Security and Cryptography

CSE 461

Ben Greenstein
Jeremy Elson

TA: Ivan Beschastnikh

Administrivia

- Project 3, part 2 due December 5
- Special extended office hours: Tuesday, December 2, 11:30-1:30 Room 218
- No HW this week

Security in Practice

- Attackers have the advantage
  - Get to think outside the box
  - Can exploit any unanticipated weakness
  - Obscurity hard to maintain

- Defense
  - Needs to anticipate all feasible attack vectors
  - Hard to prove that no attack is possible
    - Even at the crypto level
  - Hard to detect if an attack has been successful
  - Hard to re-secure a system after an attack

Fundamental Tenet: *If lots of smart people have failed to break a system, then it probably won't be broken*

To Publish or Not to Publish

- If the good guys break your system, you'll hear about it
- If you publish your system, the white hats provide free consulting by trying to crack it
- The black hats will learn about your system anyway
- Today, most (but not all) commercial systems are published; most military systems are not

To Publish or Not to Publish (Part 2)

- If you discover a workable attack, what is your responsibility?
- Gap between discovery of vulnerability, and exploiting the vulnerability can be seconds
- Should notify vendor and publish

Some Old Examples

- Western Digital
  - Compromise went undetected for months
- Thompson self-propagating back door login
  - Reinstalls itself in every new version of UNIX
- Tiger team attempt on Pentagon computer
  - No physical access
- Secure communications channel: one time pad
  - paper tape of random #'s
  - same tape used at sender, receiver
  - system XORs to each bit before xmit/receive

Some Recent Examples

- House Keys
- ATM keypad
- Pacemakers
- Mifare transit smart cards
- Washington State Driver's Licenses (EPC RFID)
- Electronic car keys
- Elevator controls
- Voting machines
- WEP

Network Security

- Networks are shared
  - each packet traverses many devices on path from source to receiver
- Attacker might be in control of any of these devices
  - Or other machines on the network
  - Or administrative machines
  - Or, …

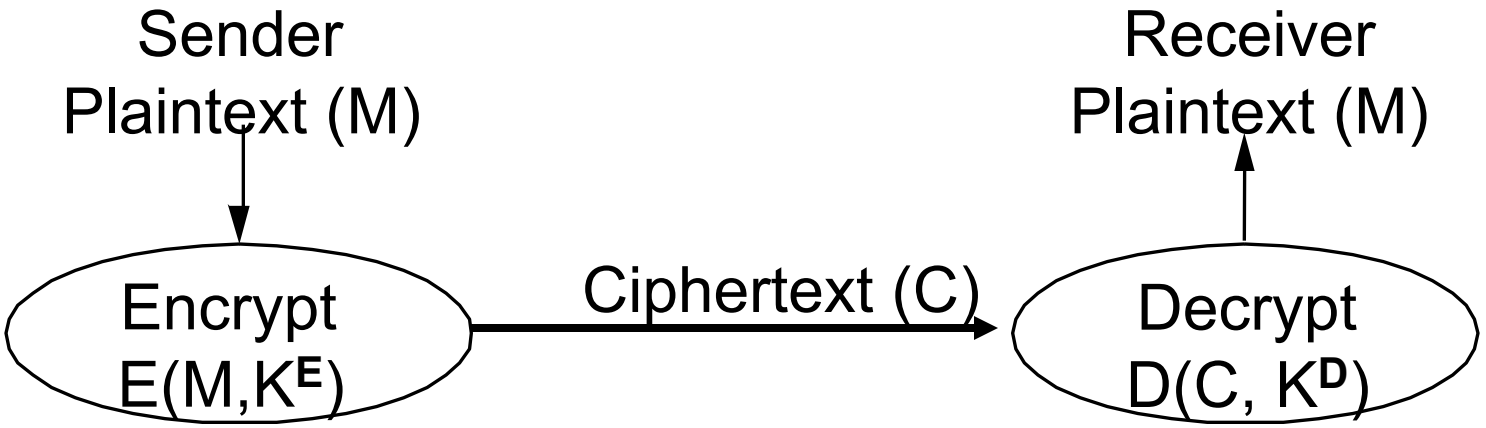
Network Security

- How do you know messages aren't:
  - Copied
  - Injected
  - Replaced/modified
  - Spoofed
  - Inferred
  - Prevented from being delivered
  - …

Security Threats, Goals in ()'s

- Impersonation (Authentication)
  - Pretend to be someone else to gain access to information or services
- Lack of secrecy (Privacy)
  - Eavesdrop on data over network
- Corruption (Integrity)
  - Modify data over network
- Denial of Service (Message Delivery)
  - Flood resource to deny use from legitimate users

Encryption



- Cryptographer chooses 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 C without detection
  - provides basis for authentication, privacy and integrity

How Secure is Encryption?

- An attacker who knows the algorithm we're using could try all possible keys
- Security of cryptography depends on the limited computational power of the attacker
- A fairly small key (e.g. 128 bits) represents a formidable challenge to the attacker
- Algorithms can also have weaknesses, independent of key size

How Practical is Encryption

- Usability depends on being efficient for the good guys
- Cost to the good guys tends to rise linearly with key length
- Cost to search all keys rises exponentially with key length
- How do we keep keys secret?
  - Short keys: easy to remember, easy to break

How Secure are Passwords?

- UNIX passwords: time to check all 5 letter passwords (lower case): 26^5 ~ 10M
  - in 75, 1 day
  - in 92, 10 seconds
  - In 08, 0.001 seconds
- Extend password to six letters, require upper, lower, number, control char: 70^6 ~ 600B
  - in 92, 6 days
  - in 08, with 1000 PC's in parallel, < 1 second (!)

