Security and privacy in networks

Damien Saucez
Inria Sophia Antipolis

Université d’Avignon et des Pays de Vaucluse, CERI - Sécurité dans les réseaux - November 2016
Contact information

- Damien Saucez
  - Email: damien.saucez@inria.fr
  - Mobile: +33 6 95.06.46.11
  - Phone: +33 4 89.73.24.18
Table of Content

1. Reminders
2. Threats by the example
3. The basics of security
4. Securing communications
5. Overlay networking
6. Privacy
References

  - slides inspired from this book
- A. Legout, Peer-to-Peer Applications From BitTorrent to Privacy, Inria
  - some slides inspired from this course
Reminders
Generalities
Network

- Network:
  - set of nodes (e.g., hosts, routers) exchanging information and interconnected with links
  - Communication rules in a network are specified by a set of protocols (e.g., IEEE 802.3, IP, OSPF, BGP)

- Example of networks:
  - Telephone System
  - Mobile network
  - Television, radio
  - Internet, LAN
Network topologies

- Bus
- Star
- Tree
- Ring
- Full-Mesh
Transmission modes

- Unicast (Point-to-Point)
  - one sender
  - one receiver
  - example: telephone
  - the variant where the receiver is taken in a set of possible receivers is called anycast
    - anycast helps scalability
Transmission modes (cont.)

- Multicast (Point-to-Multipoint)
  - one sender
  - a group of receivers
  - every member of the group receives the same information
  - example: videoconference
  - when the information is sent to every node, the term broadcast is used (e.g., Terrestrial television)
Digital networking communications modes

- Circuit switching
  - before transmitting information, a dedicated circuit is established from the source to the destination nodes
  - the information is transmitted through its dedicated circuit that guarantees the bandwidth during the whole communication
  - each intermediate node knows how to forward information received on circuits crossing itself
  - example: 19th century telephone system
Circuit switching example

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>E</td>
</tr>
<tr>
<td>Green</td>
<td>S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>SE</td>
</tr>
<tr>
<td>Blue</td>
<td>SE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>NE</td>
</tr>
<tr>
<td>Green</td>
<td>SE</td>
</tr>
<tr>
<td>Blue</td>
<td>NE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>NE</td>
</tr>
<tr>
<td>Green</td>
<td>NE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>NE</td>
</tr>
<tr>
<td>Green</td>
<td>NE</td>
</tr>
<tr>
<td>Blue</td>
<td>NE</td>
</tr>
</tbody>
</table>
Digital networking communications modes (cont.)

- Packet switching
  - data is divided in packets of information containing
    - a piece of data
    - the address of the source node
    - the address of the destination node
  - packets are transmitted on the network independently of each other
  - each intermediate node knows how to forward information to each destination
  - example: IP, Internet
### Packet Switching Example

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SW</td>
</tr>
<tr>
<td>B</td>
<td>SW</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>W</td>
</tr>
<tr>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td>C</td>
<td>SE</td>
</tr>
<tr>
<td>D</td>
<td>SE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>NW</td>
</tr>
<tr>
<td>B</td>
<td>SW</td>
</tr>
<tr>
<td>C</td>
<td>NE</td>
</tr>
<tr>
<td>D</td>
<td>SE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>NW</td>
</tr>
<tr>
<td>B</td>
<td>NW</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>NW</td>
</tr>
<tr>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td>C</td>
<td>NE</td>
</tr>
<tr>
<td>D</td>
<td>NE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Send to</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>W</td>
</tr>
<tr>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>
Layered model

- Network systems are complex
- dividing the functionality helps reasoning on them
- Divide network functionalities into layers
  - Layer $i$ provides services to layer $i+1$
  - Layer $i$ relies on services provided by layer $i-1$
<table>
<thead>
<tr>
<th>Layers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Exchange of useful information (Service Data Unit) between applications relying on the transport layer hiding the network complexity (e.g., HTTP)</td>
</tr>
<tr>
<td>Transport</td>
<td>Provide a service to (reliably) exchange data between hosts with segments (e.g., TCP, UDP)</td>
</tr>
<tr>
<td>Network</td>
<td>Provide a service to exchange packets of information between hosts that can be arbitrarily distant (e.g., IP)</td>
</tr>
<tr>
<td>Datalink</td>
<td>Provide a service to exchange structured group of bits called frames (e.g., Ethernet)</td>
</tr>
<tr>
<td>Physical</td>
<td>Transmit bits between two physically connected devices (e.g., Manchester)</td>
</tr>
</tbody>
</table>

