

CSci 4271W  
Development of Secure Software Systems  
Day 20: Public-key Cryptography and Networking

Stephen McCamant  
University of Minnesota, Computer Science & Engineering

### Preview question

Which of the following would have to be completely abandoned if scalable quantum computers become widely available?

- A. one-time pads
- B. RSA
- C. AES
- D. ROT13
- E. SHA-3

### Outline

- Public-key crypto basics
- Public key encryption and signatures
- Brief introduction to networking
- Some classic network attacks

### Pre-history of public-key crypto

- First invented in secret at GCHQ
- Proposed by Ralph Merkle for UC Berkeley grad security class project
  - First attempt only barely practical
  - Professor didn't like it
- Merkle then found more sympathetic Stanford collaborators named Diffie and Hellman

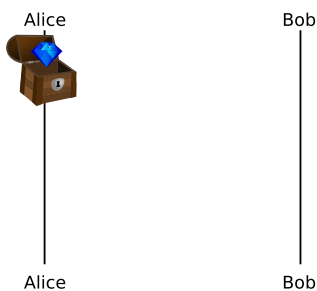
### Box and locks analogy

- Alice wants to send Bob a gift in a locked box
  - They don't share a key
  - Can't send key separately, don't trust UPS
  - Box locked by Alice can't be opened by Bob, or vice-versa

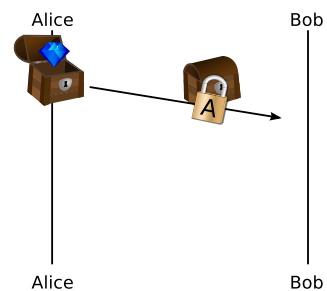
### Box and locks analogy

- Alice wants to send Bob a gift in a locked box
  - They don't share a key
  - Can't send key separately, don't trust UPS
  - Box locked by Alice can't be opened by Bob, or vice-versa
- Math perspective: physical locks commute

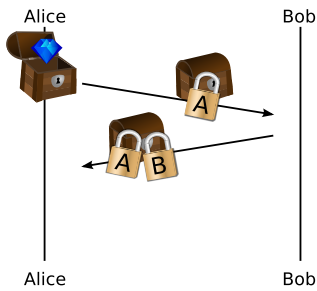
### Protocol with clip art



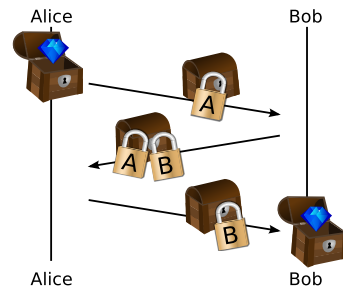
### Protocol with clip art



### Protocol with clip art



### Protocol with clip art



### Public key primitives

- 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)

### Modular arithmetic

- Fix *modulus*  $n$ , keep only remainders mod  $n$ 
  - mod 12: clock face; mod  $2^{32}$ : unsigned int
- +, -, and  $\times$  work mostly the same
- Division? Multiplicative inverse by extended GCD
- Exponentiation: efficient by square and multiply

### Generators and discrete log

- Modulo a prime  $p$ , non-zero values and  $\times$  have a nice ("group") structure
- $g$  is a *generator* if  $g^0, g, g^2, g^3, \dots$  cover all elements
- Easy to compute  $x \mapsto g^x$
- Inverse, *discrete logarithm*, hard for large  $p$

### Diffie-Hellman key exchange

- Goal: anonymous key exchange
- Public parameters  $p, g$ ; Alice and Bob have resp. secrets  $a, b$
- Alice  $\rightarrow$  Bob:  $A = g^a \pmod{p}$
- Bob  $\rightarrow$  Alice:  $B = g^b \pmod{p}$
- Alice computes  $B^a = g^{ba} = k$
- Bob computes  $A^b = g^{ab} = k$

### Relationship to a hard problem

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

### Categorizing assumptions

- Math assumptions unavoidable, but can categorize
- E.g., build more complex scheme, shows it's "as secure" as DH because it has the same underlying assumption
- Commonly "decisional" (DDH) and "computational" (CDH) variants

## Key size, elliptic curves

- Need key sizes ~10 times larger than security level
  - Attacks shown up to about 768 bits
- Elliptic curves: objects from higher math with analogous group structure
  - (Only tenuously connected to ellipses)
- Elliptic curve algorithms have smaller keys, about  $2\times$  security level

## Outline

- Public-key crypto basics
- Public key encryption and signatures
- Brief introduction to networking
- Some classic network attacks

## 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

Public-key crypto basics

Public key encryption and signatures

Brief introduction to networking

Some classic network attacks

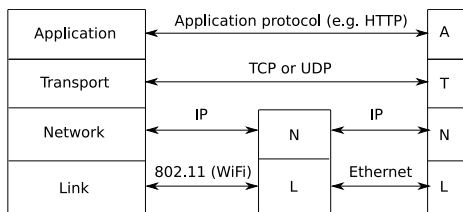
## The Internet

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

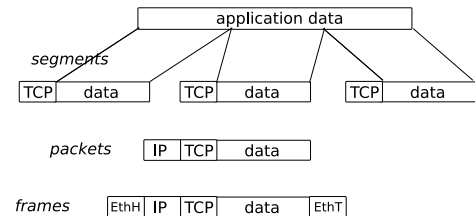
## Layered model (OSI)

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

Public-key crypto basics

Public key encryption and signatures

Brief introduction to networking

Some classic network attacks

## 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