

Security and the Internet

Original key design goals of Internet protocols:

- resiliency
- availability
- scalability

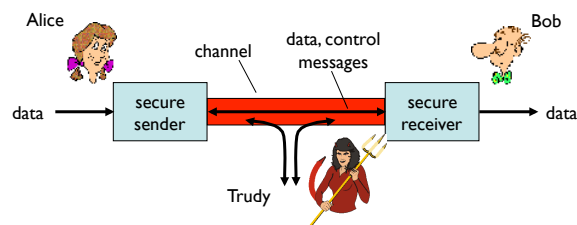
Security has not been a priority until mid 1990s

Security Attacks

Cast of characters: Alice, Bob, Trudy, well-known characters in network security world

Bob, Alice want to communicate “securely”

Trudy (intruder) may intercept, delete, add messages



Two types of security attacks:

1. against information content: secrecy, integrity, authentication, authorization, privacy, anonymity
2. against the infrastructure: system intrusion, denial of service

Security Attacks Against Content

Two types of attack against content: passive and active

Passive attacks: glean unauthorized information

- eavesdropping: intercept messages
- password “sniffing”
- traffic analysis (covert channel)
- timing analysis
- light emission analysis
- electro-magnetic emission analysis

Active attacks: modify information

- data modification: actively insert messages into connection
- play-back
- repudiation (denial) of transaction

Security Requirements

Confidentiality/secretcy: only parties involved sender, intended receiver(s) should know of (the content of) the transaction

- sender encrypts message
- receiver(s) decrypt(s) message

Authentication: sender, receiver(s) want to confirm each other's identity

- based on digital signature and certification
- compare: authorization

Security Requirements

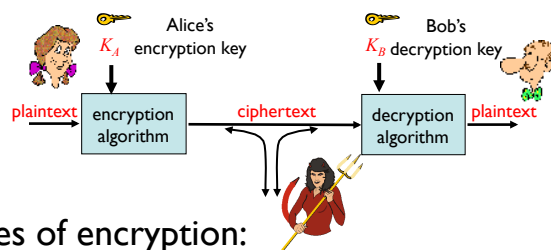
Data Integrity: sender, receiver(s) want to ensure data not altered (in transit, or afterwards) or if altered, detectable

Non-repudiation: involved parties cannot deny participation afterwards

Access and availability: services must be accessible and available to users

Counter Measure: Cryptography

Fundamental tool for achieving network security



Two types of encryption:

1. symmetric key cryptography:
 - both parties share a **secret key** that is used for both encryption and decryption
 - example algorithm: DES, AES
 - example system: Kerberos
2. public-key cryptography: encryption key *public*, decryption key *secret* (private)

DES: Data Encryption Standard

US encryption standard [NIST 1993]

56-bit symmetric key, 64-bit plaintext input

How secure is DES?

- DES Challenge: 56-bit-key-encrypted phrase (“Strong cryptography makes the world a safer place”) decrypted (brute force, try all possible keys) in 96 days in 1997
 - in early 1998 it took 41 days to crack, in mid-1998, 56 hours, in Jan 1999 only 22 hours 15 minutes
- no known “backdoor” decryption approach

Making DES more secure:

- use three keys sequentially (3-DES)
on message: $\text{DES}(k_3; \text{DES}(k_2; \text{DES}(k_1; M)))$
- use cipher-block chaining: encrypted j -th block is XOR-ed with plain $(j+1)$ -st block and then encrypted

AES: Advanced Encryption Standard

New (Nov. 2001) symmetric-key NIST standard,
replacing DES

128-bit plaintext input

128-, 192-, or 256-bit (Optimal AES) symmetric keys

Brute force attack (try each key) taking 1 sec on DES,
takes 149 trillion years for AES

Kerckhoff's principle: “The security of a cryptosystem must not depend on keeping secret the crypto-algorithm. The security depends only on keeping secret the key.” Translated from *La cryptographie militaire* (1883)

Key Escrow

Symmetric key cryptography requires participants to know shared secret key

Q: how to agree on shared key in the first place (particularly if they never “met”)?

Shared key can be distributed by **key escrow** or **key distribution center (KDC)**:

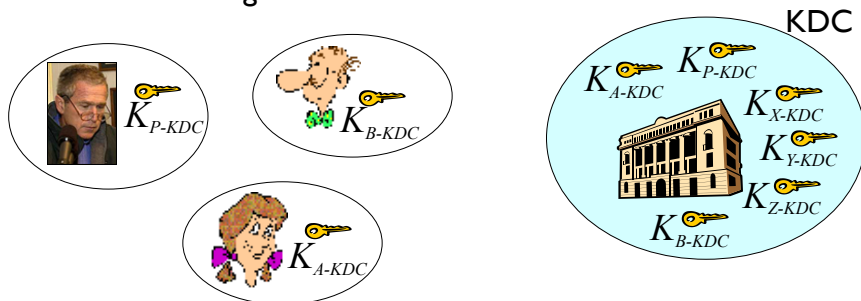
- escrow shares secret keys with both parties
- generates a session key for each session between the two parties

Key Distribution Center (KDC)

