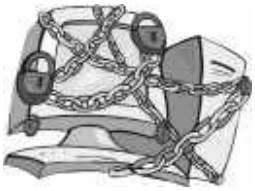
Security



Cryptography, Symmetric Key, Public Key, Authentication, Digital Signatures, Message Digests, Certificates, SSL, Signed applets, Secure e-mail, Secure credit card transactions, Firewalls

Security

What is Security?

- cryptography
- authentication
- message integrity
- availability

Security in Practice:

- application layer: secure document transfer, e-mail
- transport layer: Internet commerce, SSL/HTTPS
- network layer: IP security, firewalls, application gateways

What is Security?

- Cryptography: only sender and intended receiver are able to read message contents:
  - sender encrypts message
  - receiver decrypts message
- Authentication: sender, receiver want to confirm identity of each other and originator of messages.

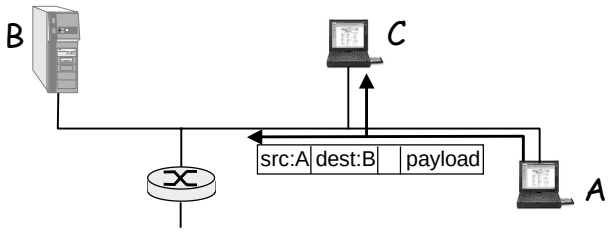
What is Security?

- Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection.
- Availability: systems, and services they provide, should not be disrupted by unauthorized access.

message secrecy/privacy threat

Packet Sniffing:

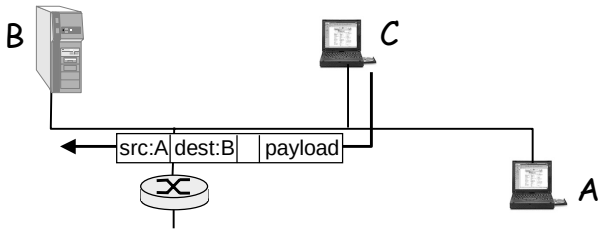
- broadcast media (e.g. Ethernet)
- promiscuous NIC reads all packets passing by
- can read all unencrypted data (e.g. passwords)
- e.g.: C sniffs A's packets destined for B



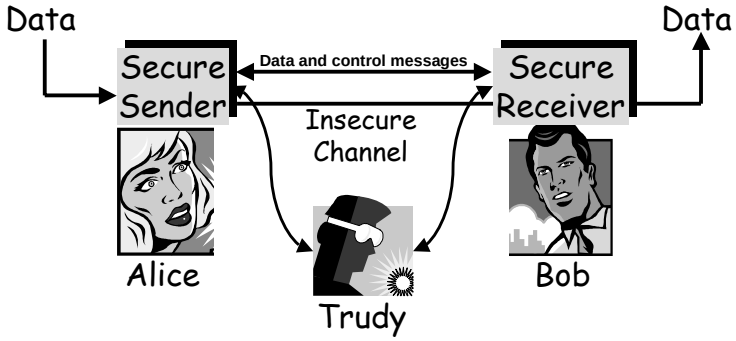
message security threat

IP Spoofing:

- Host can generate "raw" IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is real or spoofed
- e.g.: C pretends to be A