Physical transmission medium (e.g., UTP)
Layer of networking devices

- Physical Layer
- Data Link Layer
- Network Layer
- Transport Layer
- Application Layer

- Host
- Router
- Bridge
- Repeater
Middleboxes

- The original TCP/IP architecture is only composed of hosts and routers
- Modern networks contain devices that
  - process (e.g., proxies)
  - analyze (e.g., firewall)
  - modify (e.g., NAT)
- Middleboxes can work at any layer or even be cross layer
Middleboxes are everywhere

- In enterprise networks [SHC+12]

- In ISP networks [HRN+11]
  - very likely that your packet will be touched by a middlebox before reaching its destination
  - Middleboxes limit deployment of new protocols in the Internet
  - Middleboxes can be used against user interests

Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.
Naming and addressing
Name and addresses in the Internet

- DNS Names identify hosts
- IP addresses uniquely identify host interfaces
  - `nslookup example.com`
    Server: 138.96.0.10
    Address: 138.96.0.10#53

    Non-authoritative answer:
    Name: example.com
    Address: 192.0.43.10

- Ethernet address identifies network adapters in a collision domain
  - `arp -na`
    ? (138.96.192.3) at 0:50:56:88:0:0 on en1 ifscope [ethernet]
    ? (138.96.192.250) at 0:1e:4a:e0:9e:0 on en1 ifscope [ethernet]
    ? (138.96.193.164) at 0:23:df:aa:cc:4c on en1 ifscope [ethernet]
    ...

- Names and addresses may be hierarchically organized
Hierarchical naming/addressing

- Objectives: ensure uniqueness of names/addresses and provide naming/addressing scalability
- Flat: probe all the other naming/addressing authorities before choosing a name/address
  - doesn’t scale
  - not robust to network partition
- Hierarchy: carve up set of possible names/address (i.e., the name/address space) into mutually exclusive portions
Addressing in Ethernet

- Objective: determine the origin and destination of a frame within a collision domain
- Every Ethernet network adapter is assigned a unique datalink layer address encoded on 48 bits
- Every frame is transmitted to all network adapters of the collision domain
  - but only the network adapter with the address corresponding to the destination address of the frame accepts it
Addressing in IP

- Objective: determine the origin and destination of a packet in the Internet
- Every host interface has its own IP address
  - routers have multiple interfaces, each with its own IP address
  - the IP address determines the topological position of the interface
- Current version of IP is version 4 (IPv4)
  - addresses are encoded on 32 bits, fixed length
- 4 billions addresses were a lot... in 1981, but today it becomes too short for 1 billion hosts [ISC]
- IP version 6 (IPv6) starts to be deployed
  - addresses are encoded on 128 bits, fixed length*
IP address structure

- Addresses are separated in two parts
  - network number: identifies the network the address belongs to
  - local address: identifies the interface of the host in the network
    - all bits = 0: network address
    - all bits = 1: broadcast address
- Addresses are aggregated according to the network number
  - routing and packet forwarding are based on the network number only, the local address is ignored
IP address example

- **IP**: 192.0.2.1
  - \[11000000 00000000 00000010 00000001\]
- **Subnet**: 255.255.255.0
  - \[11111111 11111111 11111111 00000000\]
IP address example

- IP: 192.0.2.1

What part is for hosts? what part is for the network?

- 11000000 00000000 00000010 00000001
- 11111111 11111111 11111111 00000000
IP address example (cont.)
IP address example (cont.)

IP: 11000000 00000000 00000010 00000001
IP address example (cont.)

IP: 11000000 00000000 00000010 00000001

Subnet: & 11111111 11111111 11111111 00000000
IP address example (cont.)

IP: 11000000 00000000 00000010 00000000

Subnet: & 11111111 11111111 11111111 00000000

11000000 00000000 00000010 00000000

network  hosts
Classless InterDomain Routing (CIDR)

- No predetermined separation position between network number and local address with CIDR
  - number of bits allocated for the network number may vary from 0 to 32 bits
  - the address contains no information about the separation position
  - Routers determine the network number by using longest-prefix matching
- Notation $a.b.c.d/n$
  - $a.b.c.d$ is the address
  - $n$ is the number of bits assigned to the network number
CIDR (cont.)