Alice, Bob need shared symmetric key

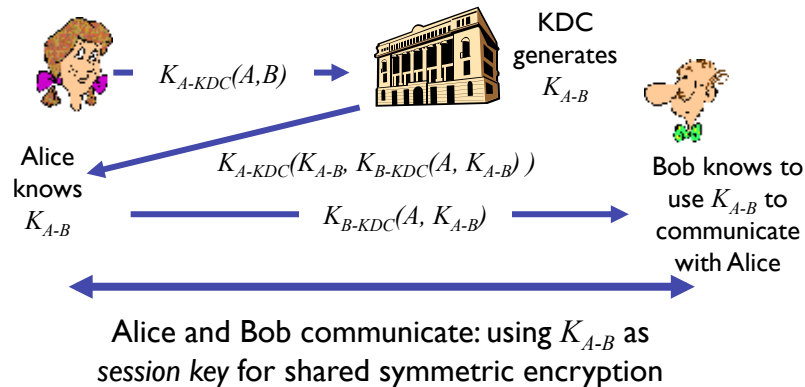
KDC: server shares different secret key with *each* registered user (many users)

Alice, Bob know own symmetric keys, K_{A-KDC} K_{B-KDC} , for communicating with KDC



Key Distribution Center (KDC)

How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Kerberos: an Authentication Service

Fundamental trade-off: security vs. convenience

Most secure, least convenient: not networked,
placed in a secure locked room

To access a networked of services, we can
challenge the users each time they want to use a
service, or authenticate them once and grant them
tickets to use several services without further
(user-level) challenge for a duration of time
(Kerberos)

Kerberos: an Authentication Service

Kerberos uses DES symmetric key: generates a shared key for each user-service pair that is valid for a period of time

How Kerberos works: three parties:

1. Authentication Server (AS)
2. client (*c*): application
3. server (*v*) for various services, e.g., name server, file server, print server, etc.

Kerberos Terminology

Authentication: verification of the ID of a party and the integrity of the data it generates

Principal: party whose ID is to be verified

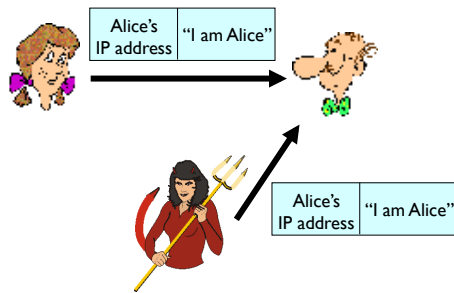
Verifier: party who is requesting verification

Data integrity: data received is the same as data generated

Nonce: a cookie that tells how fresh a piece of data is, usually a timestamp

Authentication: IP Spoofing

Bob wants Alice to “prove” her identity to him



Trudy can create a packet, “spoofing” Alice’s address

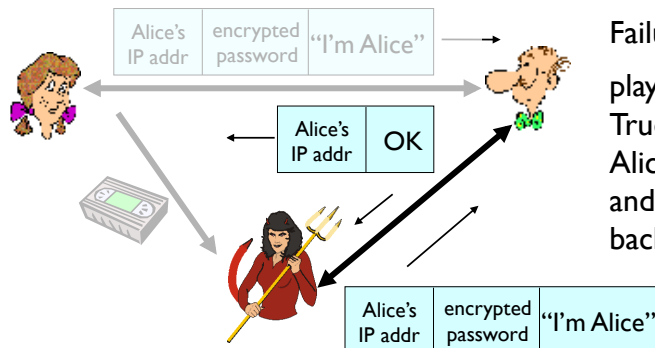
Failure scenario?



Cartoon by Peter Steiner. The New Yorker, 69(20):61, July 5, 1993

Authentication: Playback Attack

Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it



Failure scenario?

playback attack:
Trudy records Alice's packet and later plays it back to Bob

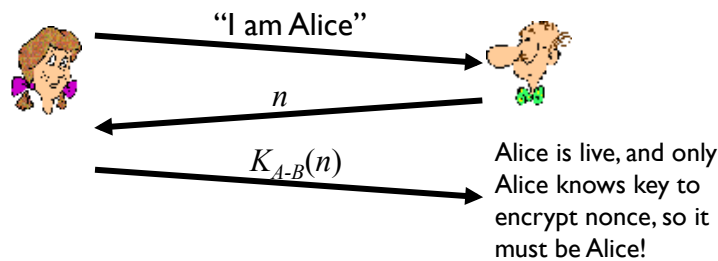
Authentication: Use of Nonce

Nonce used to avoid playback attack

Nonce: a number (n) used only *once* –in-a-lifetime

To prove Alice “live”, Bob sends Alice nonce, n

Alice must return n , encrypted with shared secret key



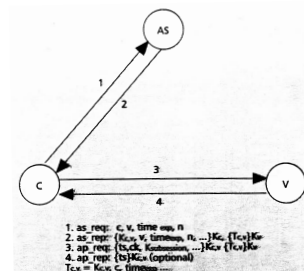
Kerberos Authentication

Authentication Server (AS):

1. keeps a list of all clients' passwords (K_c 's)
2. shares a key with each service (K_v)

Client (c):