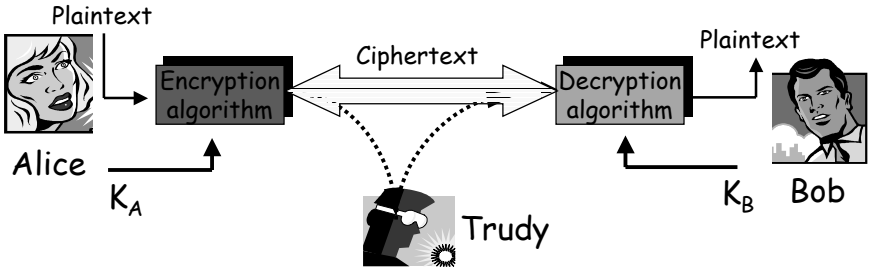


Friends and Foes: Alice, Bob, Trudy



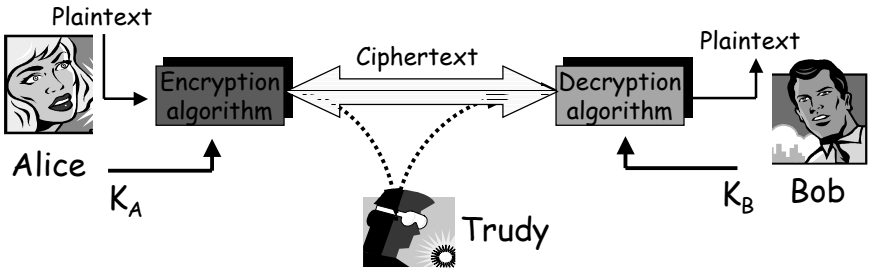
- Bob, Alice (more than friends!) want to communicate "securely"
- Trudy, the "intruder" may intercept, delete, add messages
- Communication channel "insecure"

The language of cryptography



- Plaintext – unencrypted information
- Ciphertext – encrypted information
- Key K_A together with Encryption algorithm transforms plaintext to ciphertext
- Key K_B together with Decryption algorithm transforms ciphertext to plaintext

The language of cryptography

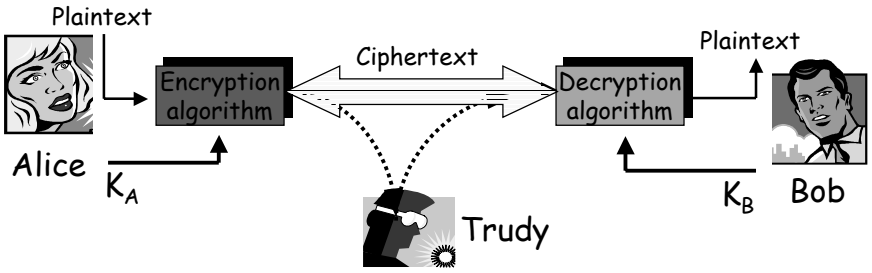


- K_A (Alice's key), K_B (Bob's key), e_B (Bob's encryption key), d_A (Alice's decryption key)
- Key – A fixed length sequence of bits
- The meaning of the bits in a Key depends on the encryption/decryption algorithm

Cryptography

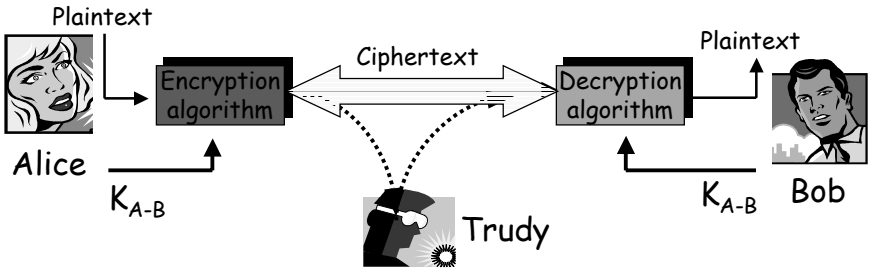
- Security provided:
  - by restricting knowledge of key(s) to trusted entities
  - not by obscurity -> algorithms/protocols used are assumed public
- Foundation for security:
  - algorithms that are computationally intractable to reverse without knowledge of the key(s) -> e.g., it would take the fastest present day computer millions of years to break/reverse the algorithm
- Encryption level proportional to data lifetime:
  - e.g. if currently sensitive data is useless in five minutes, then weak encryption may be OK

The language of cryptography



- Two basic kinds of key systems:
  - Symmetric-key cryptography: sender and receiver share *identical*, *secret* keys.
  - Public-key cryptography: encrypt key *public*, decrypt key *secret*

Symmetric Key Encryption

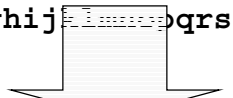


- Symmetric: both parties share single (secret) key that is used for both encryption and decryption.
- Examples: DES (data encryption standard).

Symmetric key cryptography

Simple example: *substitution cipher,* substituting one thing for another (Caesar Cipher)

monoalphabetic cipher: substitute one letter for another



plaintext: **abcdefghij qrstuvwxyz**

ciphertext: **mnbvcxzasdfg jklpoiuytrewq**

E.g.: **plaintext: bob. i love you. alice**

ciphertext: nkn. s gktc wky. mgsbc

Symmetric Key Cryptography

- Operations: substitution, transposition, and bitwise operations
- Through a series of rounds, the key and data are "scrambled" together to either encrypt or decrypt
- Why use these simple operations?
  - very fast
  - simple to implement in both hardware and software
  - sufficient repetition (number of rounds) with adequate key length can provide excellent security