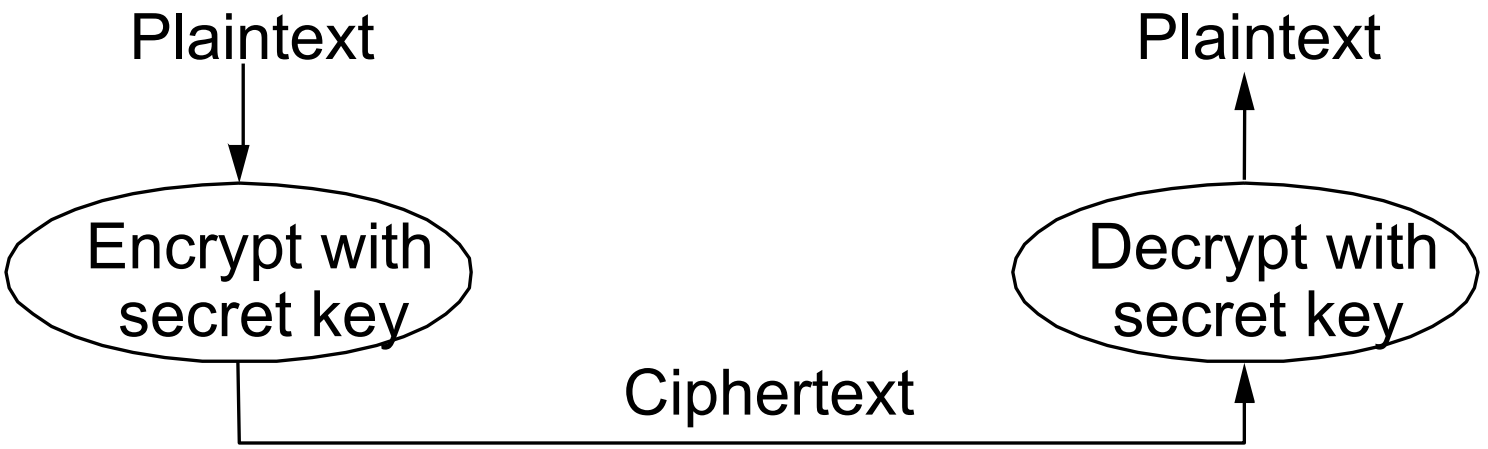
Password Attack/Response

- Moore's Law: enables large number of passwords to be checked very quickly
- Countermeasure
  - Delay password check for 1 second, so can't try them quickly
  - Need to delay both successful and unsuccessful password checks!
- Counter-countermeasure:
  - Observe network traffic; extract any packet encrypted in password; check various passwords offline
- Counter-counter-countermeasure:
  - Kerberos: don't use password to encrypt packets; instead use password to encrypt file containing shared key; use shared key to encrypt packets
- Counter-counter-counter-countermeasure: …

Cryptography

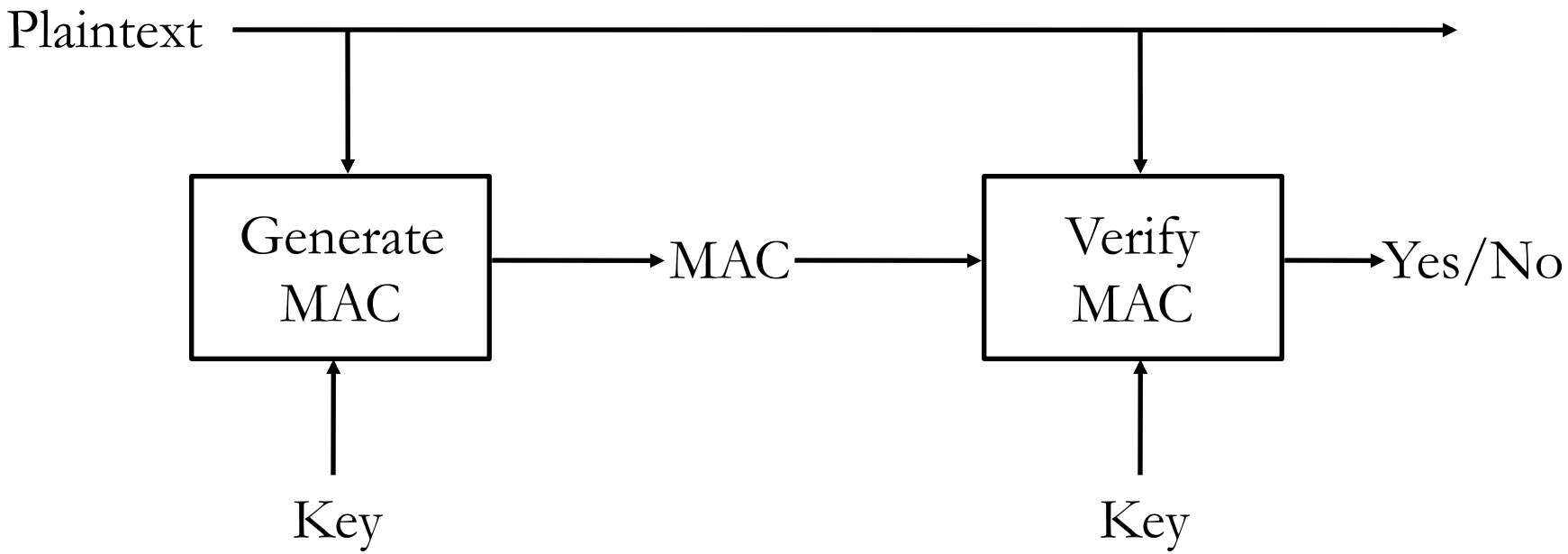
- Secret Key Cryptography (DES, IDEA, RCx, AES)
- Public Key Cryptography (RSA, Diffie-Hellman, DSS)
- Message Digests (MD4, MD5, SHA-1)