1. asks AS for a session key for a specific server (v) for a period of time, provide nonce (n): $c, v, \text{time}_{\text{exp}}, n$
2. gets back a session key ($K_{c,v}$) with expiration time, and nonce (n), encrypted with client's password (K_c) ($\{K_{c,v}, v, \text{time}_{\text{exp}}, n, \dots\} K_c$) and a ticket ($T_{c,v}$) for server, encrypted using server's key (K_v), $\{T_{c,v}\} K_v = \{K_{c,v}, c, \text{time}_{\text{exp}}, \dots\} K_v$
3. sends data (encrypted with session key, $K_{\text{subsession}}$), along with ticket ($\{T_{c,v}\} K_v$) and authenticator ($\{t_s, c_k, K_{\text{subsession}}\} K_{c,v}$), where t_s is the nonce and c_k is checksum



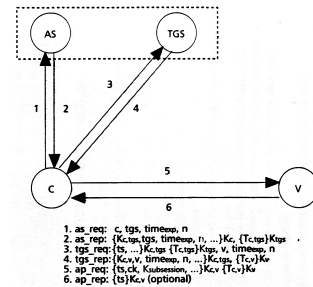
data integrity ensured by use of session key

ticket verifies client's ID to server

Kerberos Authentication Protocol

Server (v):

1. decrypts and “unpacks” $T_{c,v}$ to obtain $K_{c,v}$, makes sure it belongs to c and time hasn’t expired
2. decrypts authenticator ($\{t_s, c_k, K_{\text{subsession}}\}K_{c,v}$), checks that nonce, t_s , is within window (5 min) and has not been used
3. decrypts data using $K_{\text{subsession}}$ (optional)
4. checks that checksum, c_k , matches
5. responds with $\{t_s\}K_{c,v}$ (optional)



Ticket Granting Service:

- Each service requires a separate ticket
- Client must prompt user for password for each ticket
- More convenient: use a ticket-granting service with ticket that lives for a “short” period of time (8 hours)
- Kerberos still relies on user password, which could be “spoofed”

One-time Passcode

RSA SecurID tokens
(has a built-in accurate clock)

Constantly changing passcode

Protection against password spoofing

Secure ID card that generates a random number as passcode

Each passcode is good for 4 minutes

Login challenge is user’s password plus the random number

Passcode generation algorithms

- math algorithm: generates next passcode based on the previous one, e.g., hash chain
- time synchronization between client and authentication server
- math algorithm: next passcode based on a challenge and counter (e.g., used by smart cards)



Public Key Cryptography

Symmetric key cryptography requires participants to know shared secret key

Q: how to agree on shared key in the first place (particularly if they never “met”)?

Public key cryptography

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do not share secret key
- public encryption key known to all
- private decryption key known only to owner
- data encrypted with private key, decrypted with public key (or vice versa)
- ciphertext encrypted using the public key can be decrypted by the private key (and vice versa)

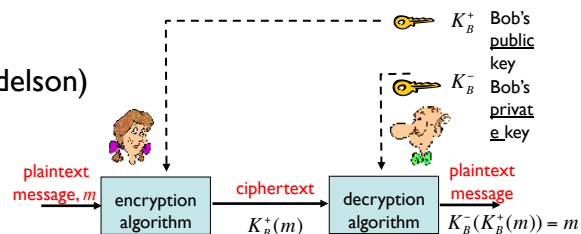
Public Key Encryption Algorithms

Requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that $K_B^-(K_B^+(m)) = m$
2. given public key, it should be impossible to compute private key
3. furthermore, $\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$

Result is the same!

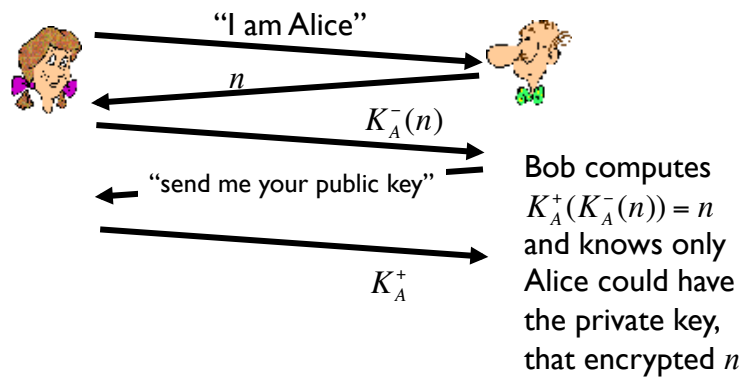
Example algorithm:
RSA (Rivest, Shamir, Adelson)



How to Obtain Public Key?

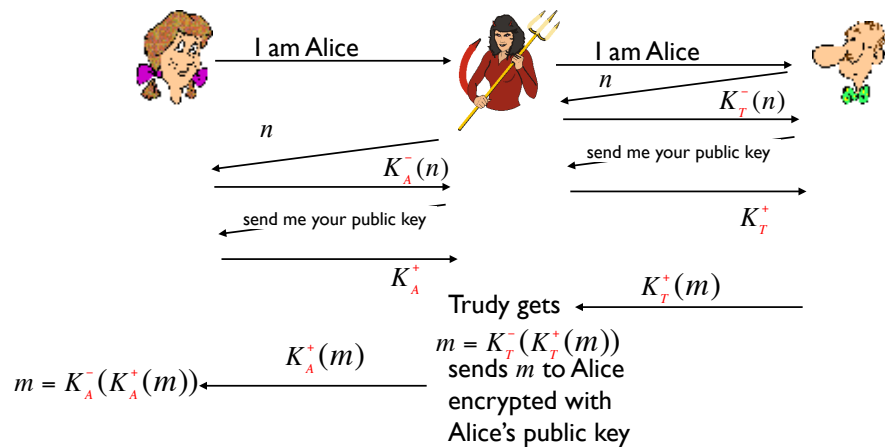
Directly from the owner?