- An address matches a route if both share the same prefix
  - 0.0.0.0/0 is the default route matched by every addresses
- With CIDR, an address can match several routes
  - 192.0.2.1 matches 128.0.0.0/1, but also 192.0.2.0/24 or 0.0.0.0/0
- Longest prefix matching is used to determine the route that has the longest prefix in common with the address
- Typically implemented with a trie
Longest prefix matching with a trie

- Routes are inserted in a trie, route prefixes being node keys.
- The key of a node is a prefix of the key of all of its children, recursively;
  - siblings cannot be prefixes
- The binary tree is descended, starting from the root, following the children with the key that is a prefix of the address to match.
- The descend ends when no children has a key prefixing the address to match.
  - the route corresponding to the node where the descent stopped is the best matching route.
Longest prefix matching with a trie (examples)
Longest prefix matching with a trie (examples)

* 11000000 00000000 00000010 00000001 (192.0.2.1)

(0.0.0.0/0)

00001010 (10.0.0.0/8)

11000000 00000000 00000010 00000001 (192.0.2.1)

(128.0.0.0/1)

10001010 01100000 110010 (138.96.200.0/22)

11000000 00000000 00000010 (192.0.2.0/24)
Longest prefix matching with a trie (examples)
Longest prefix matching with a trie (examples)
Longest prefix matching with a trie (examples)

* → 11000000 00000000 00000010 00000001 (192.0.2.1)

(0.0.0.0/0) → 00001010 (10.0.0.0/8)

(10001010 01100000 110010 00000001 (192.0.2.1)

(128.0.0.0/1) → 00000100 11000000 00000000 00000010 (138.96.200.0/22)

(192.0.2.0/24) → 11000000 00000000 00000010 00000001 (192.0.2.1)

Best match 192.0.2.0/24
Longest prefix matching with a trie (examples)

*  
(0.0.0.0/0)  
  
00001010
(10.0.0.0/8)  
  
10001010 01100000 110010
(138.96.200.0/22)  
  
11000000 00000000 00000010
(192.0.2.0/24)

(128.0.0.0/1)
Longest prefix matching with a trie (examples)
Longest prefix matching with a trie (examples)
Longest prefix matching with a trie (examples)
IP to Ethernet Address

- To put an IP packet over an Ethernet frame, its IP addresses must be resolved into Ethernet addresses.

- Protocol used:
  - Address Resolution Protocol (ARP) in IPv4
  - Neighbor Discovery Protocol (NDP) in IPv6
ARP

- ARP is used to get datalink layer address of a machine on the local subnet
- Broadcast an ARP request frame on the local subnet for the IP address to resolve
  - destination address: FF:FF:FF:FF:FF:FF (broadcast)
  - source address: Ethernet address of the network adapter that issued the ARP request
- The host (or a proxy) that owns the address replies with an ARP response frame
  - destination address: Ethernet address of the requester’s network adapter
  - source address: Ethernet address of the address’s owner’s (or proxy) network adapter
- Every network device is required to listen for ARP requests and replies on its network adapters
- Optimizations
  - replies are stored in an ARP cache to avoid that every single packet results in ARP request/response
    - cached for a limited duration as host can change their IP address
  - ARP request message contains the IP address of the origin of the frame
    - destination (or any hosts in the local subnet) can learn the IP/Ethernet mapping for free
ARP example

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.3
Ethernet: c

IP: 192.0.2.4
Ethernet: d
ARP example
ARP example

IP source: 192.0.2.2  IP destination: 192.0.2.3

<table>
<thead>
<tr>
<th>IP</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.2.5</td>
<td>a</td>
</tr>
<tr>
<td>192.0.2.2</td>
<td>b</td>
</tr>
<tr>
<td>192.0.2.3</td>
<td>c</td>
</tr>
<tr>
<td>192.0.2.4</td>
<td>d</td>
</tr>
</tbody>
</table>

who-has 192.0.2.3?  (I am 192.0.2.2)
ARP example

IP source: 192.0.2.2  IP destination: 192.0.2.3

IP: 192.0.2.5
Eth: a

IP: 192.0.2.2
Eth: b

IP: 192.0.2.3
Eth: c

IP: 192.0.2.4
Eth: d

who-has 192.0.2.3?      (I am 192.0.2.2)

I am 192.0.2.3

I am 192.0.2.3
ARP example

ARPA ARP example

<table>
<thead>
<tr>
<th>IP source: 192.0.2.2</th>
<th>IP destination: 192.0.2.3</th>
</tr>
</thead>
</table>

