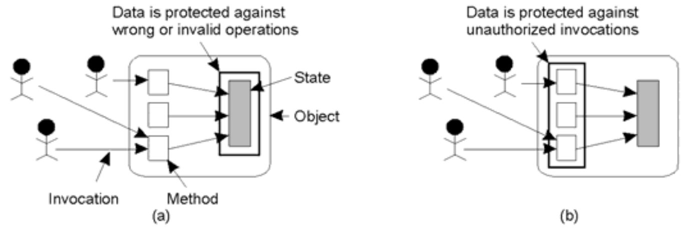
Security: Focus of Control

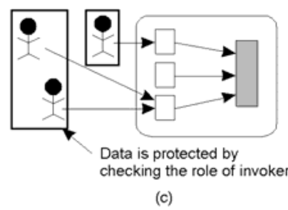


- Three approaches for protection against security threats

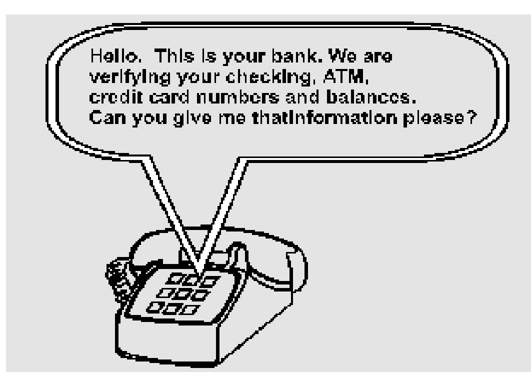
a) Protection against invalid operations

b) Protection against unauthorized invocations

c) Protection against unauthorized users



Authentication



- **Question:** how does a receiver know that remote communicating entity is who it is claimed to be?

Authentication Protocol (ap)

- Ap 1.0
  - Alice to Bob: “I am Alice”
  - Problem: intruder “Trudy” can also send such a message
- Ap 2.0
  - Authenticate source IP address is from Alice’s machine
  - Problem: IP Spoofing (send IP packets with a false address)
- Ap 3.0: use a secret password
  - Alice to Bob: “I am Alice, here is my password” (e.g., telnet)
  - Problem: Trudy can intercept Alice’s password by sniffing packets

Authentication Protocol

Ap 3.1: use encryption

use a symmetric key known to Alice and Bob

- Alice & Bob (only) know secure key for encryption/decryption

A to B: msg = encrypt("I am A")

B computes: if decrypt(msg)=="I am A"

then A is verified

else A is fradulent

- failure scenarios: playback attack
  - Trudy can intercept Alice’s message and masquerade as Alice at a later time

Authentication Using Nonces

Problem with ap 3.1: same password is used for all sessions
Solution: use a sequence of passwords
pick a "once-in-a-lifetime-only" number (nonce) for each session

Ap 4.0

A to B: msg = "I am A" /* note: unencrypted message! */
B to A: once-in-a-lifetime value, n
A to B: msg2 = encrypt(n) /* use symmetric keys */
B computes: if decrypt(msg2)==n
then A is verified
else A is fradulent

- note similarities to three way handshake and initial sequence number choice
- problems with nonces?

Authentication Using Public Keys

Ap 4.0 uses symmetric keys for authentication
Question: can we use public keys?

symmetry: DA(EA(n)) = EA (DA(n))

AP 5.0

A to B: msg = "I am A"
B to A: once-in-a-lifetime value, *n*
A to B: msg2 = DA(*n*)
B computes: if EA (DA(*n*))== *n*
then A is verified
else A is fradulent

Problems with Ap 5.0

- Bob needs Alice's public key for authentication
  - Trudy can impersonate as Alice to Bob
    - Trudy to Bob: msg = "I am Alice"
    - Bob to Alice: nonce n (Trudy intercepts this message)
    - Trudy to Bob: msg2= DT(n)
    - Bob to Alice: send me your public key (Trudy intercepts)
    - Trudy to Bob: send ET (claiming it is EA)
    - Bob: verify ET(DT(n)) == n and authenticates Trudy as Alice!!
- Moral: Ap 5.0 is only as "secure" as public key distribution