Consider the authentication problem from earlier



Man in the Middle Attack

Trudy poses as Alice (to Bob) and as Bob (to Alice)



Man in the Middle Attack



Difficult to detect:

- Bob receives everything that Alice sends, and vice versa (e.g., so Bob, Alice can meet one week later and recall conversation)
- problem is that Trudy receives all messages as well!

Trusted Intermediaries

Symmetric key problem:

How do two entities establish shared secret key over network?

Solution:

Trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

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

Solution:

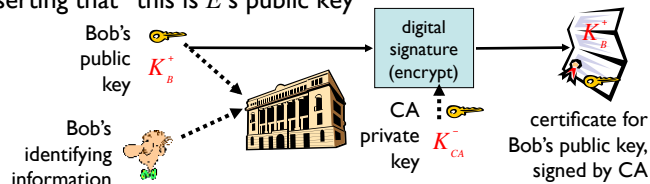
Trusted certification authority (CA)

Certificates and Certification Authorities

Certification Authority (CA) for public keys:

binds public key to particular entity, E

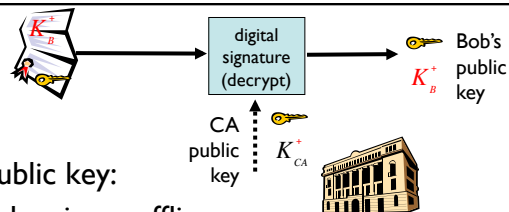
- E (person, router) registers its public key with CA securely, offline
- E provides “proof of identity” to CA
- CA “certifies” the authenticity of E ’s public key by creating a certificate binding E to its public key
- E ’s **certificate** is its public key encrypted (digitally signed) using the CA’s private key
- CA is in effect asserting that “this is E ’s public key”



Public Key Distribution

When Alice wants Bob's public key:

- Alice obtains CA's public key in an offline, secure manner (comes with browser code download, how secure is that?)
- Alice gets Bob's certificate (from Bob or from elsewhere)
- Alice decrypts Bob's certificate using the CA's public key to get Bob's public key



CA periodically publishes

Certification Revocation List (CRL)

for revoked public-keys (not currently done, how about revoking CA's public key?)

Public Key Infrastructure (PKI)

A hierarchy of CAs

Relies on chain of trust (*speak-for* relationship)

Examples:

- Verisign, Entrust, Thawte, GlobalSign, Visa, GTE, DigiCert, Equifax, etc.
- see
Firefox→Preferences→Advanced→Encryption→View Certificates→Authorities

A Certificate Contains:

Serial number (unique to issuer)

info about certificate owner, including algorithm and key value itself (not shown)

info about certificate issuer

valid dates

digital signature by issuer

Performance

On a 1 GHz PentiumIII with 128 MB of RAM, running Windows,

- DES 56-bit key performs at 43.3 Mbps
- RSA 512-bit key encrypt/decrypt speed: 543/45.5 Kbps
- RSA 768-bit key: 384/24.8 Kbps
- RSA 1024-bit key: 275/14.6 Kbps
- RSA decryption is 12-19 times slower than encryption

Symmetric keys are also more resistant to brute-force attacks
Common practice: use public-key to distribute symmetric session key (e.g., Zattoo)

Table 7.9
Symmetric and Public-key Key Lengths
with Similar Resistances to Brute-Force Attacks

Symmetric Key Length	Public-key Key Length
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

Schneier

Digital Signatures

Cryptographic technique analogous to hand-written signatures

Sender (Bob) digitally signs document by encrypting the document using his private key, establishing he is document owner/creator

Verifiable, non-forgable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Non-repudiation: Alice can take message m , and signature $K_B^-(m)$ to court and prove that Bob signed m

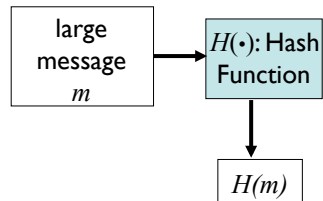
Message Digest

Computationally expensive to public-key-encrypt long messages

For purposes of authentication and certification, sufficient to encrypt only a **message digest** (a hash of the original message) instead of the whole message

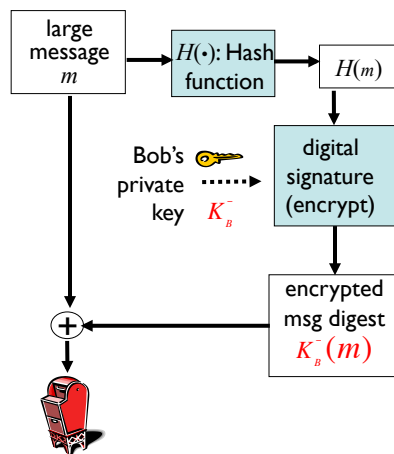
Want: a fixed-length, easy-to-compute digital “fingerprint”

