CSE/EE 461 Lecture 23
QoS Wrapup; Security

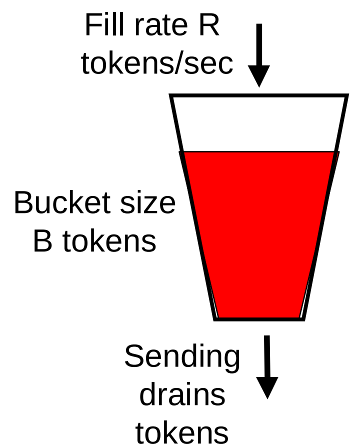
Tom Anderson
tom@cs.washington.edu
Peterson, Chapter 8

Supporting QOS Guarantees

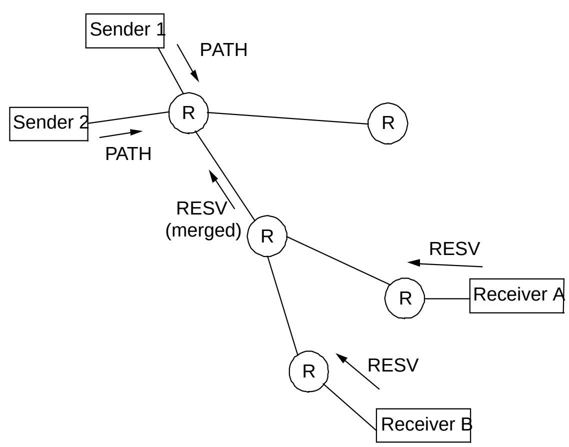
- Flowspecs. Formulate application needs
  - Need descriptor (token bucket) for guarantee
- Admission Control. Decide whether to support a new guarantee
  - Network must be able to control load to provide guarantees
- Signaling. Reserve network resources at routers
  - Analogous to connection setup/teardown, for router reservations
- Packet Scheduling. Implement guarantees
  - Various mechanisms can be used, e.g., explicit schedule, priorities, WFQ, …

Token Buckets

- Simple model
  - reflects both average, variability over time
- Use tokens to send bits
- Avg bandwidth is R bps
- Maximum burst is B bits



Resource Reservation Protocol (RSVP)



RSVP Issues

- RSVP is receiver-driven to be able to support multicast applications
- Only reserve resources at a router if there are sufficient resources along the entire path
  - both for average bandwidth and maximum bursts
- What if there are link failures and the route changes?
  - receivers periodically refresh by sending new requests toward sender
- What if there are sender/receiver failures?
  - reservations are periodically timed out

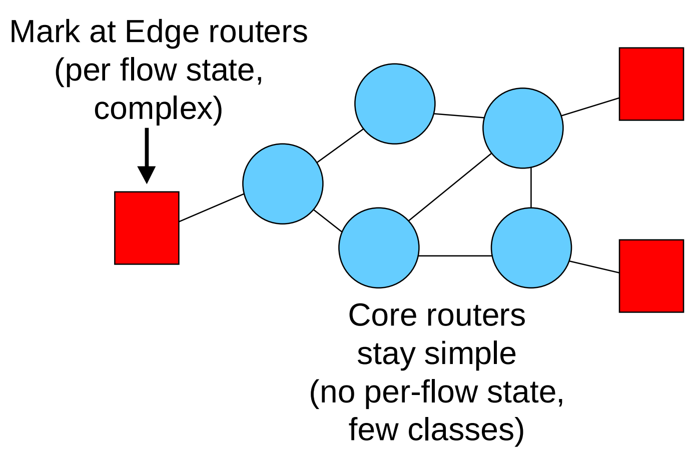
IETF Integrated Services

- Fine-grained (per flow) guarantees
  - Guaranteed service (bandwidth and bounded delay)
  - Controlled load (bandwidth but variable delay)
- RSVP used to reserve resources at routers
  - Receiver-based signaling that handles failures
  - Router can police that flow obeys reservation
- Priorities, WFQ used to implement guarantees
  - Router classifies packets into a flow as they arrive
  - Packets are scheduled using the flow’s resources
  - Flows with guaranteed service scheduled before controlled load, scheduled before best effort