Secret Key

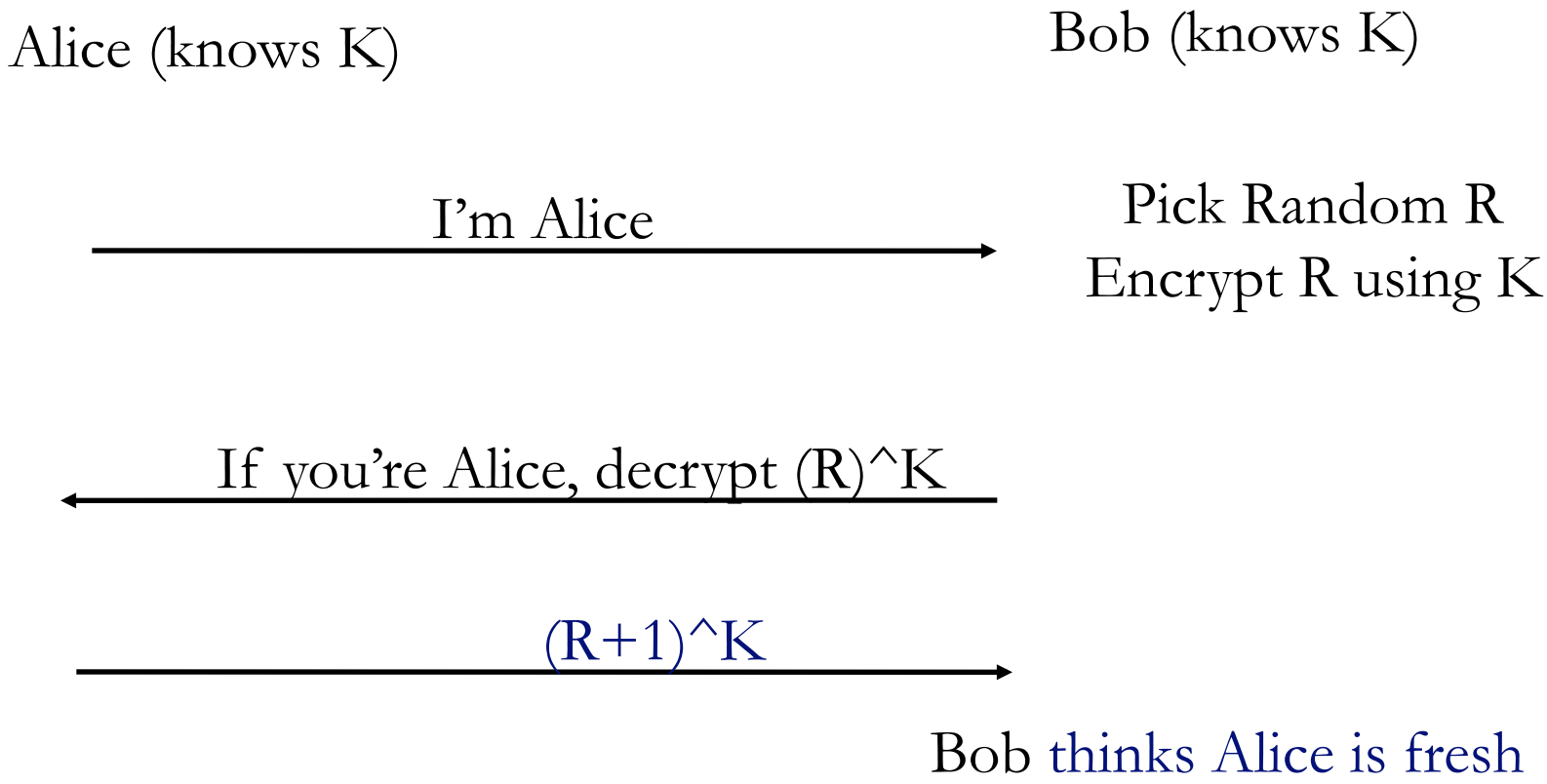


- Single key (symmetric) is shared between parties, kept secret from everyone else
  - Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
  - if K kept secret, then both parties know M is authentic and secret

Secret Key Integrity: Message Authentication Codes



Challenge / Response Authentication



Secret Key Algorithms

- DES (Data Encryption Standard) – 1970's IBM, NSA?
  - 56 bit key (+ 8 parity bits) => has become too small
  - Input and output are 64 bit blocks
  - slow in software, based on (gratuitous?) bit twiddling

Other Ciphers

- Triple-DES
  - DES three times
    - $m_c = E(D(E(m_p, k_1), k_2, k_3)$
  - Effectively 112 bits
  - Three times as slow as DES

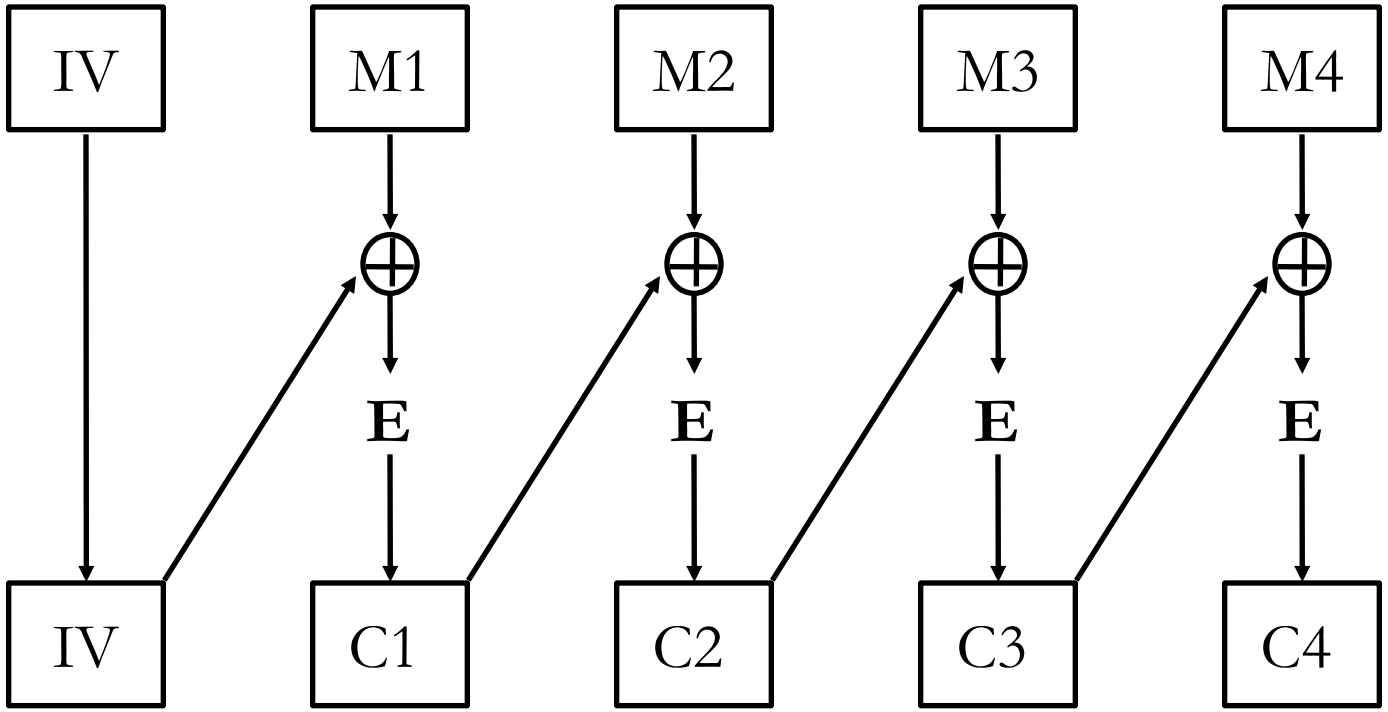
- Blowfish
  - Developed by Bruce Schneier circa 1993
  - Variable key size from 32 to 448 bits
  - Very fast on large general purpose CPUs (modern PCs)
  - Not very easy to implement in small hardware

