SECURITY BASICS

Internet Protocols

CSC / ECE 573

Fall, 2005

N. C. State University

Announcements

I. *Project progress?*

Today's Lecture

I. Security Basics
II. Encryption Algorithms
III. Digital Signatures and Message Digests
IV. Certificates
V. Authentication Protocols

SOME CRYPTOGRAPHIC PRIMITIVES

Types of Ciphers

- Substitution ciphers
  - substitute one string or character for another
- Transposition cipher
  - scramble sequence of letters
- Ciphers based on sequences of transpositions (permutations) and substitutions of bits are very common

Types of Ciphers (cont'd)

- One-time pad
  - Generate random bit string, same length as plaintext
  - XOR plaintext with random bit string
  - provably secure

```
'N'=78= 0100 1110            'Y'=89= 0101 1001
      ⊕ 1101 1101 OPD              ⊕ 1100 1010 OPD
        1001 0011                    1001 0011
```

can't tell encrypted 'N' from 'Y'

Replay Attacks

- There are lots of situations where message contents should be processed only once
  - attackers will attempt to store and *replay* the message
- *Nonce* = integer value introduced into a message to demonstrate its "freshness"
  - can also be used as a *challenge* (value to be encrypted)

Nonces

- Ways to generate
  - sequence of integer values (sequence number)
  - read the clock at the sending machine (timestamp)
  - combination of both is best
- Used only once, and generated on demand
  - can tell if received previously (i.e., allows detection of replay attacks)
  - also allows bounding the lifetime of authentication information

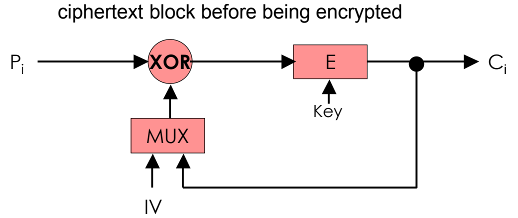
ENCRYPTION ALGORITHMS

Cipher Modes

- All ciphers work on blocks of data (i.e., data is "chunk"ed before processing)
- *Problem*: if a plaintext block appears twice in the input, same output ciphertext will appear twice
  - what's the harm in that?

Cipher Mode Approaches

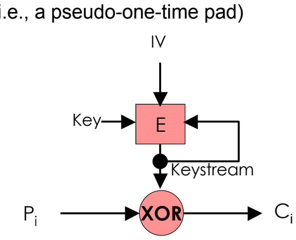
- *Solution 1*: Cipher Block Chaining (CBC)
  - each plaintext block is XOR'ed with previous ciphertext block before being encrypted



initialization vector, used for the first block only

Cipher Mode Approaches (cont'd)

- *Solution 2*: Stream Cipher Mode
  - initialization vector (IV) used to generate a sequence of output blocks
  - these blocks are XORed with plaintext to get ciphertext (i.e., a pseudo-one-time pad)



Symmetric Key Encryption

- Both parties (A and B) must share a single, *secret key*
  - exchange of this secret key must be done over secure (trusted) communications channel
  - a compromised key breaks the scheme
- The encryption and decryption functions can be identical, since the key used is secret

Symmetric Key Encryption (cont'd)

- Most important examples
  - DES – (1977), 56 bit key, not hard to break
  - Triple DES – (1979), 112 bit key, relatively strong
  - AES – (2001), 128 bits or 256 bits, very strong, efficient

Asymmetric (*Public-Key*) Encryption Algorithms

- Keys are generated in pairs
  1. public key K_1 (for encryption or decryption) – easily obtained by anyone
  2. private key K_2 (for decryption or encryption) – only known by one party
  3. $D_{K1}(E_{K2}(P)) = D_{K2}(E_{K1}(P)) = P$
- A "well-known" server stores the public keys, provides them on request

Public-Key Encryption Algorithms (cont'd)

- Must be very difficult to determine the private key from the public key
- Important examples
  - RSA – (1978), 1024 bits, very strong, based on difficulty of factoring
  - El Gamal – (1985), based on discrete logarithms
  - Elliptic curves - 1993

Public-Key Applications

- Application #1
  - anybody can encrypt a message for A, using A's public key
  - only A can decrypt these messages
- Application #2
  - only A can encrypt messages using A's private key
  - anybody can decrypt these messages, using A's public key

Comparison of Types of Cryptography

- Public key…
  - more general
  - uses stronger cryptography
  - provides stronger non-repudiation
- Shared key…
  - simpler, cheaper
  - more robust (less centralized)
  - executes faster
- Hybrid approach: use public key for negotiating, distributing secret keys
  - then use symmetric key encryption thereafter