Apply one-way hash function $H(\cdot)$ to m to get fixed size message digest, $H(m)$

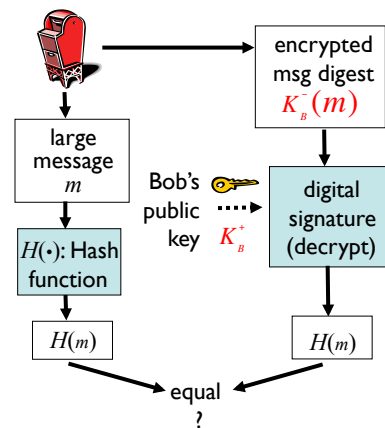


Digital Signature = Signed Message Digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



Hash Function Criteria

Required criteria of the hash function:

1. many-to-1 “compression”, but 1-1 mapping
2. produces fixed-size message digest (fingerprint), fast
3. given message digest h , computationally infeasible to find m such that $h = H(m)$

Internet checksum would not be a good hash function:

message	ASCII format	message	ASCII format
I O U 1	49 4F 55 31	I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39	0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42	9 B O B	39 42 D2 42
	B2 C1 D2 AC		B2 C1 D2 AC

different messages
but identical checksums!

Hash Function Criteria More Formally

$h = H(m)$, where h is of length k

Criteria:

- given m , it is easy to compute h
- given h , it is hard to compute m such that $H(m) = h$
- given m , it is hard (e.g., 2^k tries) to find another random message m' such that $H(m) = H(m')$
- it takes $2^{k/2}$ tries to find random m and m' , such that $H(m) = H(m')$ (**birthday attack**)

Hashed Message Authentication Code (HMAC) is an encrypted version of one-way hash function (message hashed together with a key)

Example Hash Function Algorithms

MD5 (Message Digest):

- MD4 developed by Rivest (the 'R' in RSA) in 1990, MD5 in 1992
- computes 128-bit message digest in 4-step process
- its components have been shown to be insecure

SHA-1 (Secure Hash Algorithm):

- SHA developed by the NSA (1993), revised SHA-1 in 1995
- US standard [NIST, FIPS PUB 180-1]
- produces 160-bit message digest
- SHA-256, SHA-384, SHA-512 also defined (2001)

RIPEMD-160

- developed by one of the guys who "broke" MD5 (Dobbertin) and a team of European researchers at RIPE
- produces 160-bit hash

Speed: MD5 > RIPEMD-160 > SHA-1

At Which Layer to Put Security?

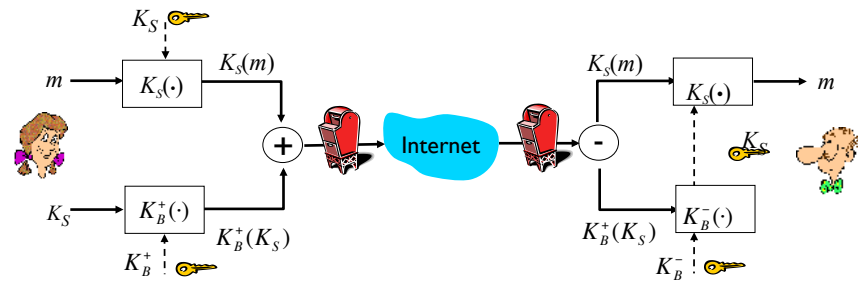
Link-oriented vs. end-to-end

Which layer?

- IPsec: Authentication Header (AH) and Encapsulating Security Payload (ESP)
- Above TCP: Secure Socket Layer (SSL) Netscape, 1994
- Application layer: secure email (PGP), ssh, web (HTTPS)

Secure e-mail: Confidentiality

Alice wants to send confidential e-mail, m , to Bob



Alice:

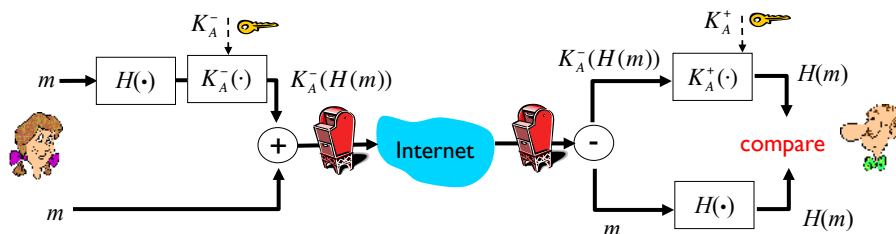
- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key $K_B^+(K_S)$
- sends both $K_S(m)$ and $K_B^+(K_S)$ to Bob

Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail: Authentication and Integrity

Alice wants to provide sender authentication and message integrity

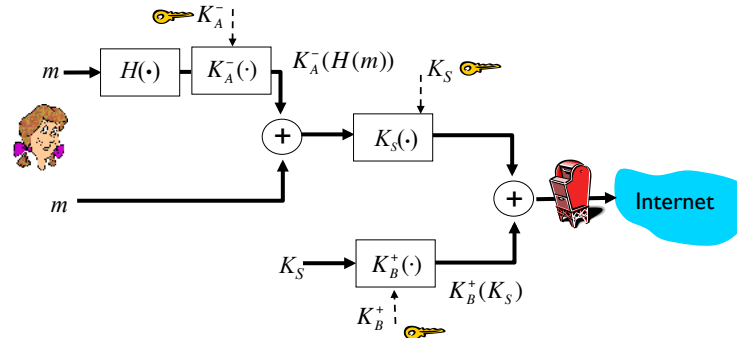


Alice digitally signs message

- sends both message (in the clear) and digital signature

Secure e-mail: Confidentiality, Authentication, and Integrity

Alice wants to provide confidentiality, sender authentication, and message integrity