Man-in-the-middle Attack

- Trudy impersonates as Alice to Bob and as Bob to Alice
  - Alice Trudy Bob
  - "I am A" "I am A"
  - nonce n
  - DT(n)
  - send me ET
  - ET
  - nonce n
  - DA(n)
  - send me EA
  - EA
  - Bob sends data using ET, Trudy decrypts and forwards it using EA!! (Trudy *transparently* intercepts every message)

Digital Signatures Using Public Keys

Goals of digital signatures:

- sender cannot repudiate message never sent ("I never sent that")
- receiver cannot fake a received message

Suppose A wants B to "sign" a message M

B sends DB(M) to A

A computes if EB (DB(M)) == M

then B has signed M

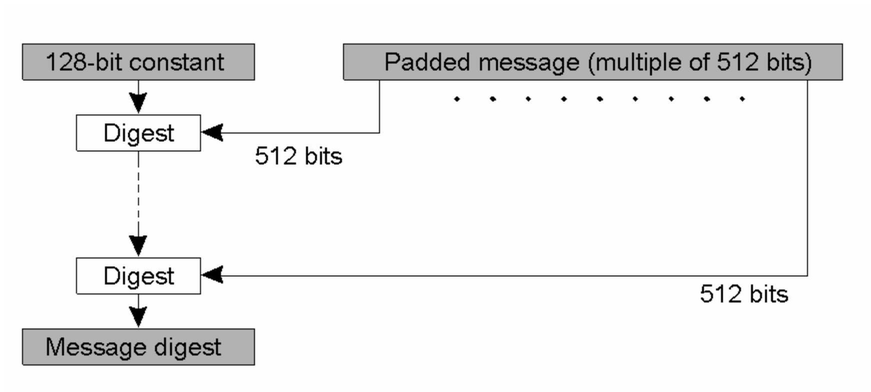
Question: can B plausibly deny having sent M?

Message Digests

- Encrypting and decrypting entire messages using digital signatures is computationally expensive
  - Routers routinely exchange data
    - Does not need encryption
    - Needs authentication and verify that data hasn't changed
- Message digests: like a checksum
  - Hash function H: converts variable length string to fixed length hash
  - Digitally sign H(M)
  - Send M, DA(H(m))
  - Can verify who sent the message and that it has been changed!
- Property of H
  - Given a digest x, it is infeasible to find a message y such that H(y) = x
  - It is infeasible to find any two messages x and y such that H(x) = H(y)

Hash Functions : MD5

- The structure of MD5



Symmetric key exchange: trusted server

Problem: how do distributed entities agree on a key?

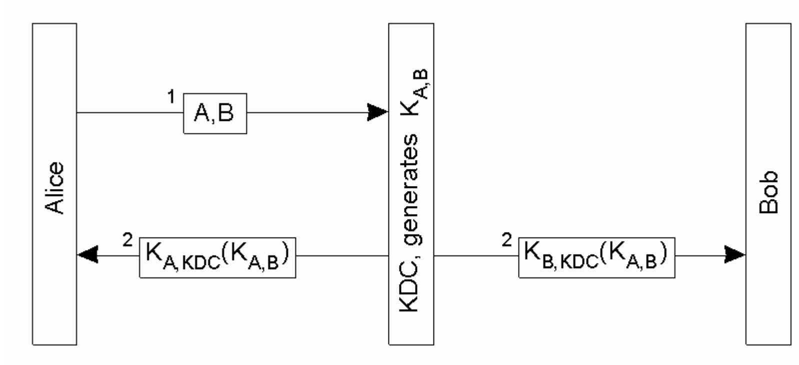
Assume: each entity has its own single key, which only it and trusted server know