- Advanced Encryption Standard (AES)
  - Selected by NIST as replacement for DES in 2001
  - Uses the Rijndael algorithm
  - Keys of 128, 192 or 256 bits

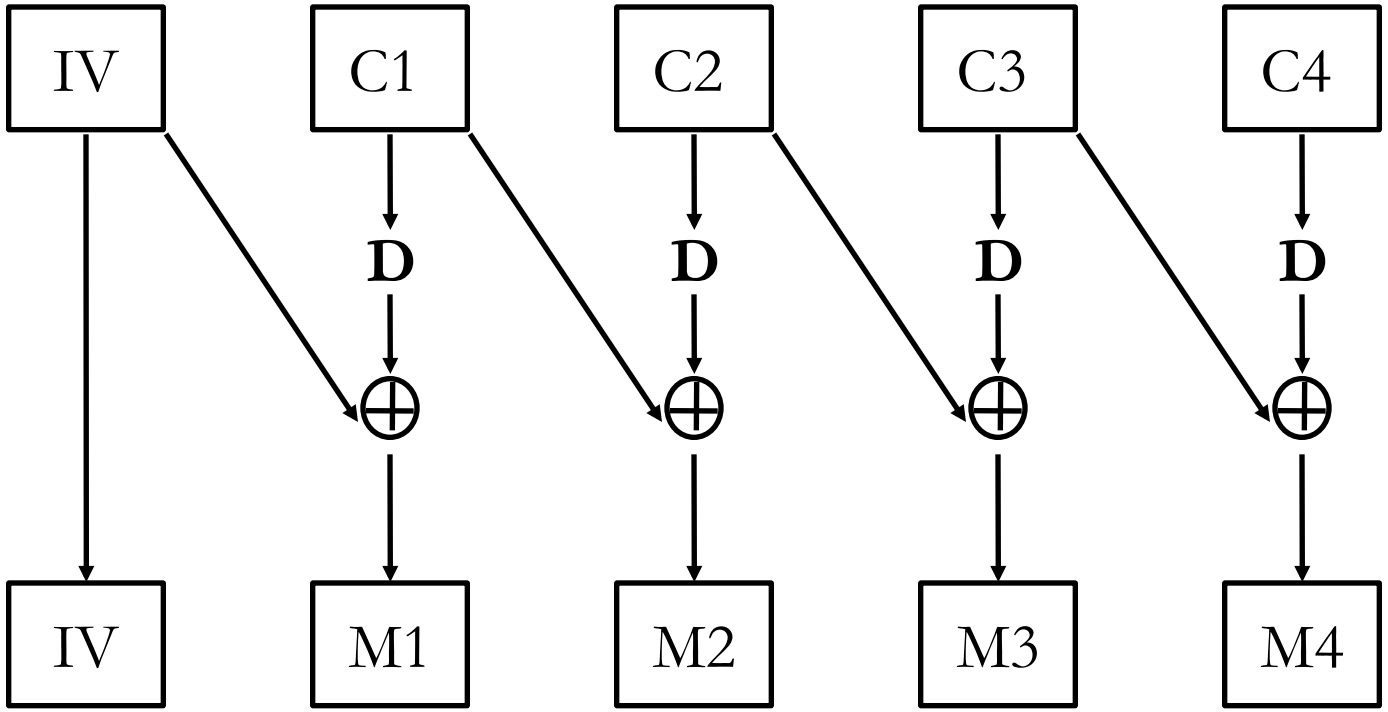
Encrypting Large Messages

- 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)
  - Leaks data: repeated plaintext blocks yield repeated ciphertext blocks
  - Does not guarantee integrity!
- Other modes "chain" to avoid this (CBC, CFB, OFB)

CBC (Cipher Block Chaining)