Symmetric Key Cryptography

- Simple approach
  - data is divided into equal sized blocks
  - each is individually encrypted
  - forming the encrypted data stream
- Problem
  - messages usually have a regular structure or pattern
  - given the plaintext and ciphertext for several messages, their structure can be exploited to more quickly break the cipher

Symmetric Key Cryptography

- Solution
  - Cipher Block Chaining (CBC)
  - each plaintext block is XORed with the previous ciphertext block before it is encrypted.
- Now each ciphertext block depends on:
  - plaintext block that generated it
  - plus all previous plaintext blocks.
- CBC eliminates patterns that can compromise message secrecy.

Symmetric Key Example: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64 bit plaintext input
- making DES more secure
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining
- How secure is DES?
  - no known "backdoor" decryption approach
  - DES Challenge: 56-bit-key-encrypted phrase ("Strong cryptography makes the world a safer place") decrypted by brute force …

DES Security

- 22 hours, 15 min. by Electronic Frontier Foundation & Distributed.Net

Advances over DES

- RSA variable-sized key symmetric
  - RS5
    - 56-bit broken in 230 days
      - (can sell 40 / 56 bit overseas)
    - 128-bit will probably stand for a while
- AES (Advanced Encryption Standard)
  - Proposals being considered now
  - 128-bit and more keys
  - Should stand up to brute force attacks.

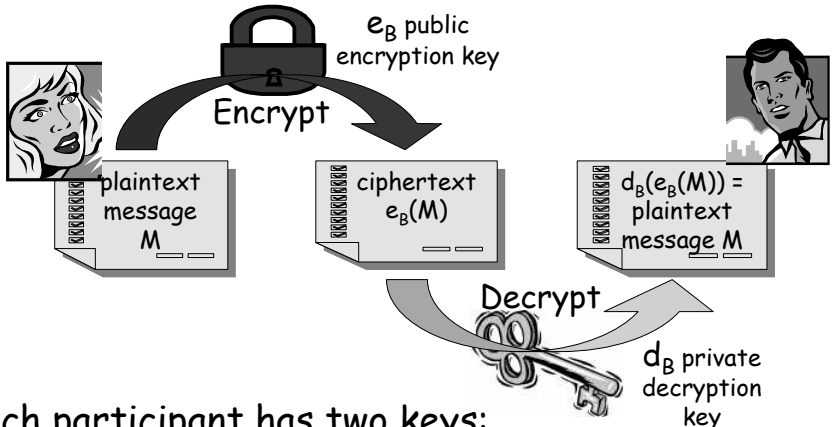
Public Key Cryptography

- radically different approach from symmetric-key [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- Each participant has two keys:

  private key is secret, used for decryption and digital signature

  public key, accessible to everyone, used for encryption and digital signature verification.

Public Key Encryption



Each participant has two keys:

- private key is secret
- public key freely accessible

Public Key Encryption algorithms

Two inter-related requirements:

1. need functions $d_B()$ and $e_B()$ such that for message m:

   $d_B(e_B(m)) = m$

2. where it is computationally hard (impossible in practice) to determine the private key given the public key.

Public Key Encryption algorithms

- RSA: Rivest, Shamir, Adelson most widely used implementation of public-key algorithm
- Based on computational infeasibility of factoring large numbers (at least 2^{512}) that can be found by multiplying 2 prime numbers
- If someone discovers an easy method of factoring large numbers, RSA would be out of business

RSA algorithm

- Take 2 large primes p, q and their product n = p*q, called the modulus
- Choose number e, less than n, such that e and (p-1)*(q-1) have no common factors
- Find another number d, such that (e*d)–1 is divisible by (p-1)*(q-1).
- e and d are called the public and private exponents
- public key is pair (n, e)
- private key is pair (n, d)

RSA algorithm

- Encryption algorithm uses public key e and message m to be encrypted, and forms ciphertext c using this formula:

  $c = m^e \bmod(n)$

- Decryption algorithm uses private key d and ciphertext c to reveal message m using this formula:

  $m = c^d \bmod(n)$

- m must be <= bit length of n.