IP: 192.0.2.5  Ethernet: a
IP: 192.0.2.2  Ethernet: b
IP: 192.0.2.3  Ethernet: c
IP: 192.0.2.4  Ethernet: d

who-has 192.0.2.3? (I am 192.0.2.2)
I am 192.0.2.3

Ethernet source: b  Ethernet destination: c  IP source: 192.0.2.2  IP destination: 192.0.2.3
ARP example (router)

gateway: 192.0.2.1/24

gateway: 203.0.113.1/24

IP: 192.0.2.5 Ethernet: a

IP: 192.0.2.2 Ethernet: b

IP: 192.0.2.1 Ethernet: f

IP: 203.0.113.1 Ethernet: d

IP: 203.0.113.2 Ethernet: e
ARP example (router)

gateway: 192.0.2.1/24

IP source: 192.0.2.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

IP: 203.0.113.2
Ethernet: e

gateway: 203.0.113.1/24

IP destination: 203.0.113.2
ARP example (router)

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 203.0.113.2
Ethernet: e

IP: 203.0.113.1
Ethernet: d

who-has 192.0.2.1? (I am 192.0.2.2)

gateway: 192.0.2.1/24
gateway: 203.0.113.1/24
ARP example (router)

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

IP: 203.0.113.2
Ethernet: e

who-has 192.0.2.1? (I am 192.0.2.2)
I am 192.0.2.1

gateway: 192.0.2.1/24

IP: 192.0.2.1
Ethernet: f

gateway: 203.0.113.1/24

IP: 203.0.113.1
Ethernet: d

who-has 192.0.2.1? (I am 192.0.2.2)
I am 192.0.2.1

IP: 203.0.113.2
Ethernet: e

who-has 192.0.2.1? (I am 192.0.2.2)
I am 192.0.2.1
ARP example (router)

gateway: 192.0.2.1/24

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 203.0.113.2
Ethernet: e

IP source: 192.0.2.2
IP destination: 203.0.113.2

I am 192.0.2.2

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.1
Ethernet: d

IP: 203.0.113.2
Ethernet: e

IP source: 192.0.2.2
IP destination: 203.0.113.2

who-has 192.0.2.1? (I am 192.0.2.2)
ARP example (router)

IP source: 192.0.2.2  IP destination: 203.0.113.2

gateway: 192.0.2.1/24

who-has 192.0.2.1? (I am 192.0.2.2)

IP: 192.0.2.5  Ethernet: a

I am 192.0.2.1

IP: 192.0.2.2  Ethernet: b

IP: 192.0.2.1  Ethernet: f

who-has 203.0.113.2? (I am 203.0.113.1)

IP: 203.0.113.1  Ethernet: d

IP: 203.0.113.2  Ethernet: e

IP source: 192.0.2.2  IP destination: 203.0.113.2

gateway: 203.0.113.1/24
ARP example (router)

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 203.0.113.2
Ethernet: e

IP: 203.0.113.1
Ethernet: d

IP source: 192.0.2.2
IP destination: 203.0.113.2

who-has 192.0.2.1? (I am 192.0.2.2)

I am 192.0.2.2

who-has 203.0.113.2? (I am 203.0.113.1)

I am 203.0.113.2

gateway: 192.0.2.1/24

gateway: 203.0.113.1/24
ARP example (router)

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2

IP source: 192.0.2.2
IP destination: 203.0.113.2
Dynamic address configuration

- Allow a set of hosts to share a pool of IP address
- Two approaches
  - stateless auto-configuration
    - no infrastructure necessary
  - Dynamic Host Configuration Protocol (DHCP)
    - hosts query a DHCP server to obtain their configuration
- Advantages
  - less address wastage: a host can use the address of another hosts when it is not connected
  - improves flexibility and reduces the risk of configuration error as no manual operation is necessary
Stateless auto-configuration

- When a host connects to the network:
  1. The host choses an address randomly in 169.254/16 (not globally routable)
  2. Sends an ARP request for the chosen address
  3. If an ARP reply is received (another host already uses the address)
     - restart from point 1
  4. Otherwise, the address the address is not used by another host and the host can use it safely

- Auto-configuration is used only for communications within the same network
  - In IPv6, hosts can auto-configure their globally routable addresses and discover network services (e.g., routers, DNS...)
Dynamic Host Configuration Protocol (DHCP)

