CS 161: Computer Security
Midterm 1 Review

Part 2

October 4, 2006

Shamir Secret Sharing

Sharing a secret

- Let's say we want to hide a secret with t people such that any q of them can reconstruct the secret but any $q-1$ cannot
- We can use a polynomial with random coefficients:

$$f(x) = x^q + a_{q-1}x^{q-1} + \cdots + a_1 x + a_0 (\mathrm{mod}\, p)$$

- The secret is $f(0) = a_0$, and the shares we distribute are $f(1), f(2), \ldots, f(t)$ (the other coefficients are also kept secret)
- Now any q people can solve the polynomial for a_0 but $q-1$ have no information
- Note: Could do something similar with real numbers, but integers are easier so we use modular arithmetic

Secret sharing: simple example

- Let $q = 3$ and $p = 11$ (don't confuse with the p, q of RSA):

$$f(x) = x^3 + a_2x^2 + a_1x + a_0 (\text{mod } 11)$$

- Let's say the secret we want to hide is "5", and we randomly choose coefficients $a_2 = 3$ and $a_1 = 9$
- For $t = 6$, we compute $f(1) = 7$, $f(2) = 10$, $f(3) = 9$, $f(4) = 10$, $f(5) = 8$, and $f(6) = 9$.
- Any three can now solve for coefficients:

$$\begin{array}{rcrcrcrcrcl} f(2) & = & 10 & = & 8 & + & 4a_2 & + & 2a_1 & + & a_0 \quad (\text{mod } 11) \\ f(3) & = & 9 & = & 5 & + & 9a_2 & + & 3a_1 & + & a_0 \quad (\text{mod } 11) \\ f(6) & = & 9 & = & 7 & + & 3a_2 & + & 6a_1 & + & a_0 \quad (\text{mod } 11) \end{array}$$

Zero-Knowledge Protocols

Zero-knowledge proof of identity

- Goal 1: Alice knows that the person she's talking to is Bob.
- Goal 2: Bob reveals no additional information to Alice.
- Assumptions:
  - Alice knows Bob's public key
  - Taking square roots modulo n is "hard"
- Protocol (everything mod pq):
  - Bob picks secret b, publishes b^2 as public key (persistent)
  - Alice wants to check Bob's identity, asks Bob to begin
  - Bob picks random r (new each run), sends "commitment" r^2 to Alice
  - Alice flips coin (or chooses); heads means reveal r (Alice verifies r^2), tails means reveal rb (Alice verifies r^2b^2)
  - If Bob passes, Alice is 50% convinced of his identity
  - Repeat arbitrarily many times

How can Mallory pretend to be Bob?

- Doesn't know b (because he can't factor), but can game the system
- To know the "heads" answer, picks r like Bob would
  - When Alice sends "heads", he sends r
  - If Alice sends "tails", what can he do? nothing!
- To know the "tails" answer, picks t and computes $r^2 = t^2/b^2$ to send to Alice
  - When Alice sends "tails", he sends t
  - Alice thinks he sent rb and checks by squaring it, but $(rb)^2 = r^2b^2 = (t^2/b^2)b^2 = t^2$ so fake Bob has fooled Alice on tails
  - If Alice sends "heads", he can't do anything because he didn't pick r and then square it, he just picked something that he called r^2, but he can't take the square root

Why doesn’t the protocol leak information to Alice?

- When she flips "heads", Bob just reveals a random number, which Alice could have picked by herself
- When she flips "tails", Bob sends her *rb*, but this is also random because multiplication induces a permutation, so a random number times anything lands on a random value

Authentication

Authentication

- Authentication is verifying an identity, or verifying the originator of a message
- Many types of authentication
  - Person $\rightarrow$ person
  - Person $\rightarrow$ local computer
  - Remote computer $\rightarrow$ person
  - etc.
- Difficult to get right and easy to screw up
- Most real attacks today are authentication attacks: phishing, “pretexting”, spyware password pop-ups, etc.

Needham-Schroeder

- Symmetric encryption with trusted server
- Each user shares symmetric key with server (A shares key a, B shares key b, etc.)
- A $\rightarrow$ S: $\{B\}_a$
- S $\rightarrow$ A: $\{t, \{A, t\}_b\}_a$
- A $\rightarrow$ B: $\{A, t\}_b$
- A $\leftrightarrow$ B: $\{$ messages. . . $\}_t$

Problem and fix

- Replay attack:
  - M $\rightarrow$ B: $\{A, t\}_b$
  - M $\rightarrow$ B: { something that shouldn't be repeated $\}_t$
- Solution: nonces (unique value, such as a random number or timestamp)
- Revised N-S: every message has timestamp, so attacker can't replay
- Problems remain: requires real-time trusted third party

Firewalls

Firewall overview

- Motivation: Every network service is a potential hole
- Block services in the network before they reach machines
- Enforces *security policy:* policy on which services should be visible, which should be blocked, and how we distinguish insiders from outsiders
- Default allow vs. default deny
  - Default allow is easier on users and bothers them less
  - Default deny is more secure in several ways: fails safe, catches unknown attacks, hedges against common mistakes

Packet filters

- Checks each packet against series of rules
- Rules test IP, protocol, port, etc. to decide *drop* or *allow*
- The first matching rule decides the action
- Syntax: $\langle action \rangle\ \langle proto \rangle\ \langle addr \rangle \rightarrow \langle addr \rangle$
- Each $\langle addr \rangle$ is of the form $\langle ip \rangle{:}\langle port \rangle[/\{in,out\}]$
- The * wildcard matches any value
- (*in/out* difference is which interface packet received on)

Firewall example

- `allow tcp *:*/out` → `1.2.3.4:25/in`
- `allow tcp *:*/in` → `*:*/out`
- `allow tcp *:*/out` → `*:*/in (if ACK is set)`
- `drop * *:*` → `*:*`

More on firewalls

- Reference monitor
  - Mediates all access to network
  - Three requirements: Always invoked, tamper resistant, verifiable
- Application-level firewall can do more than packet filter
  - Inspect and enforce application protocols
  - Do more nuanced filtering
- Stateful firewall can do more
  - Assemble TCP connections (not just look for ACK)
  - Limit number of open requests, etc.
- VPNs extend perimeter over secure channel to remote machine
- Good for working from home, bad if home computer gets virus