CBC Decryption



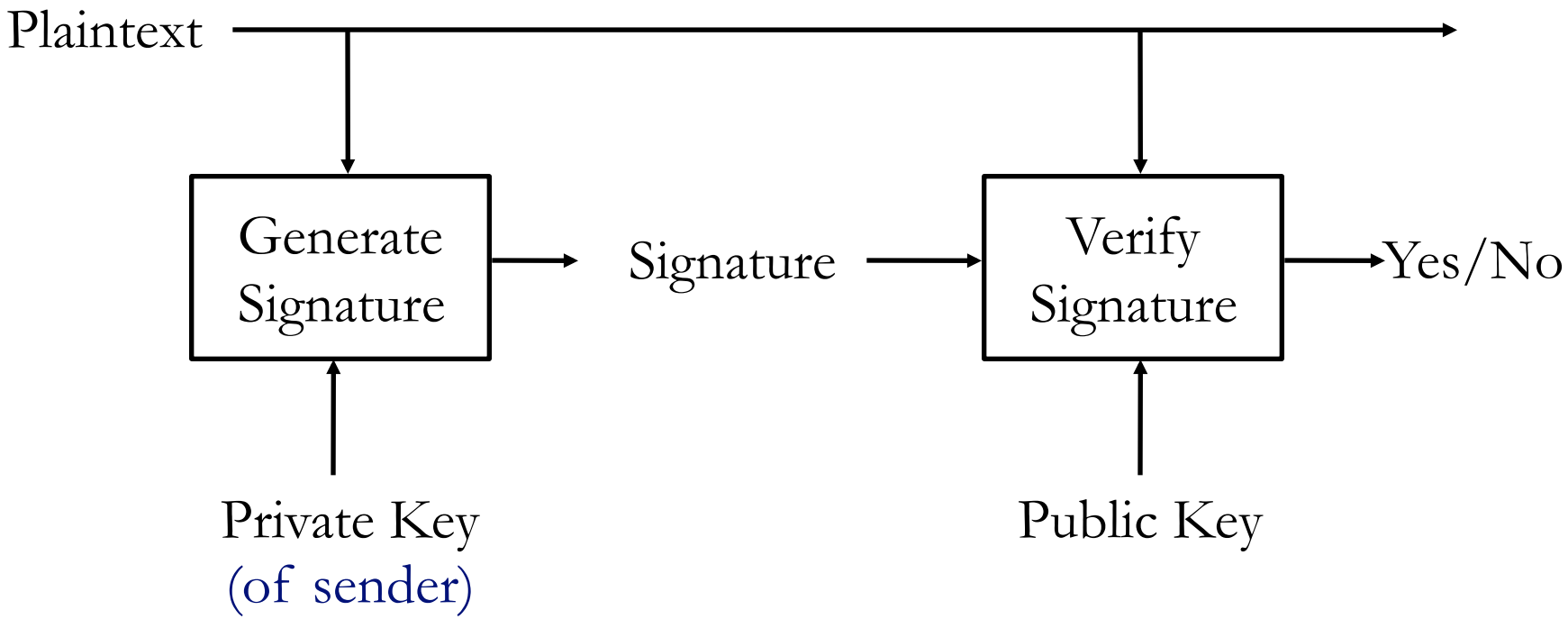
XOR (Exclusive-OR)

- Bitwise operation with two inputs where the output bit is 1 if exactly one of the two input bits is one
- (B XOR A) 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

Public Key Encryption

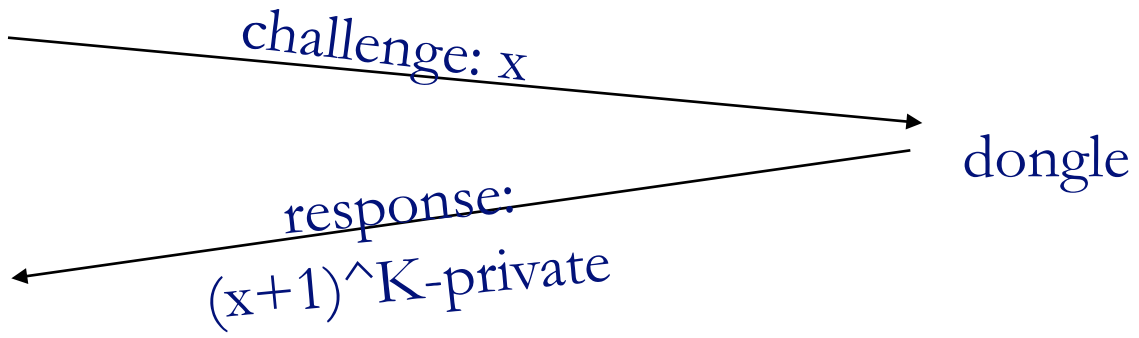
- Keys come in pairs, public and private
  - Each entity (user, host, router,…) gets its own pair
  - Public key can be published; private is secret to entity
    - can't derive K-private from K-public, even given M, (M)^K-priv
  - If encrypt with receiver's public key, ensures can only be read by receiver