IETF Differentiated Services

- A coarse-grained approach to QOS
  - Packets are marked as belonging to a small set of services, e.g, premium or best-effort, using the TOS bits in the IP header
- Marking policed at administrative boundaries
  - ISP marks 10Mbps (say) of your traffic as premium depending on your service level agreement (SLAs)
- Routers understand only the different service classes, not individual reservations
  - Use priority queues or WFQ for each class, not for each flow

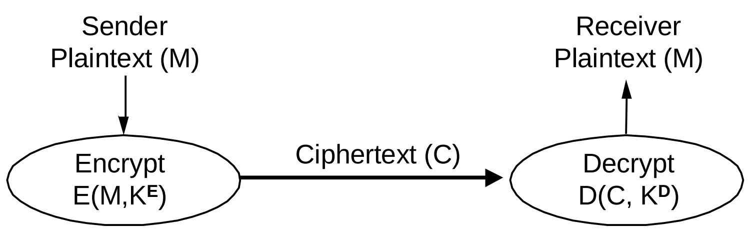
Two-Tiered Architecture



Security

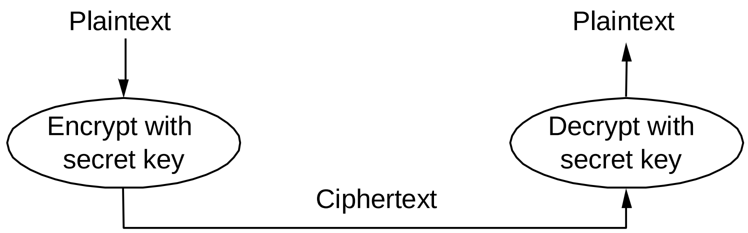
- Networks are shared
  - each packet traverses many devices on path from source to receiver
  - how do you know messages aren't copied, replaced/spoofed, modified in flight, …
- Security Goals
  - Privacy: messages can't be eavesdropped
  - Authentication: messages were sent by the right party
  - Integrity: messages can't be tampered with

Encryption



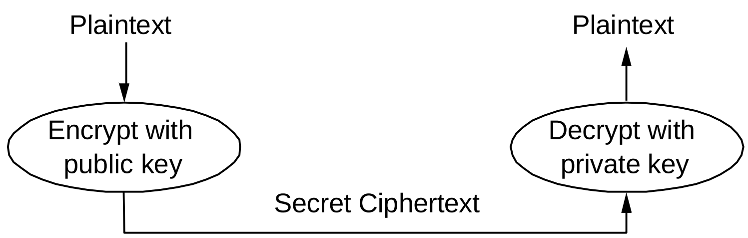
- Cryptographer chooses functions E, D and keys K^E, K^D
  - Suppose everything is known (E, D, M and C), should not be able to determine keys K^E, K^D and/or modify msg
  - provides basis for authentication, privacy and integrity

Secret Key (DES, IDEA)



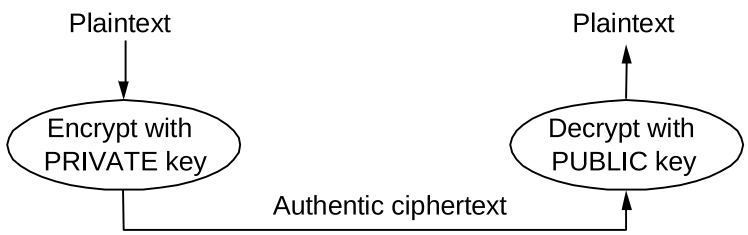
- Single key (symmetric) is shared between parties, kept secret from everyone else
  - Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
  - if K kept secret, then both parties know M is authentic and secret

Public Key (RSA, PGP)



