CSci 4271W
Development of Secure Software Systems
Day 21: Networking and security

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Precise explanations

- Don't say "we" do something when it's the computer that does it
  - And avoid passive constructions
- Don't anthropomorphize (computers don't "know")
- Use singular by default so plural provides a distinction:
  - – The students take tests
  - + Each student takes a test
  - + Each student takes two tests

Provide structure

- Use plenty of sections and sub-sections
- It's OK to have some redundancy in previewing structure
- Limit each paragraph to one concept, and not too long
  - Start with a clear topic sentence
- Split long, complex sentences into separate ones

Know your audience: Project 0.5

- For projects in this course, assume your audience is another student who already understands general course concepts
  - Up to the current point in the course
  - I.e., don't need to define "buffer overflow" from scratch
- But you need to explain specifics of a vulnerable program
  - Make clear what part of the program you're referring to
  - Explain all the specific details of a vulnerability

Inclusive language

- Avoid words and grammar that implies relevant people are male
- My opinion: avoid using he/him pronouns for unknown people
- Some possible alternatives
  - "he/she"
  - Alternating genders
  - Rewrite to plural and use "they" (may be less clear)
  - Singular "they" (least traditional, but spreading)

Outline

General description

- Public-key encryption (generalizes block cipher)
  - Separate encryption key EK (public) and decryption key DK (secret)
- Signature scheme (generalizes MAC)
  - Separate signing key SK (secret) and verification key VK (public)

RSA setup

- Choose $n = pq$, product of two large primes, as modulus
- n is public, but p and q are secret
- Compute encryption and decryption exponents e and d such that

$$M^{ed} = M \pmod{n}$$

RSA encryption

- Public key is (n, e)
- Encryption of M is $C = M^e \pmod{n}$
- Private key is (n, d)
- Decryption of C is $C^d = M^{ed} = M \pmod{n}$

RSA signature

- Signing key is (n, d)
- Signature of M is $S = M^d \pmod{n}$
- Verification key is (n, e)
- Check signature by $S^e = M^{de} = M \pmod{n}$
- Note: symmetry is a nice feature of RSA, not shared by other systems

RSA and factoring

- We're not sure factoring is hard (likely not even NP-complete), but it's been unsolved for a long time
- If factoring is easy (e.g., in P), RSA is insecure
- Converse might not be true: RSA might have other problems

Homomorphism

- Multiply RSA ciphertexts $\Rightarrow$ multiply plaintexts
- This *homomorphism* is useful for some interesting applications
- Even more powerful: fully homomorphic encryption (e.g., both $+$ and $\times$)
  - First demonstrated in 2009; still challenging

Problems with vanilla RSA

- Homomorphism leads to chosen-ciphertext attacks
- If message and e are both small compared to n, can compute $M^{1/e}$ over the integers
- Many more complex attacks too

Hybrid encryption

- Public-key operations are slow
- In practice, use them just to set up symmetric session keys

\+ Only pay RSA costs at setup time

− Breaks at either level are fatal

Padding, try #1

- Need to expand message (e.g., AES key) size to match modulus
- PKCS#1 v. 1.5 scheme: prepend 00 01 FF FF .. FF
- Surprising discovery (Bleichenbacher'98): allows adaptive chosen ciphertext attacks on SSL
  - Variants recurred later (c.f. "ROBOT" 2018)

Modern "padding"

- Much more complicated encoding schemes using hashing, random salts, Feistel-like structures, etc.
- Common examples: OAEP for encryption, PSS for signing
- Progress driven largely by improvement in random oracle proofs

Simpler padding alternative

- "Key encapsulation mechanism" (KEM)
- For common case of public-key crypto used for symmetric-key setup
  - Also applies to DH
- Choose RSA message r at random mod n, symmetric key is $H(r)$
- – Hard to retrofit, RSA-KEM insecure if e and r reused with different n