DIGITAL SIGNATURES, MESSAGE DIGESTS, AND CERTIFICATES

Digital Signatures

- A *digital signature* is a piece of information attached by the creator of a message
- Purposes
  1. verify the claimed originator of a message is the real originator
  2. verify the message has not been subsequently altered by someone else
  3. make sure the message cannot be *repudiated* by the originator

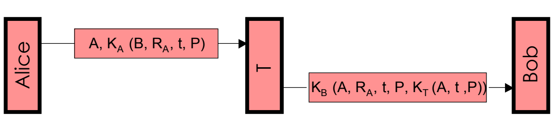
Digital Signatures (cont'd)

- Should be possible for any recipient of the message to verify the signature is valid
- Simplest approach: to each message, append a copy of the message contents, encrypted with the key of the originator
  - encrypted version proves identity of originator, and that message has not been tampered with

Signatures Based on Symmetric Keys

- Uses trusted third party
- Need to include nonce to prevent replay attacks
- Notation
  - A = Alice, B = Bob
  - T = Trusted (3rd party) Server
  - K_A = Encrypt with A's Key
  - R_A = Random # generated by A
  - t = timestamp
  - P = Plaintext

Signatures Based on Symmetric Keys (cont'd)

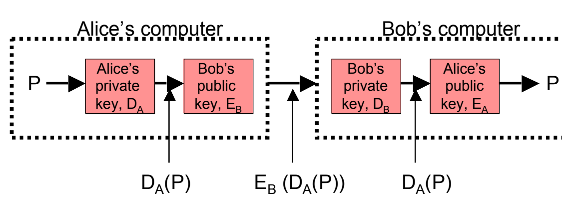


- Assumption: A and T share K_A,

  B and T share K_B, K_T

Signatures Based on Public Keys

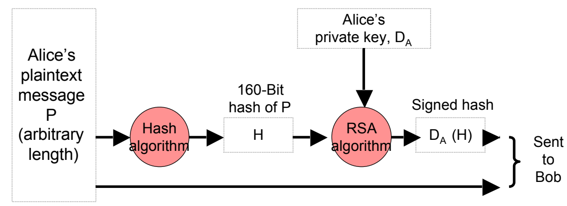
- Eliminates trust in third party, but requires method of distributing public keys



Message Digests

- Drawback of basic signatures = expense of encrypting the entire message
  - improvement: produce a *digest* of the message, encrypt just this digest
  - *digest* = a summary or secure *hash* of a message
  - less overhead (computing and communication)

Message Digest Example



Message Digests

- Desired properties
  1. easy to compute digest from message, but impossible to recover original message from the digest
  2. change of 1 bit of message produces very different digest, and very difficult to find two messages with same digest (*collisions*)

Message Digests (cont'd)

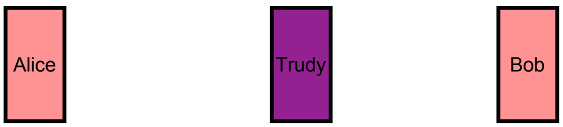
- Algorithms
  - MD5 – (1992), widely used, generates 128-bit digest (RFC 1321), breakable with some effort
  - SHA-1 – (1993), generates 160-bit digest, breakable?
- HMAC (RFC 2104)
  - message authentication code based on a secret key, can be used with any message digest method

CERTIFICATES

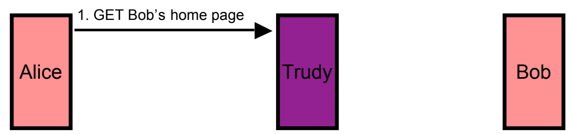
Certificates

- A *certificate* is a binding of key to an identity
  - signed by trusted party (e.g., certificate authority)
- Certificates are the means of learning public keys

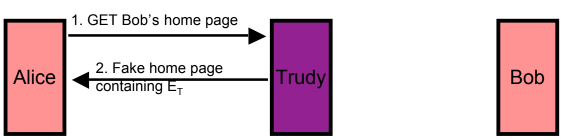
Why Are Certificates Needed?



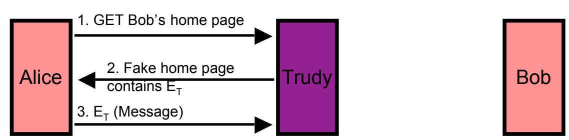
Why Are Certificates Needed?