RSA Security

- RSA-129 (p = 64 bits, q = 65 bits)
  - Martin Gardner published challenge in 1977. Rivest estimated time on PDP-10 to be 40 quadrillion years
  - 1994: done in 8 months by a loosely distributed network
- Current challenges not yet broken
  - RSA-576 ($10,000)
    - Factor the following number into 2 primes:
      - 188198812920607963838697239461650439807163563379417382700763356422988859715234665485319060606504743045317388011303396716199692321205734031879550656996221305168759307650257059
    - Likely to be solved soon
  - RSA-2048($200,000)
    - Should last a decade or so
- Not yet proven secure

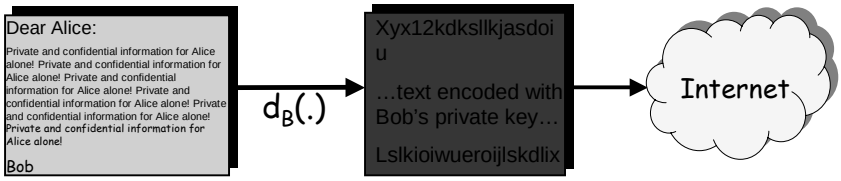
Public Key Encryption algorithms

- Public Key encryption is much more computationally expensive (slower) than symmetric encryption (because it is based on arithmetic operations on very large integers)
- In practice a hybrid system is employed with public keys being used to exchange symmetric session keys between parties

Digital Signatures

Cryptographic technique analogous to hand-written signatures

- Sender (Bob) digitally signs document with his private key $d_B(.)$, establishing he is the document owner/creator.
- Signature is verifiable, non-forgeable, non-repudiatable: recipient (Alice) can verify that Bob, and no one else, signed document.



Digital Signatures (cont'd)

- Suppose Alice receives msg *m*, and digital signature $d_B(m)$
- Alice verifies *m* signed by Bob by applying Bob's public key e_B to $d_B(m)$ *then* checks $e_B(d_B(m)) = m$.
- If $e_B(d_B(m)) = m$, whoever signed *m* must have used Bob's private key.

Alice thus verifies that:

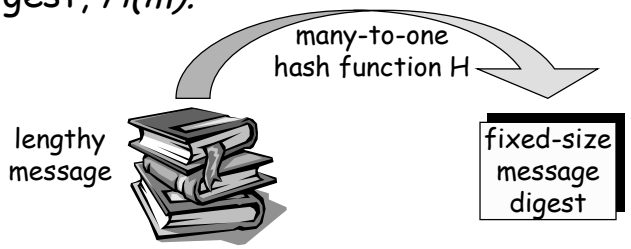
- Bob signed *m*.
- No one else signed *m*.
- Bob signed m and not *m'*.

Non-repudiation:

- Alice can take *m*, and signature $d_B(m)$ to court and prove that Bob signed *m*.

Message Digests

- Computationally expensive to public-key-encrypt long messages
  - Goal: fixed-length, easy to compute digital signature, "fingerprint"
- apply hash function H to *m*, get fixed size message digest, *H(m)*.



Message Digests

Hash function H() properties:

- Many-to-one
- Produces fixed-size msg digest (fingerprint) from variable size input
- Easy (cheap) to compute
- Public (non-secret) algorithms for H()
- Given message digest x, computationally infeasible to find m such that x = H(m)
- Computationally infeasible to find any two messages m and m' such that H(m) = H(m').

Message Digest

- One-way function: maps larger message into smaller, fixed-length number.
  - "Cryptographic checksum" over message.
  - Protects message against changes to its content, since can't find m' such that H(m)=H(m').
  - One-way function, so given H(m), practically impossible to derive message m and therefore can make the digest public w/o revealing the message content.
- Receiver, given message m and digest H(m), re-computes H(m) on received message m, and checks for match between the computed and received hash values.

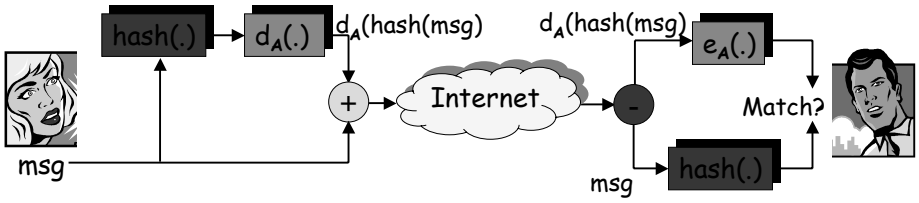
Digital Signature (Signed message digest)

Alice sends digitally signed message:

- Applies hash function to produce message digest
- Encrypts digest with private key
- Sends encrypted digest + unencrypted message

Bob verifies integrity of digitally signed message:

- Computes hash of unencrypted message
- Decrypts encrypted digest using Alice's public key
- Compares the results
- Match confirms msg integrity



Hash Function Algorithms

- Common checksum algorithms would make poor message digests:
  - Too easy to find two messages with same checksum (lose message integrity guarantee).
- MD5 hash function widely used.
  - Computes 128-bit message digest in 4-step process.
  - For arbitrary 128-bit string x, difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - U.S. standard
  - 160-bit message digest

Trusted Intermediaries

Problem:

- How do two entities establish shared symmetric secret key over network?

Solution:

- trusted *Key Distribution Center* (KDC) acting as intermediary between entities

Problem:

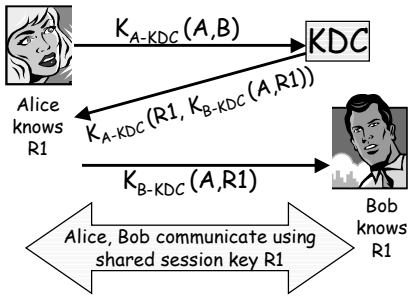
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is actually Bob's public key, not Trudy's?

Solution:

- trusted Certification Authority (CA) distributes certified public keys

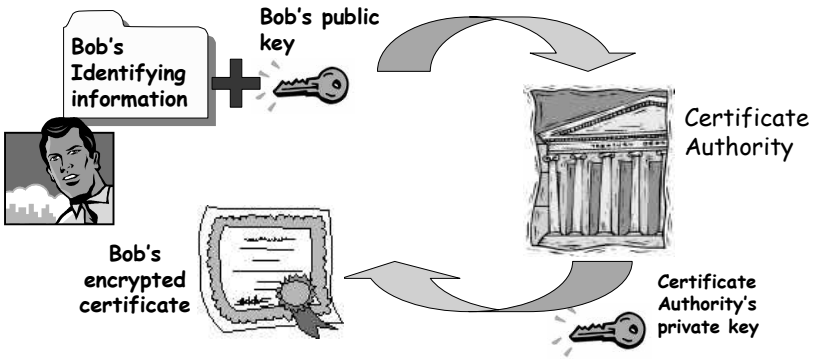
Key Distribution Center (KDC)

- Alice,Bob need shared symmetric key.
- KDC: server shares different secret key with each registered user.
- Alice, Bob each have own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.
- Alice communicates with KDC, gets session key R1, and $K_{B\text{-}KDC}(A,R1)$



- Alice sends Bob $K_{B\text{-}KDC}(A,R1)$, Bob extracts R1
- Alice, Bob now share the symmetric, secret key R1.

Certificate Authority



- Certificate Authority (CA) binds public key to particular entity (person, router, etc).

Certificate Authority

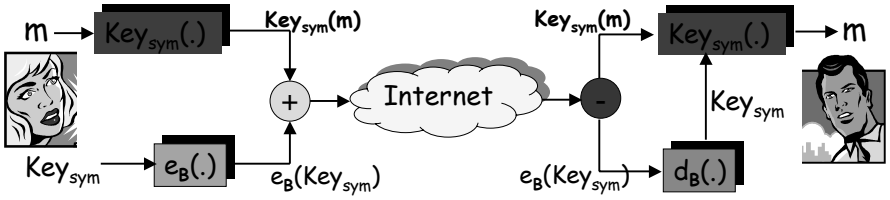
Entity registers its public key with CA:

- Entity provides "proof of identity" to CA at time of registration.
- CA creates certificate binding entity to public key.
- Certificate digitally signed by CA.

- When Alice wants Bob's public key:
- gets Bob's certificate (from Bob or elsewhere).
- Apply CA's public key to Bob's certificate to get Bob's public key

Secure e-mail (e.g. PGP)

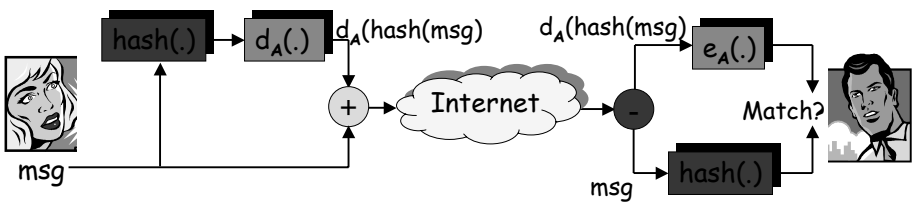
- Alice wants to send *secret* e-mail message, m, to Bob.