- Keys come in pairs, public and private
  - Each entity (user, host, router,…) gets its own pair
  - Public key can be published; private is secret to entity
    - can't derive K-private from K-public, even given M, (M)^K-priv
  - Ciphertext = (M)^K-public; M = ((M)^K-public)^K-private
  - Ensures privacy: can only be read by receiver

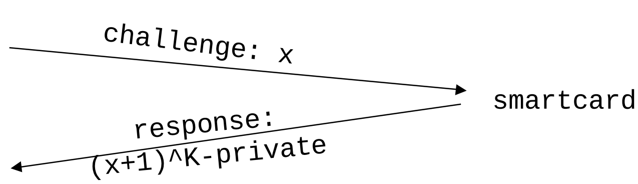
Public Key: Authentication



- Keys come in pairs, public and private
  - M = ((M)^K-private)^K-public
  - Ensures authentication: can only be sent by sender
  - Get both authentication and secrecy, by encrypting in private key of sender, public key of receiver

Public Keys and Smart Cards

- Can be difficult for people to remember encryption keys
  - keys that are easy to remember, are easier to break
  - keys that aren't easy to break, can't be remembered!
- Instead, store K-private inside a chip
  - use challenge-response to authenticate smart card



a

Public Key -> Session Key

- Public key encryption/decryption is slow; so can use public key to establish (shared) session key
  - assume both sides know each other's public key

```
client                                    server
              client ID, x
    ------------------------------------->

          ((K,y,x+1)^C-public)^S-priv
client   <------------------------------------
authenticates
server
                  (y+1)^K
    ------------------------------------->
                                          server
                                          authenticates
                                          client
```

Public Key Distribution

- How do we know public key of other side?
  - infeasible for every host to know everyone's key
  - need public key infrastructure (PKI)
- Certificates (X.509)
  - Distribute keys by trusted *certificate authority* (CA)
    - "I swear X's public key is Y", signed by CA (their private key)
  - Example CA's: Verisign, Microsoft, UW CS Dept., …
- How do we know public key of CA?
  - Can build chains of trust, e.g., given public key of UW CS's CA, who can sign for Verisign's public key, who can sign for xyz's public key

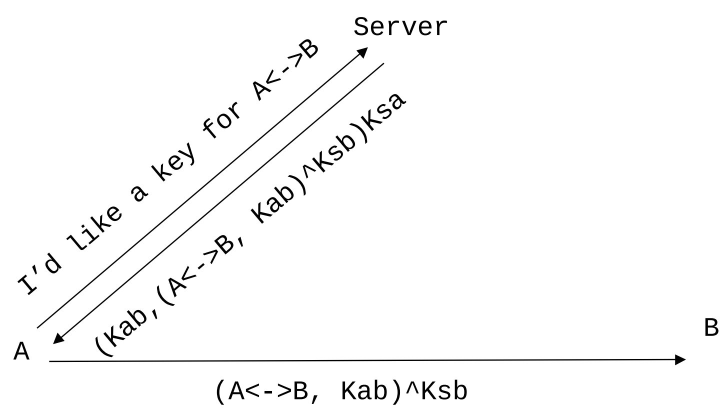
Public Key Revocation

- What if a private key is compromised?
  - need certificate revocation list (CRL)
    - and a CRL authority for serving the list
  - everyone using a certificate is responsible for checking to see if it is on CRL
  - ex: certificate can have two timestamps
    - one long term, when certificate times out
    - one short term, when CRL must be checked
    - CRL is online, CA can be offline

Shared Key -> Session Key

- In shared key systems, how do we gain a shared key with other side?
  - infeasible for everyone to share a secret with everyone else
  - solution: “authentication server” (Kerberos)
    - everyone shares (a separate) secret with server
    - server provides shared session key for A <-> B
  - everyone trusts authentication server
    - if compromise server, can do anything!