Alice uses three keys: her private key, Bob's public key, and a newly created symmetric key

Pretty Good Privacy (PGP)

Internet e-mail encryption scheme, *de facto* standard

Uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described

Provides secrecy, sender authentication, data integrity

Inventor, Phil Zimmerman, was target of 3-year federal investigation

A PGP signed message:

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

Bob: IOU $100.

Sincerely yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRhhGJGhg/12EpJ
+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

SSH [RFC 4251]

Establishes a secure channel between a local and remote computer

Uses public-key cryptography to authenticate remote host and user

Provides confidentiality and data integrity with symmetric cryptography and digital signature

Authentication

- Password-based
- Public-key based
 - Public and private key pair generation using `ssh-keygen`

Secure Sockets Layer (SSL)

Transport layer security to TCP-based apps

Used between Web browsers, servers (https)

Security services:

- server authentication
- data encryption
- client authentication (optional)

Server authentication:

- SSL-enabled browser includes public keys for trusted CAs
- browser requests server certificate, issued by trusted CA, from server
- browser uses CA's public key to extract server's public key from certificate

SSL

Encrypted SSL session:

- browser generates a *symmetric session key*, encrypts it with server's public key, sends encrypted key to server
- using private key, server decrypts session key
- browser and server now share session key
- all data sent into TCP socket (by client or server) encrypted with session key
- client/browser authentication can be (but hardly ever) done with client certificates

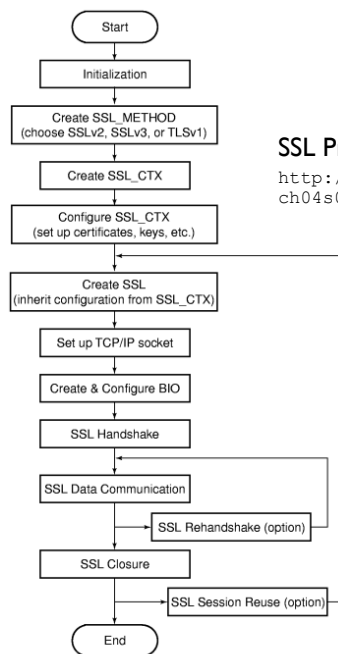
SSL: basis of IETF Transport Layer Security (TLS)

SSL can be used for non-Web applications, e.g., IMAP

SSL Programming

SSL Programming Tutorial:

http://h71000.www7.hp.com/doc/83final/ba554_90007/ch04s03.html



VM-0909A-AI

Message Digest Example

```
#include <stdio.h>
#include <openssl/evp.h>
#include <string.h>

main(int argc, char *argv[])
{
    EVP_MD_CTX mdctx;
    const EVP_MD *md;
    char mess1[] = "Test Message\n";
    char mess2[] = "Hello World\n";
    unsigned char md_value[EVP_MAX_MD_SIZE];
    unsigned int md_len, i;

    if (!argv[1]) {
        printf("Usage: mdtest digestname\n");
        exit(1);
    }

    OpenSSL_add_all_digests();

    md = EVP_get_digestbyname(argv[1]);
    if(!md) {
        printf("Unknown message digest %s\n", argv[1]);
        exit(1);
    }

    EVP_MD_CTX_init(&mdctx);
    EVP_DigestInit_ex(&mdctx, md, NULL);
    EVP_DigestUpdate(&mdctx, mess1, strlen(mess1));
    EVP_DigestUpdate(&mdctx, mess2, strlen(mess2));
    EVP_DigestFinal_ex(&mdctx, md_value, &md_len);
    EVP_MD_CTX_cleanup(&mdctx);

    printf("Digest is: ");
    for(i = 0; i < md_len; i++)
        printf("%02x", md_value[i]);
    printf("\n");
}
```

Compiling SSL (Unix)

You have to tell the linker to include the `openssl` libraries you're using, most likely `libssl` and/or `libcrypto`

- use the `-l ssl / -l crypto` command line options to `gcc` when compiling

Include files in `/usr/include/openssl/`

On *NIX machines, see `ssl(3)` and `crypto(3)`, the former deals with SSL, the latter with its implementation of cryptographic algorithms

Or try openssl.org

IPsec: Network Layer Security

Provides:

- network-layer authentication: destination host can authenticate source IP address
- network-layer secrecy: sending host encrypts the data in IP datagram

Two principal protocols:

- authentication header (AH) protocol
- encapsulation security payload (ESP) protocol
- both can be used as is with IPv6

For both AH and ESP, source, destination handshake:

- create a network-layer end-to-end logical channel called a security association (SA)
- SA sets up a shared secret between the two
- each SA unidirectional, uniquely determined by:
 - security protocol (AH or ESP)
 - source IP address
 - 32-bit connection ID

