 3 (e.g., IP) is just an envelope in which you specify source, destination address (and hop count)

• Layer 2 used to be pt-to-pt links, not needing an address, but became LANs, with their own addresses (“MAC” address)

• The way IP works is that all nodes on the LAN share a prefix
IP Forwarding

• You can tell from address if someone is a neighbor (iff same prefix)
• If so, send to them, but need their layer 2 address
• Use ARP…broadcast “who has this layer 3 address” get back reply from them
• If not on same link, send to router
How to find router

• Lots of ad hoc methods. Was better designed in ISO’s layer 3 (CLNP). In IP, some just configured with a rtr address
• Thus comes in VRRP (virtual router redundancy protocol)
• Don’t design security unless you understand what threats you’re addressing
VRRP

- A bunch of routers on the LAN elect one of them to have the IP address “R1”, and associated MAC address. That one periodically issues “I’m still alive” messages to the other VRRP routers, sending from its MAC address.
Interacts with Bridges

- Bridges/switches are “invisible” layer 2 devices which listen promiscuously and learn the location of stations based on the source address in the layer 2 header
VRRP Security

• 2 types
  – cleartext password
  – cryptographic

• What do these solve?
Review of crypto from last lecture

- secret key: 2 operations, inverses
- public key: one op, 2 keys (public, private), which are inverses
- operations
  - encrypt with secret/public, decrypt with secret/private
  - authenticate with secret/private, verify with secret/public
  - compute integrity with secret/private, verify with secret/public
Authentication with Secret Key

both know secret $K$

Alice    Bob

I’m Alice

R

$\{R\}_K$

compare:

or:

I’m Alice, $\{\text{timestamp}\}_K$
Secret Key Encryption for Privacy

Plaintext $\rightarrow$ Encrypt $\rightarrow$ Ciphertext $\rightarrow$ Decrypt $\rightarrow$ Plaintext
Integrity-Protection with Secret Key

• Stay awake…will cover it later in lecture
Secret Key algorithms

- **Stream ciphers (e.g., RC4)**
  - takes key and generates a stream of pseudorandom bits, XOR’d into data

- **Block ciphers (e.g., DES, AES)**
  - takes key and fixed size input block to generate fixed size output block
  - How many mappings are there if n-bit blocks?
  - How many can be specified with k-bit key?
XOR (Exclusive-OR)

- Bitwise operation with two inputs where the output bit is 1 if exactly one of the two input bits is one
- \((B \text{ XOR } A) \text{ XOR } A) = B\)
- If A is a “one time pad”, very efficient and secure
- Common encryption schemes (e.g. RC4) calculate a pseudo-random stream from a key
Types of attacks

• ciphertext only: can brute-force attack if recognizable plaintext.

• sometimes a system allows other attacks:
  – known plaintext
  – chosen plaintext

• Shannon proved XOR with one-time pad unbreakable (no information with brute force attack)
Block size considerations

• If small block size, could build a table
• If see same ciphertext block, get hints about plaintext
• to avoid probably seeing repeated ciphertext blocks, should change key in number of blocks $2^{\text{half the block size}}$ (birthday problem)
• what is the max effective key size with a block size of $k$ bits?
Authentication

• “factors”
  – what you know (e.g., password)
  – what you have (e.g., physical object)
  – what you are (e.g., biometric)

• Something is said to be “2 factor” if it requires two of the above (not two passwords!)
Token cards

• vs smart card: token does not attach to computer (or “human” is the interface)
• token is self-contained computer and has:
  – its own secret, known by the server
  – display (usually 8 digits, each 16 possibilities)
  – possibly keyboard
  – human inputs value from display into computer
  – typically engineered to require 2-factor
Types of token cards

- challenge/response
- time-based
- sequence based
Encrypting Large Messages

- The basic algorithms encrypt a fixed size block.
- Obvious solution is to encrypt a block at a time. This is called Electronic Code Book (ECB).
- Repeated plaintext blocks yield repeated ciphertext blocks.
- Other modes “chain” to avoid this (CBC, CFB, OFB).
- Encryption does not guarantee integrity!
Problems with ECB

- If $c_i = c_j$, then you know $p_i = p_j$
- Can reorder blocks
- Can rearrange blocks to affect plaintext
Consider this

transmit r1, c1, r2, c2, r3, c3, r4, c4
Problems with previous slide

- Need to send twice as much data
- Can still rearrange blocks
- If two ciphertext blocks equal, know XOR of two plaintext blocks = XOR of the corresponding two random numbers
- CBC generates its own “random numbers” by using previous ciphertext block, plus one additional block (the “IV”, initialization vector)
CBC (Cipher Block Chaining)
CBC Decryption
CBC

• What happens if ci gets lost? Garbled? How much data gets lost?
• How can attacker that sees and can modify the ciphertext, and knows the plaintext, modify the plaintext in a predictable way? What other effects will it have?
Creating a stream cipher

- (OFB) stream generated:
  - IV (transmitted in the clear)
  - pad_1 = e(IV, key)
  - pad_2 = e(pad_1, key)
  - pad_i = e(pad_{i-1}, key)

- Can be generated in advance
- Can encrypt arbitrary # of bits (vs block cipher)
- What is ciphertext garbled or lost?
- If know plaintext, can easily modify stream
Counter Mode

- $c_i = f\text{(key, IV, block number, } p_i)$
- Can decrypt an arbitrary block (useful for, e.g., random access file encryption)
Triple DES (3DES)

- **Purpose:** expand key size from 56 bits (almost enough until recently) to >80 bits or so

- **Why not double DES?**
  - encrypt with $K_1$ twice. How much more work (over DES) for good guys? Bad guys?
  - encrypt with $K_1$ then $K_2$. What is time/memory for bad guys? Good guys?
Triple DES (3DES)

- Defined as doing EDE with K1, K2, K3, but standardly K1 is set equal to K3.
  - reason: because of “meet-in-the-middle” attack, 3DES is considered to only have time-strength equal to 112 bit key, not 168.
  - also, 112 bits considered enough.
Why EDE instead of EEE?

- Initial and final permutations would cancel each other out with EEE (minor advantage to EDE)
- EDE compatible with single DES if $K1=K2=K3$. 
Integrity Check (MAC) with Secret Key

• “CBC Residue”, uses key K and generates integrity check on message M
• Do CBC encryption on M using key K, throw away all but last block. Last block is “residue”, and is used as integrity check
• Used in banking
• Has property that if you don’t know the key you can’t generate (or verify) the MAC, or modify the message without (probably) changing the MAC
CBC Residue

IV → M1 → M2 → M3 → M4

E → C1 → E → C2 → E → C3 → residue
CBC Residue

- Note that it is easy to generate an arbitrary message with a particular residue
Feistel Cipher Encryption

\[ L_n \quad R_n \]

\[ L_{n+1} \quad R_{n+1} \]

mangler

\[ K_n \]
Feistel Cipher Decryption

\[
\begin{array}{c|c}
L_n & R_n \\
\hline
L_{n+1} & R_{n+1}
\end{array}
\]

\[mangler\]

\[K_n\]
Feistel Cipher Decryption

A  B

C  D

mangler

$K_n$
Why Feistel

• So Mangler function doesn’t need to be reversible
  – $\text{enc}(A,B): C=B$, $D=x(A,m(B))$
  – $\text{dec}(C,D): B=C,$
    \[ A=x(D, m(C))=x(A,m(B),m(B)) \]

• DES is Feistel

• AES, IDEA not. All functions are reversible.