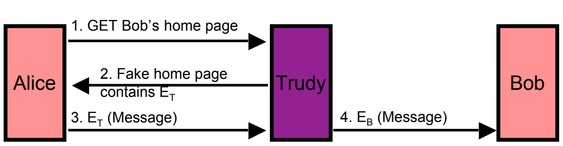
Why Are Certificates Needed?



Why Are Certificates Needed?



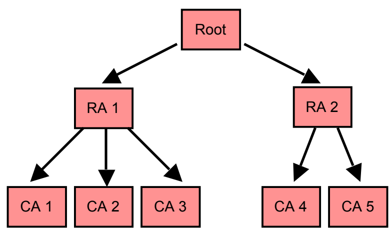
Why Are Certificates Needed?



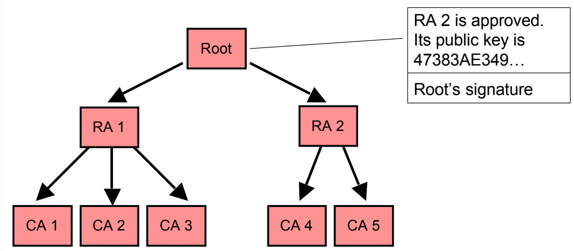
Certificate Hierarchy

- Public key infrastructure (PKI)
  - a hierarchy of certificate authorities
  - each level certifies keys of next level down in the hierarchy
- Basis of trust in certificate hierarchies: root servers
  - many root servers
  - web browsers are preloaded with identity of root servers that can be trusted

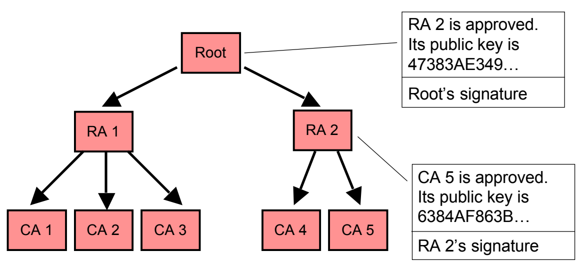
Certificate Hierarchy Example



Certificate Hierarchy Example (cont'd)



Certificate Hierarchy Example (cont'd)



Certificate Revocation

- Keys may change or expire or be compromised
- "Revoking" a certificate is then required
- Example approach: publish *certificate revokation lists (CRLs)*
- Difficult problem, not completely solved

AUTHENTICATION PROTOCOLS

Authentication Protocols

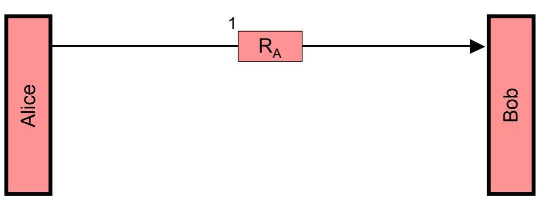
- *Authentication* = verifying identity of someone
- *Authorization* = granting access to resource based on identity

Authentication Based on Shared Secret Key

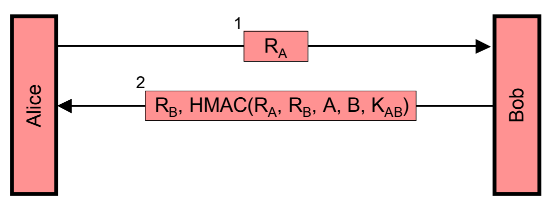
- "Challenge-Response" schemes
  1. send a challenge in encrypted form
  2. wait for expected response, also in encrypted form
- Notation
  - K_{AB} = Shared Key
  - R_A , R_B = random numbers generated by A, B

Authentication Based on Shared Secret Key (cont'd)

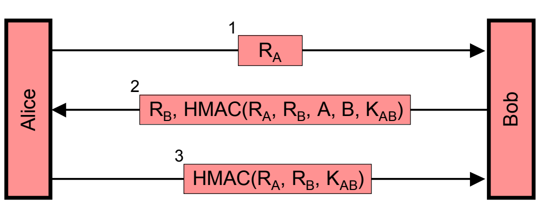
Authentication Based on Shared Secret Key (cont'd)



Authentication Based on Shared Secret Key (cont'd)