Post-quantum cryptography

- One thing quantum computers would be good for is breaking crypto
- Square root speedup of general search
  - Countermeasure: double symmetric security level
- Factoring and discrete log become poly-time
  - DH, RSA, DSA, elliptic curves totally broken
  - Totally new primitives needed (lattices, etc.)
- Not a problem yet, but getting ready

Box and locks revisited

- Alice and Bob's box scheme fails if an intermediary can set up two sets of boxes
  - Middleperson (man-in-the-middle) attack
- Real world analogue: challenges of protocol design and public key distribution

Outline

Good technical writing (cont'd)

Public key encryption and signatures

Announcements intermission

Brief introduction to networking

Some classic network attacks

Cryptographic protocols

Deadline reminders

- OWASP reading questions: tonight
- Project 0.5 regular deadline: Wednesday night
- Project one-time extension: to Friday night

Outline

Good technical writing (cont'd)

Public key encryption and signatures

Announcements intermission

Brief introduction to networking

Some classic network attacks

Cryptographic protocols

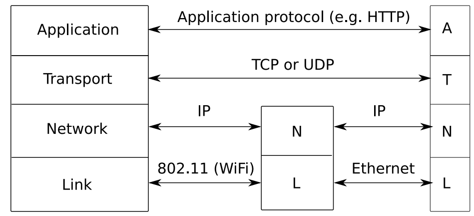
The Internet

- A bunch of computer networks voluntarily interconnected
- Capitalized because there's really only one
- No centralized network-level management
  - But technical collaboration, DNS, etc.

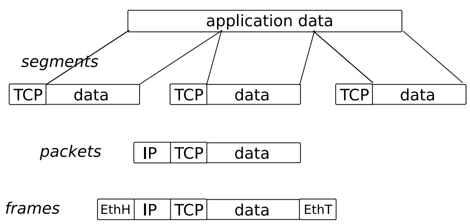
Layered model (OSI)

7. Application (HTTP)
6. Presentation (MIME?)
5. Session (SSL?)
4. Transport (TCP)
3. Network (IP)
2. Data-link (PPP)
1. Physical (10BASE-T)

Layered model: TCP/IP



Packet wrapping



IP(v4) addressing

- Interfaces (hosts or routers) identified by 32-bit addresses
  - Written as four decimal bytes, e.g. 192.168.10.2
- First k bits identify network, $32 - k$ host within network
  - Can't (anymore) tell k from the bits
- We'll run out any year now

IP and ICMP

- Internet Protocol (IP) forwards individual packets
- Packets have source and destination addresses, other options
- Automatic fragmentation (usually avoided)
- ICMP (I Control Message P) adds errors, ping packets, etc.

UDP

- User Datagram Protocol: thin wrapper around IP
- Adds source and destination port numbers (each 16-bit)
- Still connectionless, unreliable
- OK for some small messages

TCP

- Transmission Control Protocol: provides reliable bidirectional stream abstraction
- Packets have sequence numbers, acknowledged in order
- Missed packets resent later

Flow and congestion control

- Flow control: match speed to slowest link
  - "Window" limits number of packets sent but not ACKed
- Congestion control: avoid traffic jams
  - Lost packets signal congestion
  - Additive increase, multiplicative decrease of rate

Routing

- Where do I send this packet next?
  - Table from address ranges to next hops
- Core Internet routers need big tables
- Maintained by complex, insecure, cooperative protocols
  - Internet-level algorithm: BGP (Border Gateway Protocol)

Below IP: ARP

- Address Resolution Protocol maps IP addresses to lower-level address
  - E.g., 48-bit Ethernet MAC address
- Based on local-network broadcast packets
- Complex Ethernets also need their own routing (but called switches)

DNS

- Domain Name System: map more memorable and stable string names to IP addresses
- Hierarchically administered namespace
  - Like Unix paths, but backwards
- `.edu` server delegates to `.umn.edu` server, etc.