- generate random *symmetric* private session key, K_S
- encrypt message with K_S
- encrypt K_S with Bob's public key
- sends both $K_S(m)$ and $e_B(K_S)$ to Bob

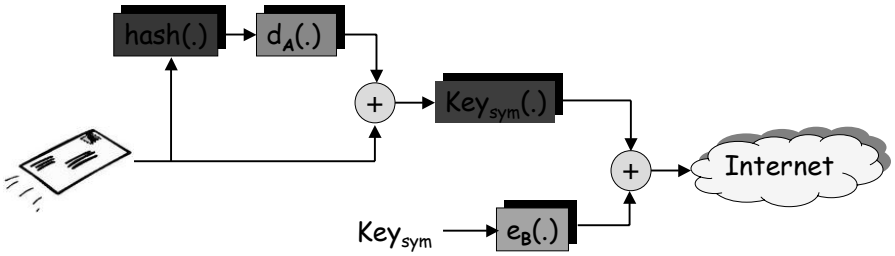
Authenticated e-mail

- Alice wants to provide sender authentication and message integrity
- Alice digitally signs message with secret key d_A
- Alice sends cleartext message together with digital signature
- Bob compares a hash of the cleartext message with the decoded hash of the encoded hashed message



Secure, Authenticated e-mail

- Alice wants to send secret e-mail to Bob with sender authentication and message integrity.
- Note, Alice uses both her *private* key and Bob's *public* key.



Pretty Good Privacy (PGP)

- Internet e-mail encryption scheme, a de-facto standard.
- Uses symmetric key cryptography, public key cryptography, hash function, and digital signature
- Provides secrecy, sender authentication, integrity and efficiency for handling large messages.