Kerberos Example

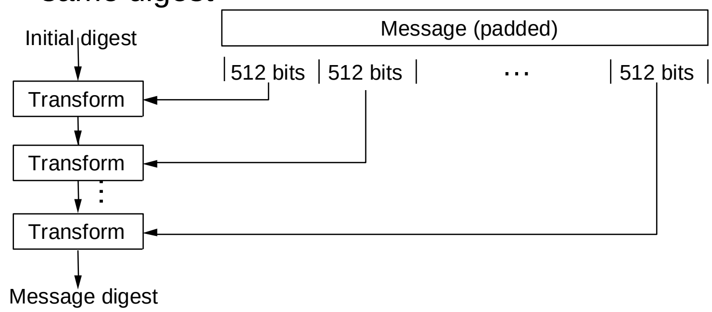


Kerberos Details

- Any key can be broken if given a long enough time
  - Use timestamps to ensure that keys were created recently
- Need to ensure attacker doesn't change messages in flight
  - ex: replace parts of message
  - use encrypted checksum on entire message
- Passwords are often easily broken
  - Derive Ksa from A's password
  - Use Ksa to establish temporary key, Ksa-temp

Message Digests (MD5, SHA)

- Cryptographic checksum: message integrity
  - Typically small compared to message (MD5 128 bits)
  - “One-way”: infeasible to find two messages with same digest



Example Systems

- Cryptography can be applied at multiple layers
- Pretty Good Privacy (PGP)
  - For authentic and confidential email
- Secure Sockets (SSL) and Secure HTTP (HTTPS)
  - For secure Web transactions
- IP Security (IPSEC)
  - Framework for encrypting/authenticating IP packets

PGP

- Application level system
- Based on public keys and a “grass roots” Web of trust
- Sign messages for integrity/authenticity
  - Encrypt with private key of sender
- Encrypt messages for privacy
  - Could just use public key of receiver …
  - But encrypt message with secret key, and secret key with public key of receiver to boost performance

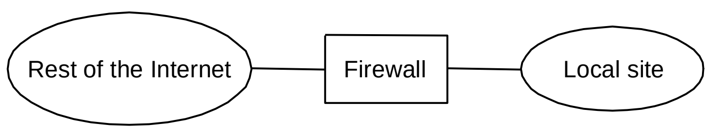
SSL/TLS and HTTPS

- Secure transport layer targeted at Web transactions
  - SSL/TLS inserted between TCP and HTTP to make secure HTTP
- Extra handshake phase to authenticate and exchange shared session keys
  - Client might authenticate Web server but not vice-versa
    - Certificate Authority embedded in Web browser
- Performance optimization
  - Refer to shared state with session id
  - Can use same parameters across connections
    - Client sends session id, allowing server to skip handshake

IPSEC

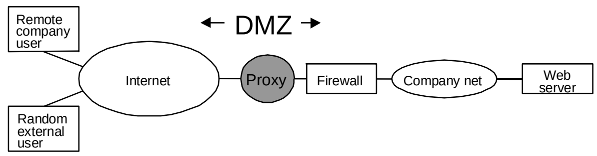
- Framework for encrypted IP packets
  - Choice of algorithms not specified
- Uses new protocol headers inside IPv4 packets
  - Authentication header
    - For message integrity and origin authenticity
    - Optionally “anti-replay” protection (via sequence number)
  - Encapsulating Security Payload
    - Adds encryption for privacy
- Depends on key distribution (ISAKAMP)
  - Sets up security associations
- Ex: secure tunnels between corporate offices

Filter-based Firewalls



- Sit between site and rest of Internet, filter packets
  - Enforce site policy in a manageable way
  - e.g. pass (*,*, 128.7.6.5, 80), then drop (*, *, *, 80)
  - Rules may be added dynamically to pass new connections
- Sometimes bundled with a router: “level 4” switch
  - Acts like a router (accepts and forwards packets)
  - Looks at information up to TCP port numbers (layer 4)

Proxy-Based Firewalls



- Problem: Filter ruleset can be complex/insufficient
  - Adequate filtering may require application knowledge
  - Example: email virus signature
- Run proxies for Web, mail, etc. just outside firewall
  - External requests go to proxies, only proxies connect inside
    - – External user may or may not know this is happening
  - Proxies filter based on application semantics