DNS caching and reverse DNS

- To be practical, DNS requires caching
  - Of positive and negative results
- But, cache lifetime limited for freshness
- Also, reverse IP to name mapping
  - Based on special top-level domain, IP address written backwards

Classic application: remote login

- Killer app of early Internet: access supercomputers at another university
- Telnet: works cross-OS
  - Send character stream, run regular login program
- rlogin: BSD Unix
  - Can authenticate based on trusting computer connection comes from
  - (Also rsh, rcp)

Outline

Good technical writing (cont'd)

Public key encryption and signatures

Announcements intermission

Brief introduction to networking

Some classic network attacks

Cryptographic protocols

Packet sniffing

- Watch other people's traffic as it goes by on network
- Easiest on:
  - Old-style broadcast (thin, "hub") Ethernet
  - Wireless
- Or if you own the router

Forging packet sources

- Source IP address not involved in routing, often not checked
- Change it to something else!
- Might already be enough to fool a naive UDP protocol

TCP spoofing

- Forging source address only lets you talk, not listen
- Old attack: wait until connection established, then DoS one participant and send packets in their place
- Frustrated by making TCP initial sequence numbers unpredictable
  - Fancier attacks modern attacks are "off-path"

ARP spoofing

- Impersonate other hosts on local network level
- Typical ARP implementations stateless, don't mind changes
- Now you get victim's traffic, can read, modify, resend

rlogin and reverse DNS

- rlogin uses reverse DNS to see if originating host is on whitelist
- How can you attack this mechanism with an honest source IP address?

rlogin and reverse DNS

- rlogin uses reverse DNS to see if originating host is on whitelist
- How can you attack this mechanism with an honest source IP address?
- Remember, ownership of reverse-DNS is by IP address

Outline

Good technical writing (cont'd)

Public key encryption and signatures

Announcements intermission

Brief introduction to networking

Some classic network attacks

Cryptographic protocols

A couple more security goals

- Non-repudiation: principal cannot later deny having made a commitment
  - I.e., consider proving fact to a third party
- Forward secrecy: recovering later information does not reveal past information
  - Motivates using Diffie-Hellman to generate fresh keys for each session

Abstract protocols

- Outline of what information is communicated in messages
  - Omit most details of encoding, naming, sizes, choice of ciphers, etc.
- Describes honest operation
  - But must be secure against adversarial participants
- Seemingly simple, but many subtle problems

Protocol notation

$$A \to B : N_B, \{T_0, B, N_B\}_{K_B}$$

- $A \to B$: message sent from Alice intended for Bob
- B (after :): Bob's name
- $\{\cdots\}_K$: encryption with key K

Example: simple authentication

$A \to B : A, \{A, N\}_{K_A}$

- E.g., Alice is key fob, Bob is garage door
- Alice proves she possesses the pre-shared key K_A
  - Without revealing it directly
- Using encryption for authenticity and binding, not secrecy

Nonce

$A \to B : A, \{A, N\}_{K_A}$

- N is a *nonce*: a value chosen to make a message unique
- Best practice: pseudorandom
- In constrained systems, might be a counter or device-unique serial number

Replay attacks

- A nonce is needed to prevent a verbatim replay of a previous message
- Garage door difficulty: remembering previous nonces
  - Particularly: lunchtime/roommate/valet scenario
- Or, door chooses the nonce: *challenge-response* authentication

Middleperson attacks

- Older name: man-in-the-middle attack, MITM
- Adversary impersonates Alice to Bob and vice-versa, relays messages
- Powerful position for both eavesdropping and modification
- No easy fix if Alice and Bob aren't already related

Chess grandmaster problem

- Variant or dual of middleperson
- Adversary forwards messages to simulate capabilities with his own identity
- How to win at correspondence chess
- Anderson's MiG-in-the-middle

Anti-pattern: "oracle"

- Any way a legitimate protocol service can give a capability to an adversary
- Can exist whenever a party decrypts, signs, etc.
- "Padding oracle" was an instance of this at the implementation level