Public Key Integrity Protection



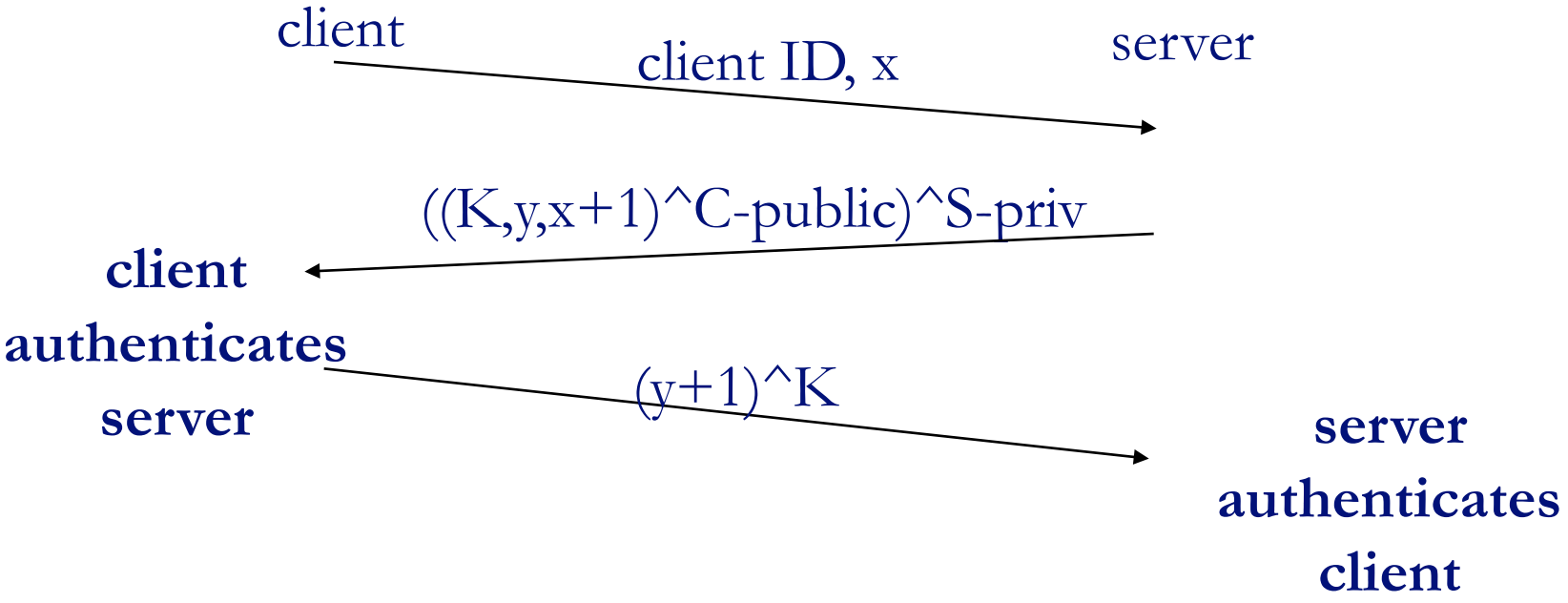
Zero Knowledge Authentication

- Where to keep your private key?
  - keys that are easy to remember, are easier to break
  - keys that aren't easy to break, can't be remembered!
  - If stored online, can be captured
- Instead, store private key inside a chip
  - use challenge-response to authenticate user



Public Key -> Session Key

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



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., …
  - But! Doesn't mean entity is trustworthy!
- How do we know public key of CA?
  - Typically, hard-coded into browsers
  - Alternative: build chain of trust, e.g., from UW's CA to list of CA's that UW trusts

Public Key Revocation

- What if a private key is compromised?
  - Hope it never happens?

- 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

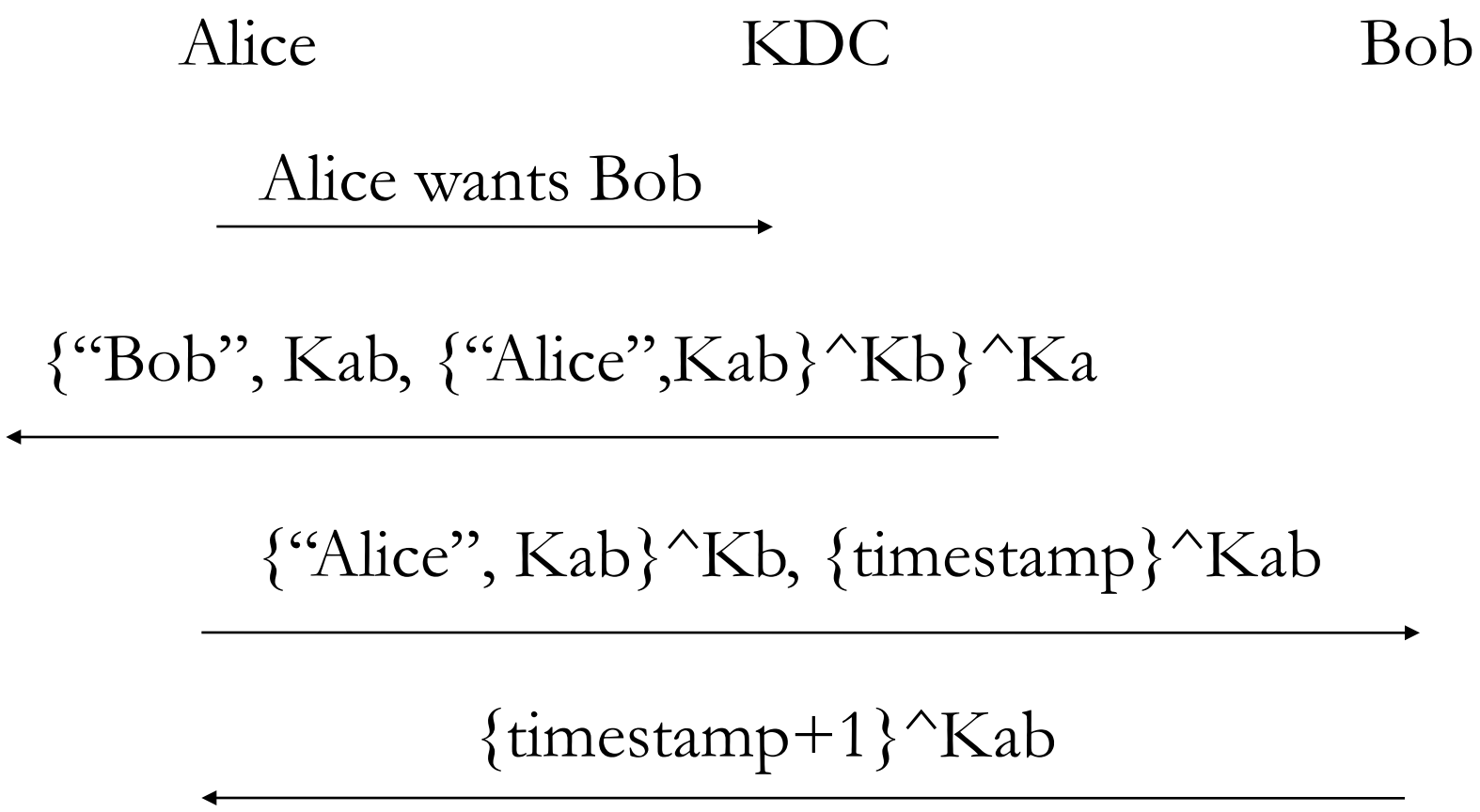
Secret Key -> Session Key

- In secret key systems, how do we get a secret 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 session key for A <-> B
  - everyone trusts authentication server
    - if compromise server, can do anything!