- When a host connects to the network, it broadcasts a DHCP discovery datagram.
- Any DHCP server that receives such a message replies with a DHCP offer datagram that contains an offer of IP address.
- The host picks one offer and broadcasts a DHCP request message to announce the offers it selected.
- The selected DHCP server assigns the address to the host and sends it back a DHCP acknowledgment that confirms the lease of the address and gives additional parameters such as the lease time, the IP address of the default gateway, or the IP address of the DNS servers.
  - When the lease time is elapsed, the address is released and made available for other hosts.
- The other DHCP servers withdraw their offers.
Naming

- Objective: provide a mean for human to easily identify (and remember) hosts
- Hosts receive textual names easy to remember but long and of variable size (e.g., goo.gl, www.example.org, 3.141592653589793238462643383279502884197169399375105820974944592.com...)
  - wastes space to carry them in packet headers
  - hard to parse
- Address are shorter and easy to process by hosts
- Indirection
  - multiple names may point to the same address
  - upon address change, only the resolution table has to be updated
Hierarchical naming

- Simplifies distributed naming/addressing
  - level $i$ deals only with level $i+1$
- Global uniqueness is guaranteed
  - level $i$ ensures uniqueness at level $i+1$
- Scales arbitrarily
  - level $i+1$ does not influence level $i-1$
Iterative resolution

- The resolver learns the hierarchy
- Responses can be cached to avoid querying twice the same server
Iterative resolution

The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- Responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

The resolver learns the hierarchy

- responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

The resolver learns the hierarchy
responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Iterative resolution

- The resolver learns the hierarchy
- responses can be cached to avoid querying twice the same server
Transport
Transport of data between hosts

- Transport layer provides an end-to-end communication service
- Applications just deal with stream of bytes
- Most popular protocols:
  - UDP: connection-less, non reliable
  - TCP: connection-full, reliable
TCP connection establishment
TCP connection establishment

A

B

LISTEN
TCP connection establishment

A

SYN, sequence number=123

B

LISTEN
TCP connection establishment

A

SYN-SENT

SYN, sequence number=123

B

LISTEN

SYN-RECEIVED
TCP connection establishment

A

SYN-SENT

B

LISTEN

SYN, sequence number=123

SYN+ACK, sequence number=789,
acknowledgment number=124

SYN-RECEIVED
TCP connection establishment

A

SYN-SENT

SYN, sequence number=123

B

LISTEN

SYN-RECEIVED

SYN+ACK, sequence number=789,
acknowledgment number=124

ACK, acknowledgment number=790
TCP connection establishment

A

SYN-SENT

SYN, sequence number=123

SYN+ACK, sequence number=789, acknowledgment number=124

ACK, acknowledgment number=790

ESTABLISHED

B

LISTEN

SYN-RECEIVED

ESTABLISHED
TCP data transfer

A

window size = 1500B

sent 1000 to 1499

sent 1500 to 1999

sent 2000 to 2499

waiting to send the rest

sent 2500 to …

…

sequence number=1000

ACK, acknowledgment number=1500

ACK, acknowledgment number=2000

ACK, acknowledgment number=2500

sequence number=2500

B

ready to receive data sequenced between 1000 and 2499

ready to receive data sequenced between 1500 to 2999

ready to receive data sequenced between 2000 to 3499

ready to receive data sequenced between 2500 to 3999
TCP connection termination

A

B
TCP connection termination

A
ESTABLISHED

B
ESTABLISHED

46
TCP connection termination

A
ESTABLISHED

B
FIN, sequence number = 567
ESTABLISHED
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN, sequence number = 567

B

ESTABLISHED
TCP connection termination

A

FIN-WAIT-1

FIN, sequence number = 567

ACK, acknowledgment number=568

B

ESTABLISHED
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN, sequence number = 567

ACK, acknowledgment number = 568

B

ESTABLISHED

CLOSE-WAIT
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN-WAIT-2

B

FIN, sequence number = 567

ACK, acknowledgment number=568

ESTABLISHED

CLOSE-WAIT
TCP connection termination

**A**
- ESTABLISHED
- FIN-WAIT-1
- ACK, acknowledgment number = 568
- FIN-WAIT-2
- FIN, sequence number = 987

**B**
- ESTABLISHED
- CLOSE-WAIT
- FIN, sequence number = 567
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN, sequence number = 567