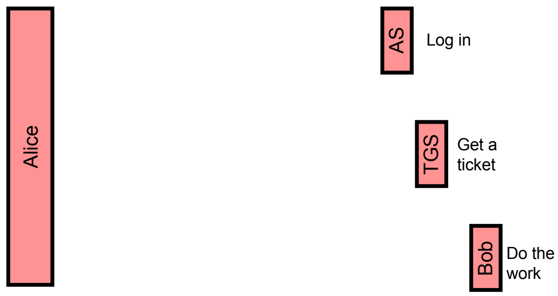
Authentication Based on Shared Secret Key (cont'd)



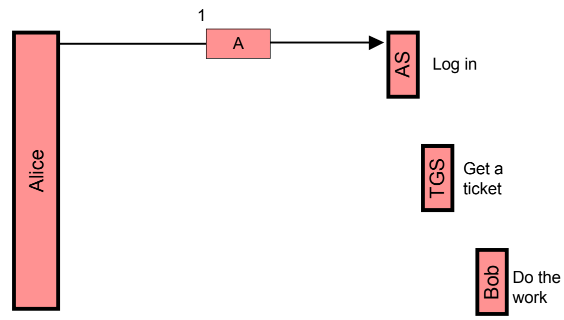
Authentication with Kerberos

- Two servers are needed
  - authentication server (AS) to verify user's identity
  - ticket-granting server (TGS) to issue "proof of identity" certificate
- *Result*: can securely access multiple servers without needing to exchange a password with each one
- Servers then determine what service to provide / allow to user, based on identity
- Password never transmitted across network

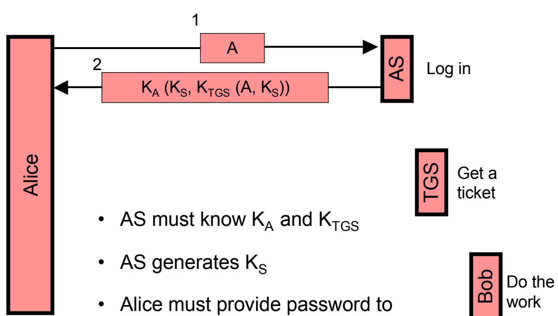
Kerberos Authentication



Kerberos Authentication (cont'd)

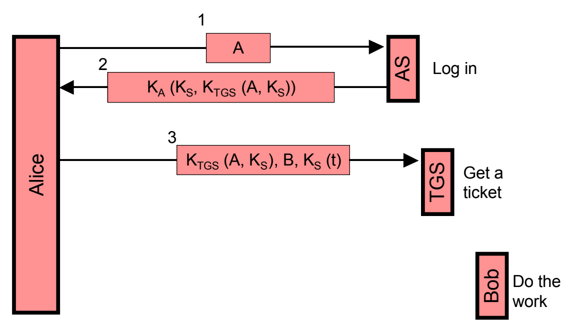


Kerberos Authentication (cont'd)

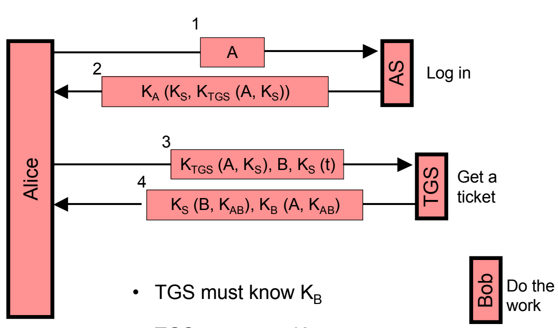


- AS must know K_A and K_{TGS}
- AS generates K_S
- Alice must provide password to decrypt message 2

Kerberos Authentication (cont'd)

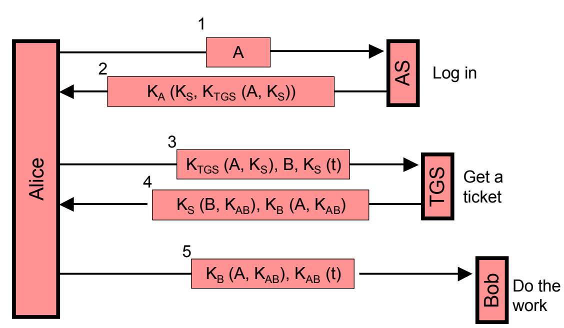


Kerberos Authentication (cont'd)

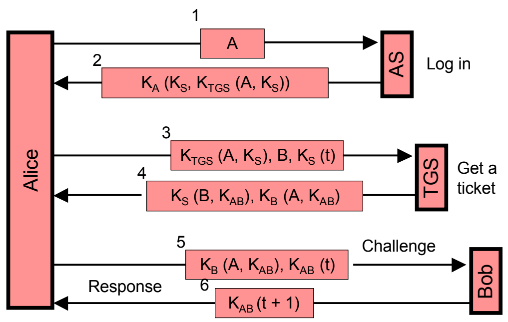


- TGS must know K_B
- TGS generates K_{AB}

Kerberos Authentication (cont'd)