Authentication Header (AH) Protocol

Provides source authentication, data integrity, but no confidentiality/secrecy

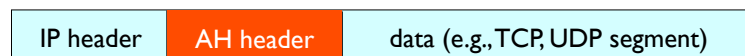
AH header inserted between IP header and data field

Protocol# 51

Intermediate routers process datagrams as usual

AH header includes:

- connection identifier
- authentication data: source-signed message digest calculated over original IP datagram
- next header field: specifies type of data (e.g., TCP, UDP, ICMP)



ESP Protocol

Provides source authentication, data integrity, and confidentiality/secretcy

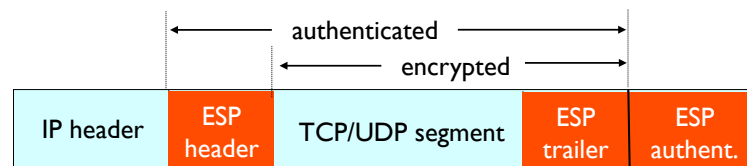
Data, ESP trailer encrypted

Next header field is in ESP trailer

ESP authentication field is similar to AH authentication field

Protocol# 50

ESP used to provide VPNs



What is a VPN?

Making a shared network look like a private network

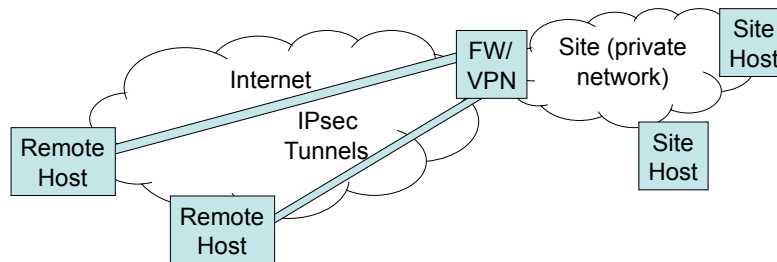
Why do this?

- VPN makes separated IP sites look like one private IP network
- private addresses and domain names (useful for authorization)
- security
- bandwidth guarantees, QoS, SLAs across ISP
- simplified network operation: ISP can do the routing for you
- building a real private network is expensive (cheaper to use shared resources rather than to have dedicated resources)

End-to-end VPNs

Solves problem of how to connect remote hosts to a firewalled network

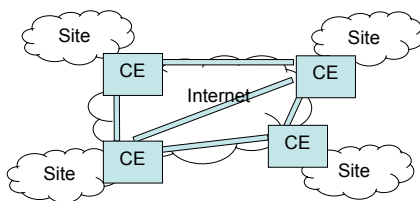
- commonly used for roaming
- security and private addresses benefits only
- not simplicity or QoS benefits



Network VPNs

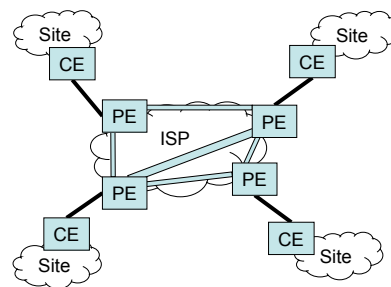
Customer based:

- customer buys own equipment, configures IPsec tunnels over the global Internet, manages addressing and routing
- ISP plays no role
- customer has more control over security and ISP choices, but requires skills



Provider based:

- provider manages all the complexity of the VPN
- customer simply connects to the provider equipment

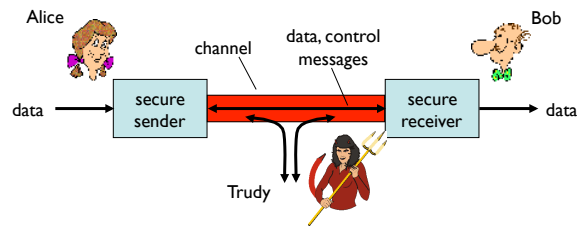


Security Attacks

Cast of characters: Alice, Bob, Trudy, well-known characters in network security world

Bob, Alice want to communicate “securely”

Trudy (intruder) may intercept, delete, add messages



Two types of security attacks:

1. against information content: secrecy, integrity, authentication, authorization, privacy, anonymity
2. against the infrastructure: system intrusion, denial of service

Denial of Service (DoS) Attack

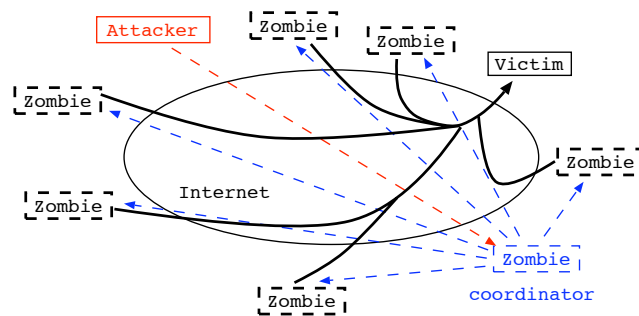
An attacker inundates its victim with otherwise legitimate service requests or traffic such that victim's resources are overloaded and overwhelmed to the point that the victim can perform no useful work and legitimate users are denied service

Examples:

- SYN flood: send lots of TCP SYN to fill up victim's listen queue
- Smurf attack: broadcast ICMP echo requests with the source address spoofed to victim's
- IP fragmentation
- TCP reassembly

