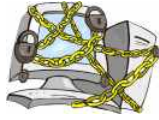


Security



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

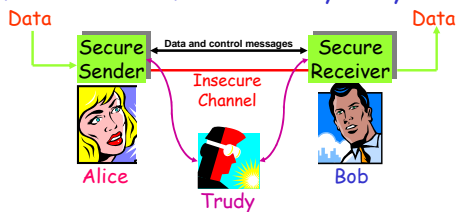
1

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.
- **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.

2

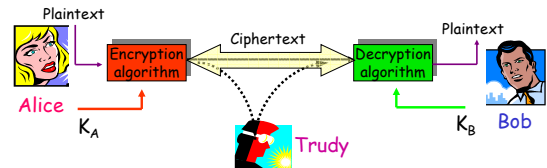
Friends and Foes: Alice, Bob, Trudy



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

3

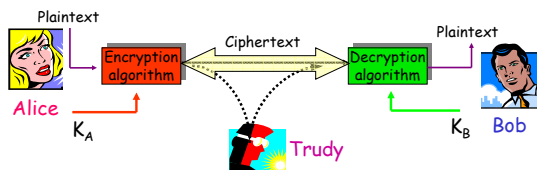
The Language of Cryptography



- 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

4

The Language of Cryptography



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

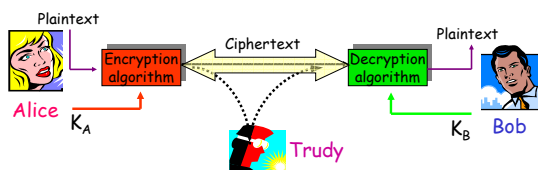
5

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

6

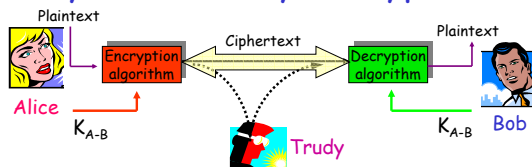
The Language of Cryptography



- 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*

7

Symmetric Key Encryption



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

8

Symmetric Key Cryptography

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

monoalphabetic cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfrgjkpoiuytrewq

E.g. plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc

9

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

10

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

11

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.

12

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

13

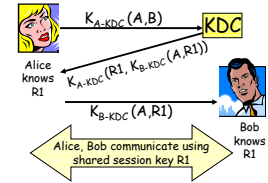
Key Distribution Center (KDC)

- 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-KDC} , K_{B-KDC} , for communicating with KDC.

- Alice communicates with KDC, gets session key $R1$, and $K_{B-KDC}(A,R1)$



- Alice sends Bob $K_{B-KDC}(A,R1)$, Bob extracts $R1$

- Alice, Bob now share the symmetric, secret key $R1$.

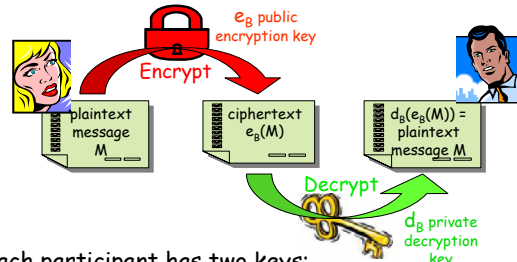
14

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.

15

Public Key Encryption



Each participant has two keys:

- private key** is secret
- public key** freely accessible

16

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.

17

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

18

RSA algorithm

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

19

RSA algorithm

- Encryption algorithm uses public key e and message m to be encrypted, and forms ciphertext c using this formula:
 $c = m^e \text{ mod}(n)$
- Decryption algorithm uses private key d and ciphertext c to reveal message m using this formula:
 $m = c^d \text{ mod}(n)$
- m must be \leq bit length of n .

20

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:
 $1881988129206079638388972394616004398071636537941738270076356422688697152346654853190606065$
 $04743046317388011303396716199892321205734031879550656996221305168759307650257059$
 - Likely to be solved soon
 - RSA-2048 (\$200,000)
 - Should last a decade or so
- Not yet proven secure

21

Public Key Encryption algorithms

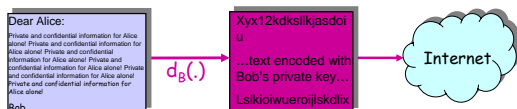
- Public Key encryption is much more computationally expensive (slower) than symmetric encryption (because it is based on arithmetic operations on very large integers)
- In practice a hybrid system is employed with public keys being used to exchange symmetric session keys between parties

22

Digital Signatures

Cryptographic technique analogous to hand-written signatures

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



23

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' .

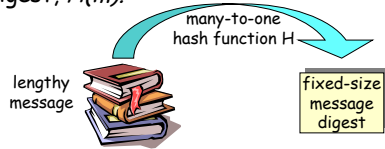
Non-repudiation:

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

24

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, $H(m)$.



25

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 $H(m) = H(m')$.

26

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.

27

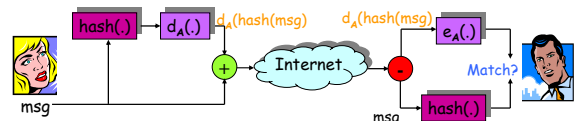
Digital Signature (Signed message digest)

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



28

Trusted Intermediaries

Problem:

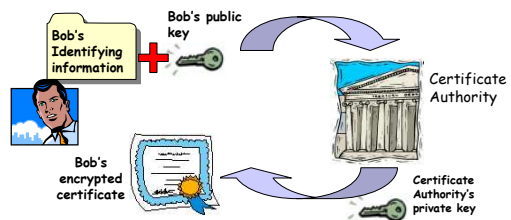
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is actually Bob's public key, not Trudy's?

Solution:

- trusted Certification Authority (CA) distributes certified public keys

29

Certificate Authority



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

30

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

31

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)

32

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

33

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

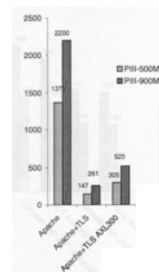
34

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

35

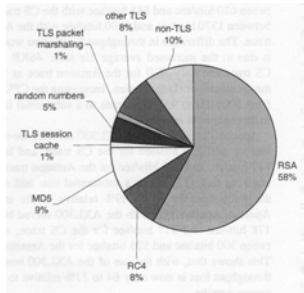
HTTPS Performance



Coarfa, Druschel, and Wallach 2002

36

HTTPS Performance



37

HTTPS and Tomcat

- Digital certificate
 - Certification authority
 - VeriSign Inc. TrustCenter, Thawte Consulting
 - Self certified certificate
 - JRE keytool
 - `keytool -genkey -alias tomcat -keyalg RSA -keystore <CATALINA_HOME>/keystore`
- Enable Tomcat's SSL Connector
 - Manually edit `server.xml`
 - Tomcat Administration tool

Type	HTTPS
Port number	8443
Key store filename	.keystore
Key store password	secret

38

HTTPS and Apache

- Implemented in `mod_ssl` module
- Only available in source code
- Requires *GCC* and the *OpenSSL* library
- Massive changes to `httpd.conf`

- Apache 2 with SSL/TLS: Step-by-Step
 - <http://www.securityfocus.com/infocus/1818>

39