ACK, acknowledgment number = 568

FIN-WAIT-2

FIN, sequence number = 987

B

ESTABLISHED

CLOSE-WAIT

LAST-ACK
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN-WAIT-2

TIME-WAIT

FIN, sequence number = 567

ACK, acknowledgment number = 568

FIN, sequence number = 987

B

ESTABLISHED

CLOSE-WAIT

LAST-ACK
TCP connection termination

A

ESTABLISHED
FIN-WAIT-1
ACK, acknowledgment number=568
FIN-WAIT-2
FIN, sequence number = 987
TIME-WAIT
ACK, acknowledgment number=988

B

ESTABLISHED
CLOSE-WAIT
LAST-ACK
TCP connection termination

A

ESTABLISHED

FIN-WAIT-1

FIN-WAIT-2

TIME-WAIT

CLOSED

FIN, sequence number = 567

ACK, acknowledgment number=568

B

ESTABLISHED

CLOSE-WAIT

LAST-ACK

FIN, sequence number = 987

ACK, acknowledgment number=988

ACK, acknowledgment number=568
TCP connection termination

A

ESTABLISHED
FIN-WAIT-1
FIN-WAIT-2
TIME-WAIT
CLOSED

FIN, sequence number = 567
ACK, acknowledgment number = 568
FIN, sequence number = 987
ACK, acknowledgment number = 988

B

ESTABLISHED
CLOSE-WAIT
LAST-ACK
CLOSED
Threats by the example
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is f

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is f

who-has 192.0.2.2? (I am 192.0.2.1)
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is a

who-has 192.0.2.2? (I am 192.0.2.1)
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is a

IP: 192.0.2.5
Ethernet: a

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

who-has 192.0.2.2? (I am 192.0.2.1)

I am 192.0.2.2
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is a

| IP source: 192.0.2.2 | IP destination: 203.0.113.2 |

IP: 192.0.2.5
Ethernet: a

who-has 192.0.2.2? (I am 192.0.2.1)

I am 192.0.2.2

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

IP: 192.0.2.2
Ethernet: b
ARP poisoning

0.0.0.0/0 via 192.0.2.1
192.0.2.1 is a

IP source: 192.0.2.2 | IP destination: 203.0.113.2

IP: 192.0.2.5
Ethernet: a

who-has 192.0.2.2? (I am 192.0.2.1)

I am 192.0.2.2

IP: 192.0.2.2
Ethernet: b

IP: 192.0.2.1
Ethernet: f

Ethernet source: b | Ethernet destination: a
IP source: 192.0.2.2 | IP destination: 203.0.113.2
Why does it work?
Why does it work?

- Conceptual vulnerability
  - using non-requested information as ground truth is dangerous
  - using non-authenticated information is dangerous
DNS cache poisoning
DNS cache poisoning

Query: rnd.example.org

resolver

192.0.2.1
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Query: rnd.example.org
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02
DNS cache poisoning

Query: rnd.example.org, ID: 0x02
example.org. @{192.0.2.1}

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

...
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0xff

Query: rnd.example.org

Example.org. @192.0.2.1
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02

Query: rnd.example.org

Response: rnd.example.org, ask example.org. @203.0.113.2: ID: 0x02

Response: rnd.example.org, ask example.org. @203.0.113.2: ID: 0x01

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0xffff
DNS cache poisoning

Query: rnd.example.org, ID: 0x02
Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02

Query: rnd.example.org
Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01

Response: rnd.example.org, ask example.org. @203.0.113.2: ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0xff

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x02
DNS cache poisoning

Query: rnd.example.org, ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Query: rnd.example.org
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

...
DNS cache poisoning

Query: rnd.example.org, ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x01

Query: rnd.example.org
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

Query: rnd.example.org, ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

Query: bank.example.org

Response: rnd.example.org, ask example.org. @{203.0.113.2}: ID: 0x02
Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0x02

Response: rnd.example.org, ask example.org. @{192.0.2.1}: ID: 0xff

192.0.2.1
resolver
DNS cache poisoning

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0x01

Query: bank.example.org

Response: bank.example.org

Response: rnd.example.org, ask example.org. @203.0.113.2: ID: 0x02

Response: rnd.example.org, ask example.org. @192.0.2.1: ID: 0xff

Query: bank.example.org

Response: bank.example.org
DNS cache poisoning

Query: bank.example.org

Response: rnd.example.org, ask example.org. @\{192.0.2.1\}: ID: 0x02

Response: bank.example.org

Query: bank.example.org

Response: rnd.example.org, ask example.org. @\{203.0.113.2\}: ID: 0x02

Query: rnd.example.org, ID: 0x02

Response: rnd.example.org, ask example.org. @\{192.0.2.1\}: ID: 0x01

Response: rnd.example.org, ask example.org. @\{192.0.2.1\}: ID: 0x02

Response: rnd.example.org, ask example.org. @\{192.0.2.1\}: ID: 0xff
Why does it work?
Why does it work?