Example PGP signed message:

```
---BEGIN PGP SIGNED
  MESSAGE---
Hash: SHA1

Bob:Felix is out of town
  on a business trip
  tonight. Amorously
  yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4
  vB3mqJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

Secure Sockets Layer (SSL)

- RSA/PGP provide security for specific network applications
- SSL works at Internet transport layer. Provides security to any TCP-based application using SSL services.
- Used between WWW browsers and servers for e-commerce (https://www…).
- Security services:
  - server (and optionally client) authentication using public key signatures
  - encryption of all data flowing between client and server (including URL's, any submitted form contents, data returned by the server, etc)

SSL/HTTPS (cont'd)

- Server authentication:
  - SSL-enabled browser includes public keys for trusted Certificate Authorities (CAs)
  - Browser requests server certificate, issued by trusted CA
  - Browser uses CA's public key to extract server's public key from certificate
- Visit your browser's security menu to see its trusted CA's

SSL/HTTPS (cont'd)

Encrypted SSL session:

- Browser generates symmetric session key, encrypts it with server's public key, sends encrypted key to server
- Using its private key, server decrypts session key
- Browser, server agree that future messages will be encrypted
- All data sent into TCP socket (by client and server) is encrypted with session key

SSL/HTTPS (cont'd)

- HTTPS and HTTP use different ports, so single server can provide both secure and insecure service simultaneously
- Use of different URL schemes (http and https), mean that a non-SSL-capable browser will not even attempt to send secure info collected through a form with submit action specified as https:…
- Client authentication can be done with client certificates
- SSL can be used for non-Web applications, such as an e-mail user agent

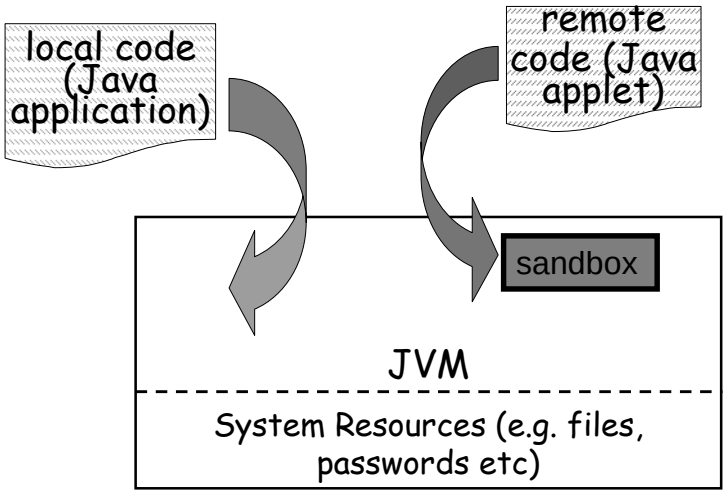
Signed Java Applets

- Why? Download applets from source about which you know nothing – how to be sure the applet is not malicious?
- Code is authenticated by checking its digital signature using the source's public key to verify the author and that code hasn't been modified.
- the signature tells who the applet comes from, and that the applet has not been tampered with. A signature doesn't tell you anything about the content or quality of the applet, just that it comes from the source that signed it.

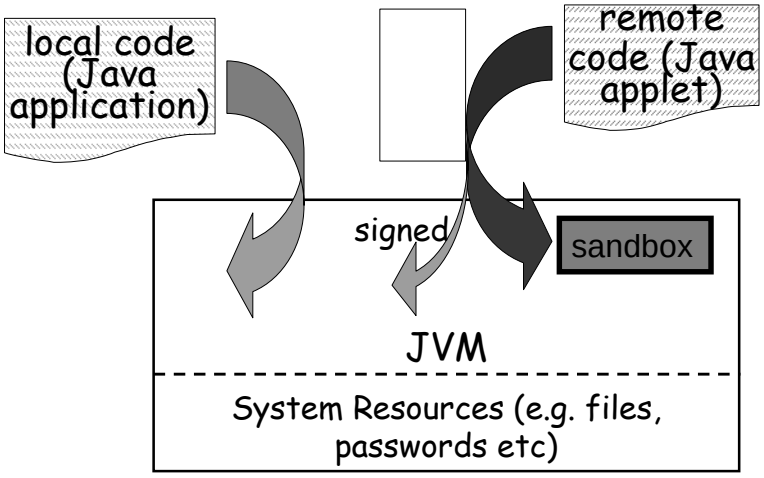
Signed Applets

- Loading a signed applet triggers a Java security dialog, indicating that a java applet from "the sender's signature id" is requesting additional privileges, e.g. reading, modification, or deletion of any of your files. Granting the privilege is noted as high risk.
- The Java Security dialog should indicate "Identity verified by *original issuer of certificate*" and display a button for you to examine the certificate
- Client can grant permissions for applets to execute or not based on who signed the code.

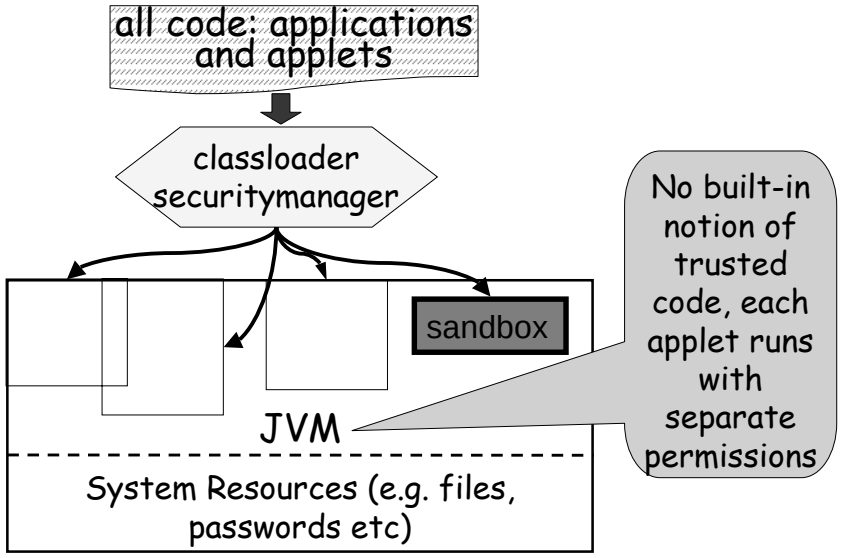
JDK 1.0 Security Model



JDK 1.1 Security Model



JDK 1.2 Security Model



Java Security

- JDK 1.2 designed to support easily configurable security policy
- In JDK 1.2, no longer a built-in concept that all local code is trusted
- local code (e.g., non-system code, application packages installed on the local file system) is subject to the same security control as applets
- the same principles apply to signed applets and any Java application.

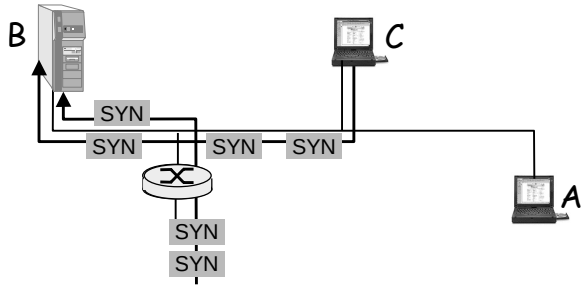
Signing Java applets

- To sign a jar file you need public/private keys and a certificate authority to validate the public key
- A signed jar file includes at least one signature file, as well as normal manifest file
- JDK provides these tools for signing applets:
  - keytool – manages identities, keys and certificates in JDK identity DB