Kerberos

- Developed at MIT
- Based on secret key cryptography
- Code is publicly available (for a long time not legally exportable from the U.S.)
- Early version used block cipher
  - Vulnerability caught and fixed
- Embedded in a variety of commercial products
  - Ex: in use by UW CSE

Kerberos Authentication (Basic)



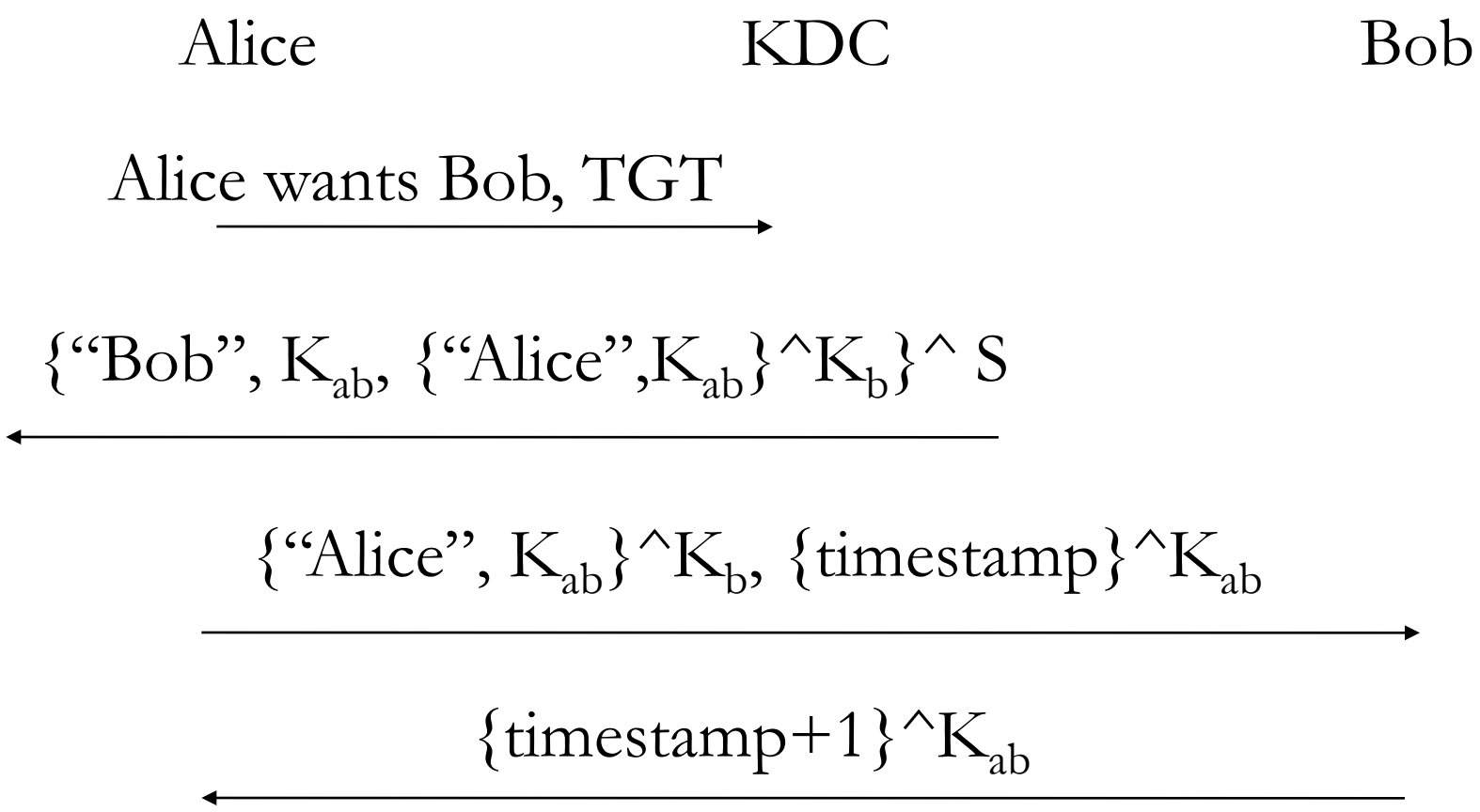
Ticket Granting Tickets

- It is dangerous for the workstation to hold Alice's secret for her entire login session
- Instead, Alice uses her password to get a short lived "ticket" to the "Ticket Granting Service" which can be used to get tickets for a limited time
- For a login session >8 hours, she must enter her password again

Ticket Granting Tickets

- TGT looks just like ticket but encrypted with KDC's key
- WS keeps TGT = {"Alice",S}K_{kdc} and S

Kerberos Authentication (with TGT={"Alice",S}K_{kdc})



Pre-authentication

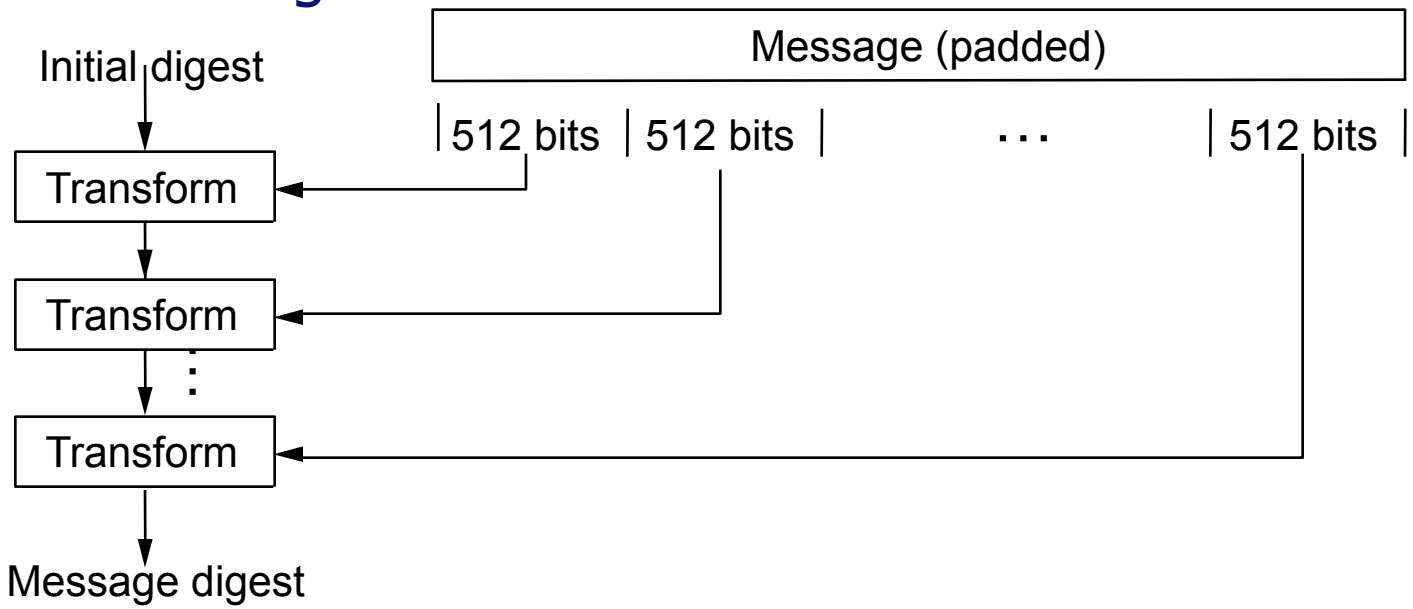
- Anyone can request a ticket on behalf of Alice, and the response will be encrypted under her password
- This allows an off-line password guessing attack
- Kerberos V5 requires an encrypted timestamp on the request
  - Only an eavesdropper can guess passwords