- Birthday paradox
  - probability that \( n \) elements uniformly picked from the finite set \( T \) is

\[
p(n) = 1 - \frac{|T|!}{(|T| - n) \cdot |T|^n}
\]

- Relying solely on transaction ID is dangerous
  - particularly when IDs are small (16 bits in DNS)
DNS Distributed Denial of Service (DDoS)

- Attacks against Dyn DNS infrastructure
- Two bursts: 2016-10-21 11:10 UTC - 13:20 UTC; 15:50 UTC - 20:30 UTC
- Not usual DDoS
  - many more addresses than usual, non spoofed (between 40k and 100k addresses)

Why does it work?

- Attacks performed via a Mirai-based botnet
- IoT devices
- End-to-End principle
  - maximizes the intelligence at the edge
  - network avoids making decisions
- What if the edge is “bad”?
YouTube Hijacking

- **BBC Breaking news:** A router problem made YouTube inaccessible for many

- **RIPE NIS:** “On Sunday, 24 February 2008, Pakistan Telecom (AS17557) started an unauthorised announcement of the prefix 208.65.153.0/24. One of Pakistan Telecom's upstream providers, PCCW Global (AS3491) forwarded this announcement to the rest of the Internet, which resulted in the hijacking of YouTube traffic on a global scale”

YouTube Hijacking (contd.)
YouTube Hijacking (contd.)

- Before, during and after Sunday, 24 February 2008: AS36561 (YouTube) announces 208.65.152.0/22.
YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces 208.65.152.0/22.

- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.
YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces 208.65.152.0/22.

- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.

- **Sunday, 24 February 2008, 20:07 (UTC)**: AS36561 (YouTube) starts announcing 208.65.153.0/24. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube’s traffic.
YouTube Hijacking (contd.)

- **Before, during and after Sunday, 24 February 2008**: AS36561 (YouTube) announces 208.65.152.0/22.

- **Sunday, 24 February 2008, 18:47 (UTC)**: AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.

- **Sunday, 24 February 2008, 20:07 (UTC)**: AS36561 (YouTube) starts announcing 208.65.153.0/24. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.

- **Sunday, 24 February 2008, 20:18 (UTC)**: AS36561 (YouTube) starts announcing 208.65.153.128/25 and 208.65.153.0/25. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.
YouTube Hijacking (contd.)

- Before, during and after Sunday, 24 February 2008: AS36561 (YouTube) announces 208.65.152.0/22.

- Sunday, 24 February 2008, 18:47 (UTC): AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.


- Sunday, 24 February 2008, 20:18 (UTC): AS36561 (YouTube) starts announcing 208.65.153.128/25 and 208.65.153.0/25. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.

- Sunday, 24 February 2008, 20:51 (UTC): All prefix announcements, including the hijacked /24 which was originated by AS17557 (Pakistan Telecom) via AS3491 (PCCW Global), are seen prepended by another 17557. The longer AS path means that more routers prefer the announcement originated by YouTube.

YouTube Hijacking
(contd.)

- Before, during and after Sunday, 24 February 2008: AS36561 (YouTube) announces 208.65.152.0/22.

- **Sunday, 24 February 2008, 18:47 (UTC):** AS17557 (Pakistan Telecom) starts announcing 208.65.153.0/24. AS3491 (PCCW Global) propagates the announcement. Routers around the world receive the announcement, and YouTube traffic is redirected to Pakistan.

- **Sunday, 24 February 2008, 20:07 (UTC):** AS36561 (YouTube) starts announcing 208.65.153.0/24. [...] BGP decision process means that AS17557 (Pakistan Telecom) continues to attract some of YouTube's traffic.

- **Sunday, 24 February 2008, 20:18 (UTC):** AS36561 (YouTube) starts announcing 208.65.153.128/25 and 208.65.153.0/25. Because of the longest prefix match rule, every router that receives these announcements will send the traffic to YouTube.

- **Sunday, 24 February 2008, 20:51 (UTC):** All prefix announcements, including the hijacked /24 which was originated by AS17557 (Pakistan Telecom) via AS3491 (PCCW Global), are seen **prepended by another 17557.** The longer AS path means that more routers prefer the announcement originated by YouTube.