Distributed Denial of Service (DDoS) Attack

Attacker commandeers systems (*zombies*) distributed across the Internet to send correlated service requests or traffic to the victim to overload the victim



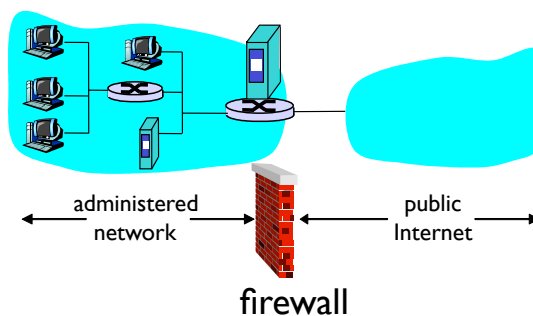
The Making of Zombies

1. Zombies, the commandeered systems, are usually taken over by exploiting program bugs or backdoors left by the programmers
2. Zombie intrusion done by sending harmless looking legitimate code or data containing hidden malicious code (Trojan Horse) to the vulnerable candidate
3. Once a host is taken over and made a zombie, the malicious code will run as a background process that performs the actual attack

Firewalls

A barrier that restricts the free flow of data between the “inside” and the “outside”; isolates organization’s internal net from the larger Internet, allowing some packets to pass through, blocking others

Intrusion detection: border checkpoints monitor traffic for known attack patterns



Firewalls

Design criteria:

- all traffic must pass through the firewall
- only authorized traffic will be allowed to pass
- authorization is a local security policy
- the firewall itself must be immune to penetration

Goals:

- prevent illegal modification or access of internal data
- allow only authorized access to inside network (set of authenticated users/hosts)
- prevent host subjugation for DDoS attack

Firewalls

Advantages of firewall vs. security hardening of each host:

- more convenient: border checkpoint
- more secure: a firewall is not a general purpose machine
- administrator of firewall is more security conscious (hopefully)
- firewall has restricted access, usually can be made less convenient to use (e.g., requiring one-time passcode)
- central point of control for mail, ftp, web administration
- can also be used internally

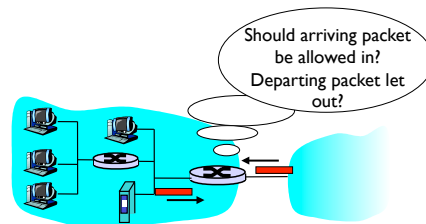
Types of Firewall

Three types of firewall:

1. packet filter
2. circuit-level gateway
3. application-level gateway

Packet-filter:

- drops packet not matching any given pattern of packet header
- filter usually constructed out of src addr, dst addr, src port, dst port, and TCP flag fields
- allows for wild-card value



Packet Filter Examples

A	action	ourhost	port	theirhost	port	comment	
	block	*	*	SPIGOT	*	we don't trust these people	
	allow	OUR-GW	25	*	*	connection to our SMTP port	
B	action	ourhost	port	theirhost	port	comment	
	block	*	*	*	*	default	
C	action	ourhost	port	theirhost	port	comment	
	allow	*	*	*	25	connection to their SMTP port	
D	action	src	port	dest	port	flags	comment
	allow	{our hosts}	*	*	25		our packets to their SMTP port
	allow	*	25	*	*	ACK	their replies
E	action	src	port	dest	port	flags	comment
	allow	{our hosts}	*	*	*		our outgoing calls
	allow	*	*	*	*	ACK	replies to our calls
	allow	*	*	*	>1024		traffic to nonservers

Cheswick & Bellon

Gateways

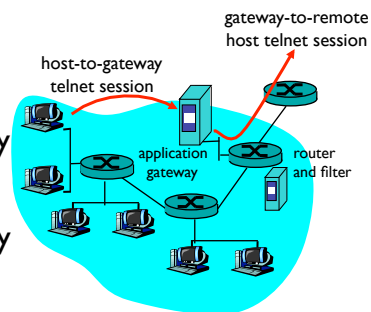
Circuit-level gateways: force every connection to go through the gateway

Circuit-level gateways usually do not look at the bytes sent, it serves mainly for logging of allowed connections

Example:

1. require all telnet users to telnet through gateway
2. for authorized users, gateway sets up telnet connection to dest host
3. gateway relays data between 2 connections
4. router filter blocks all telnet connections not originating from gateway

Application-level gateways parse traffic and allow only known operations



Limitations of Firewalls and Gateways

- Filter specification language could make certain policies hard to express
- Filters interaction could give rise to unintended policies (bugs in filter specifications)
- Filter specification for protocols without fixed port number difficult
- IP spoofing: router can't know if data "really" comes from claimed source
- Not very effective against connectionless UDP (filters often use all or nothing policy for UDP)
- If multiple app's need special treatment, each needs its own app gateway
- Client software must know how to contact gateway, e.g., must set IP address of proxy in Web browser
- Packet-filter and circuit-level gateways can be (ab)used for tunneling under the firewall

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)
- authentication
- message integrity
- key distribution

.... used in many different security scenarios

- secure email (PGP)
- secure transport (SSL)
- IPsec
- 802.11

Network security is an ongoing arms race

- breaking things is fun to some
- ad hoc approaches

And then there's unwanted traffic: spam, etc.