Server:

- will generate a one-time session key that A and B use to encrypt communication
- will use A and B's single keys to communicate session key to A, B

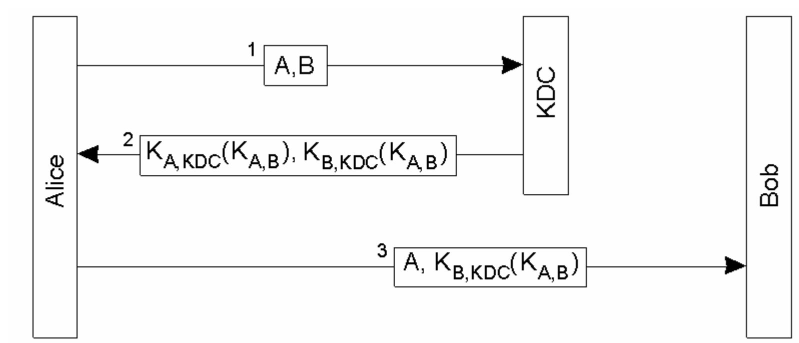
Key Exhange: Key Distribution Center (1)

- The principle of using a KDC.

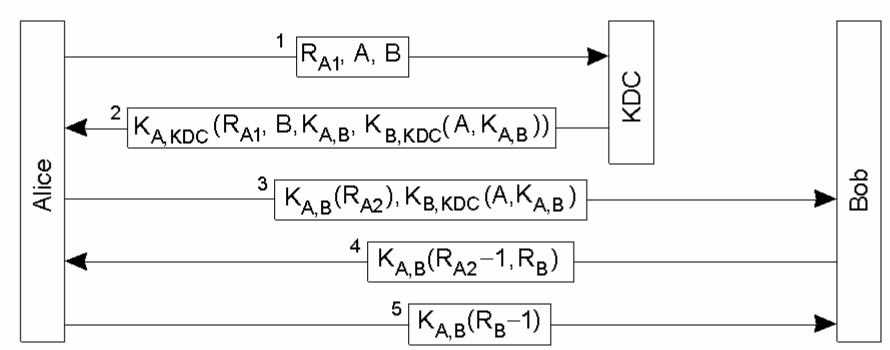


Authentication Using a Key Distribution Center (2)

- Using a ticket and letting Alice set up a connection to Bob.

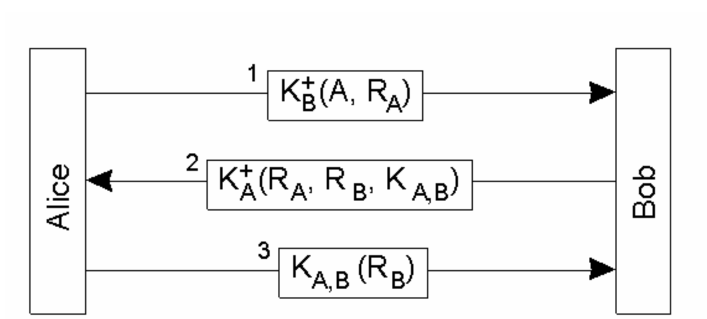


Authentication Using a Key Distribution Center (3)



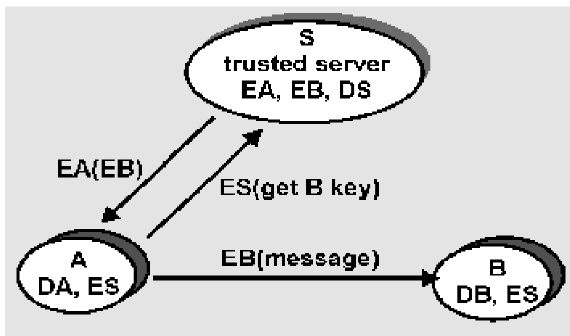
Public Key Exchange

- Mutual authentication in a public-key cryptosystem.