Kerberos Weaknesses

- Early versions of Kerberos had several security flaws
  - block cipher: allows encrypted blocks to be replaced
    - solution: add encrypted CRC over entire message
  - uses timestamps to verify communication was recent
    - time server communication not encrypted (!)
    - get time from authentication server
  - Kerberos login program downloaded over NFS
    - NFS authenticates requests, but data is unencrypted
    - disallow diskless operation?

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



Comparative Performances

- According to Peterson and Davie
- MD5: 600 Mbps
- DES: 100 Mbps
- RSA: 0.1 Mbps

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

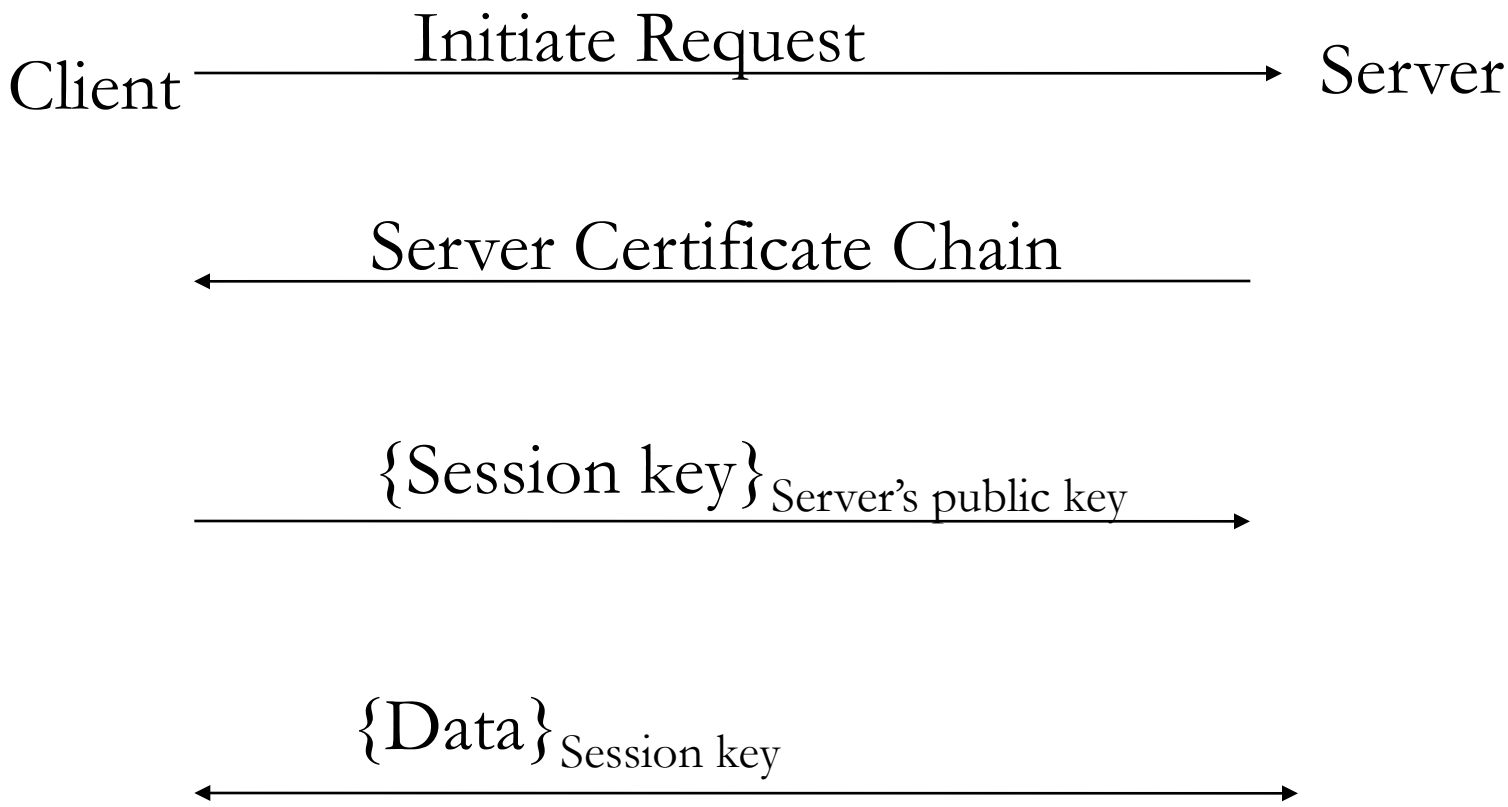
PGP

- 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

SSL/TLS


Client
Initiate Request
Server
Server Certificate Chain
{Session key}Server's public key
{Data}Session key

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

Summary

- Security goals: Authenticity, Integrity, Privacy
- Public key crypto slow, good for signing
- Secret (symmetric) key faster, e.g., AES
- Important security practices: IPSEC, TLS/SSL, PGP, 802.11i