  - jarsigner – generates signatures for jar files and verifies signatures for signed jar files
  - policytool – creates and modifies policy configuration files that define security policy

Internet availability threats

Denial of service (DOS):

- flood of maliciously generated packets "swamp" receiver
- Distributed DOS (DDOS): multiple coordinated sources swamp receiver
- e.g., C and remote host SYN-attack B



Secure Electronic Transactions

- Secure Electronic Transactions (SET) designed for payment-card transactions over the Internet
  - Open technical standard developed by Visa & MasterCard
- provides security services among 3 players:
  - customer
  - merchant
  - merchant's bank
  - All must have certificates.
- SET specifies legal meanings of certificates
  - apportionment of liabilities for transactions

Secure Electronic Transactions

- Customer's card number passed to merchant's bank without merchant ever seeing number in plain text
  - Prevents merchants from stealing or leaking payment card numbers
- Uses all of the techniques previously described.

Secure Electronic Transactions

- Ensures:
  - Confidentiality of info – implemented using symmetric key
  - Integrity of data – implemented using digital signature
  - Cardholder authentication – implemented with digital signature and cardholder certificates
  - Merchant authentication – implemented with digital signature and merchant certificates

Firewalls

firewall

isolates organization's internal net (*Intranet*) from larger Internet, allowing some packets to pass, blocking others.

Two categories of firewall:

- packet filter
- application gateways

Firewalls are used to prevent:

- Denial of Service (DoS) attacks:
  - SYN flooding: attacker establishes many bogus TCP connections. Attacked host alloc's TCP buffers for bogus connections, none left for "real" connections.
- Illegal modification of internal data:
  - e.g., attacker replaces legitimate electronic banking records with false transaction information
- Intruders from obtaining secret internal data:
  - e.g. proprietary software source code, such as MS Word
- Tampering that could disrupt internal operations:
  - e.g. configuration changes on corporate servers or network switches

Packet Filtering

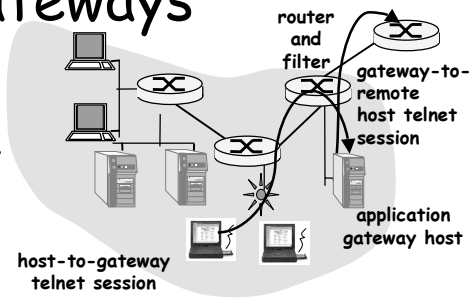
- Internal network is connected to Internet through a router.
- Router manufacturer provides options for filtering packets, based on:
  - source IP address
  - destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

Packet Filtering (cont)

- Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and outgoing UDP flows and telnet connections are blocked.
- Example 2: Block inbound TCP segments with ACK=0.
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Application gateways

- Filters packets based on *application data* as well as on IP/TCP/UDP fields.
- Example: allow select internal users to telnet outside.



1. Require all telnet users to telnet through gateway.
2. For authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. Router filter blocks all telnet connections not originating from gateway.

Limitations of Filters and Gateways

- IP spoofing: router can't know if data "really" originated at source listed in header
- If multiple app's. need special treatment, each has own app. gateway.
- Client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- Filters often use all or nothing policy for UDP.
- Tradeoff: degree of communication with outside world vs level of security
- Many highly protected sites still suffer from attacks.

Network Security (summary)

Basic techniques……

- cryptography (symmetric and public key)
- authentication
- message integrity
- firewalls and application gateways

…. used in many different security scenarios

- secure e-mail
- secure Web transactions (https)
- secure electronic transactions
- infrastructure protection