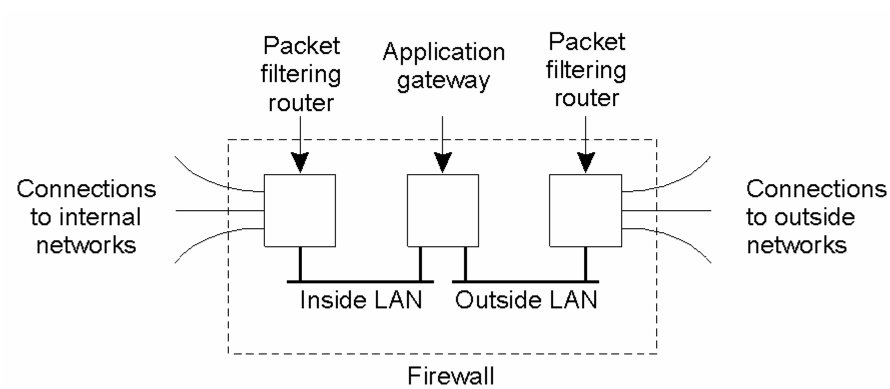
Public key exchange: trusted server

- public key retrieval subject to man-in-middle attack
- locate all public keys in trusted server
- everyone has server's encryption key (ES public)
- suppose A wants to send to B using B's "public" key



Protection Against Intruders: Firewalls

- A common implementation of a firewall.



Firewalls

Firewall: network components (host/router+software) sitting between inside ("us") and outside ("them)

Packet filtering firewalls: drop packets on basis of source or destination address (i.e., IP address, port)

Application gateways: application specific code intercepts, processes and/or relays application specific packets

- e.g., email of telnet gateways
- application gateway code can be security hardened
- can log all activity

Secure Email

- Requirements:
  - Secrecy
  - Sender authentication
  - Message integrity
  - Receiver authentication
- Secrecy
  - Can use public keys to encrypt messages
    - Inefficient for long messages
  - Use symmetric keys
    - Alice generates a symmetric key K
    - Encrypt message M with K
    - Encrypt K with E_B
    - Send K(M), E_B(K)
    - Bob decrypts using his private key, gets K, decrypts K(M)

Secure Email

- Authentication and Integrity (with no secrecy)
  - Alice applies hash function H to M (H can be MD5)
  - Creates a digital signature $D_A(H(M))$
  - Send M, $D_A(H(M))$ to Bob
- Putting it all together
  - Compute H(M), $D_A(H(M))$
  - M'= { H(M), $D_A(H(M))$ }
  - Generate symmetric key K, compute K(M')
  - Encrypt K as $E_B(K)$
  - Send K(M'), $E_B(K)$
- Used in PGP (pretty good privacy)

Secure Sockets Layer (SSL)

- SSL: Developed by Netscape
  - Provides data encryption and authentication between web server and client
  - SSL lies above the transport layer
  - Useful for Internet Commerce, secure mail access (IMAP)
  - Features:
    - SSL server authentication
    - Encrypted SSL session
    - SSL client authentication

Secure Socket Layer

- Protocol: https instead of http
  - Browser -> Server: B's SSL version and preferences
  - S->B: S's SSL version, preferences, and certificate
    - Certificate: server's RSA public key encrypted by CA's private key
  - B: uses its list of CAs and public keys to decrypt S's public key
  - B->S: generate K, encrypt K with with E_S
  - B->S: "future messages will be encrypted", and K(m)
  - S->B: "future messages will be encrypted", and K(m)
  - SSL session begins…

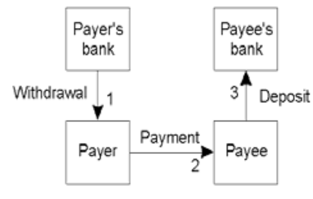
SSL

- Homework: get your own digital certificate
  - Click on "security" icon (next to "print" icon) in Netscape 4.7
  - Click on "Certificates" and then on "obtain your certificate"
  - Send an email to yourself signed with your certificate
  - Also examine listed of trusted CAs built into the browser