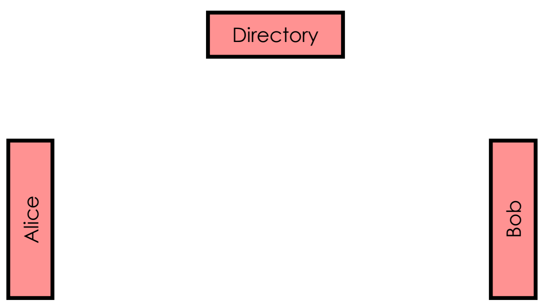
Kerberos Authentication (cont'd)



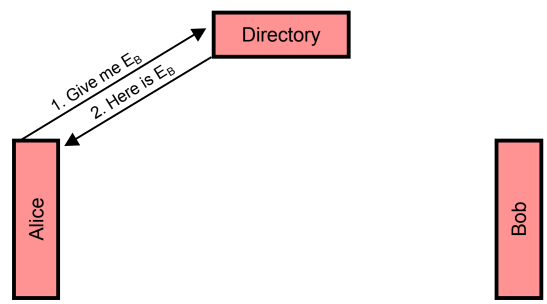
Public-Key Authentication

- Much easier
- Note: communication with directory must be authenticated

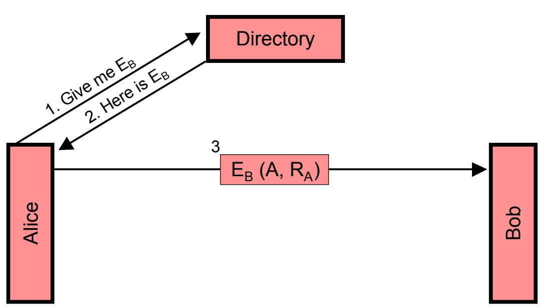
Public-Key Authentication (cont'd)



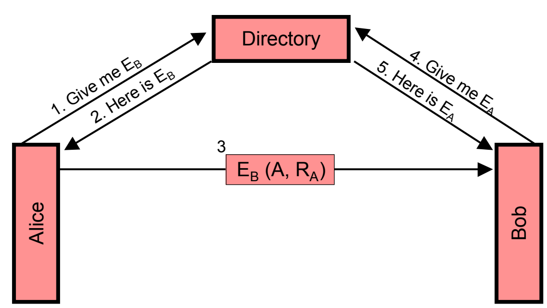
Public-Key Authentication (cont'd)



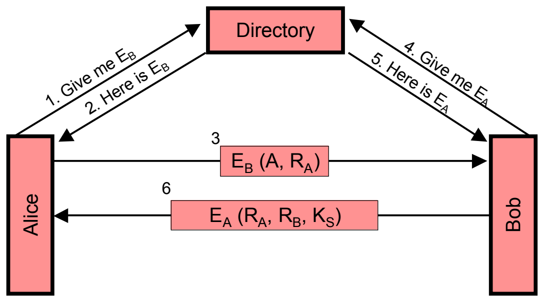
Public-Key Authentication (cont'd)



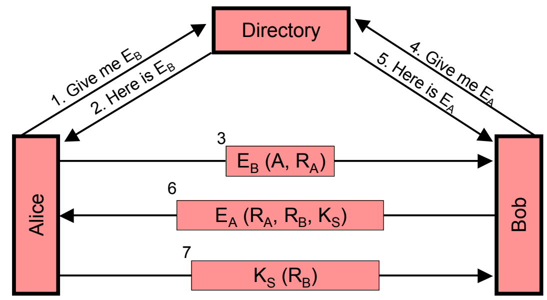
Public-Key Authentication (cont'd)



Public-Key Authentication (cont'd)



Public-Key Authentication (cont'd)



Summary

- Security is a high priority
  - protecting Internet protocols
  - using Internet protocols to provide secure communication
- There are two types of encryption
  1. symmetric key is more widely used and cheaper
  2. public key is more powerful
- Signatures are a means of verifying the origin and validity of messages

Summary (cont'd)

- Digests are hashes provide secure, low-cost signatures
- Certificates are a way to delegate trust
- Authentication protocols are surprisingly complex
  - most widely used = Kerberos

Next Lecture

- Tunneling, VPNs, and NAT