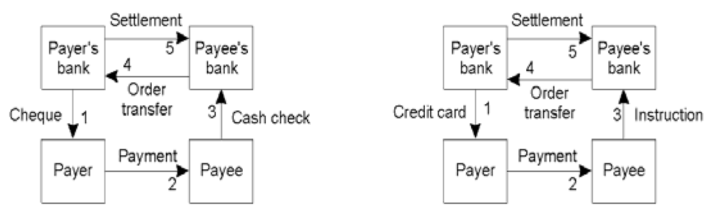
Electronic Payment Systems (1)

- Payment systems based on direct payment between customer and merchant.

a) Paying in cash.
b) Using a check.
c) Using a credit card.

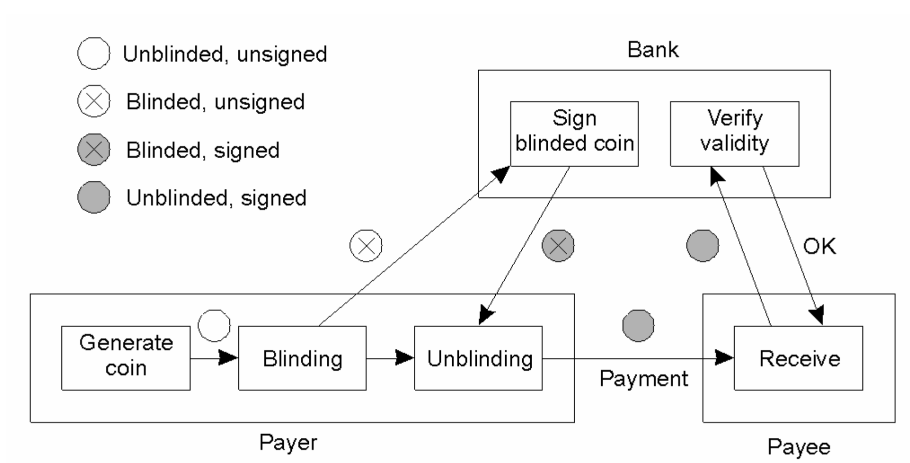


(a)

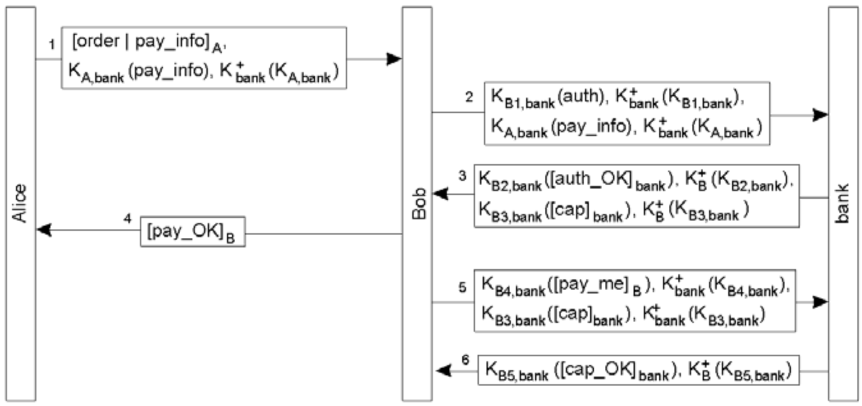


(b) (c)

E-cash



Secure Electronic Transactions (SET)



Security: conclusion

key concerns:

- encryption
- authentication
- key exchange

also:

- increasingly an important area as network connectivity increases
- digital signatures, digital cash, authentication, increasingly important
- an important social concern
- further reading:
  - Crypto Policy Perspectives: S. Landau et al., Aug 1994 CACM
  - Internet Security, R. Oppliger, CACM May 1997
  - www.eff.org