- **Sunday, 24 February 2008, 21:01 (UTC):** AS3491 (PCCW Global) **withdraws all prefixes originated by AS17557** (Pakistan Telecom), thus stopping the hijack of 208.65.153.0/24. Note that AS17557 was not completely disconnected by AS3491. Prefixes originated by other Pakistani ASs were still announced by AS17557 through AS3491.
Why does it work?
Why does it work?

- Any AS can claim to be the originator of a prefix (i.e., she hijacks the prefix)
- To protect against that, only the import filters can be used
  - rely on databases that are not so accurate
- A not secure global routing system is a major threat against freedom
TCP session hijacking

window size = 1500B

Client

sent 1000 to 1023

sequence number=1000, data="ls"

ACK, acknowledgment number=1024

sequence number=7568, data="www"

ACK, acknowledgment number=1024

ACK, acknowledgment number=7599

Telnet server
TCP session hijacking

Client

- sent 1000 to 1023
- sequence number=1000, data="ls"
- sequence number=7568, data="www"
- ACK, acknowledgment number=1024
- sequence number=7600, data=""
- ACK, acknowledgment number=1096

window size = 1500B

Telnet server

- sequence number=1024, data="rm -rf /"
- sequence number=7599

57
Why does it work?
Why does it work?

- If the attacker can
  - guess the initial sequence number
  - guess actions from the sender
- then easy to guess a sequence number that will be accepted by the receiver
The basics of security
Security threats

- Intrusion
  - an attacker gains remote access to some resources that are normally denied to her
    - e.g., steal processing power, botnets
- Eavesdropping
  - an attacker collects traffic of a target in order to gain access to restricted sensitive information
    - e.g., steal passwords by sniffing wireless traffic
- Denial of Service (DoS)
  - an attacker disrupts a specific targeted service
    - e.g., block the youtube website
The attackers

- Hackers
  - look for challenge, notoriety, and fun
    - e.g., hackers, script kiddies, students :-D
- Spies
  - look for political/business gains
    - e.g., intelligence, police, industrial spies
- Criminals
  - look for financial gains, religious/political visibility, or just to break something
    - e.g., criminals, terrorists, vandals
Definitions

- **Key**
  - input of cryptographic functions to determine its output
- **Authentication**
  - proof that the message is coming from the one claiming to be at the origin of the message
- **Integrity**
  - proof that the message has not been altered since its creation
- **Non-repudiation of origin**
  - an entity that generated a message cannot deny have generated the message
- **Encryption**
  - action of encoding of a message such that an eavesdropper can’t read the message but legitimate destination can
- **Decryption**
  - action of decoding an encrypted message
- **Signature**
  - a mathematically constructed proof of authenticity of a message
Hall of fame

- Alice and Bob
  - are legitimate users, Alice and Bob exchange messages
- Chuck
  - is a malicious user that is not between Alice and Bob
- Eve
  - is a malicious user that can eavesdrop
- Trudy
  - is a malicious user that can perform (wo)man-in-the-middle attacks
- Trent
  - is a legitimate user that plays the role of a trusted arbitrator
Why is good security level so hard to obtain?

- The security level of a system equals the security level of the weakest part of the system.
  - e.g., encrypting your HDD to avoid information leak if the laptop is stolen is useless if the password is written on a post-it attached on the laptop.
- Digital systems are complex with many components, interactions, and easily bugged.
Security is a tradeoff

- Compare cost and probability of an attack and cost of securing the system against this attack
  - e.g., is that necessary to make data unbreakable for 20 years if they are outdated after 1 hour?
- Explain the security systems and their reasons
  - if a user does not understand why he must follow a procedure, he will not follow it
    - e.g., how many of you already give their password to someone else?
- Never “over-secure” a system
  - if the system is too hard to use, people will find countermeasure
    - e.g., too hard to use corporate mails? Then use gmail to send corporate mails...
Security is a tradeoff (contd.)

- Protection system
  - lifetime = 10 years
  - cost = 10,000 EUR

- Attack
  - yearly probability = 10%
  - cost of restoring the system = 1,000 EUR

Do I invest?
Procedures!

- Protection will never be perfect
- Prepare procedures
  - what to do BEFORE an attack?
    - what to do to limit the risk (e.g., passwords) of attack and to be ready if an attack happens (e.g., backup)
  - what to do DURING an attack?
    - the attack is on going, how to stop it
  - what to do AFTER an attack?
    - the attack succeeded, how to recover from it
Threat Risk Modelling*

* https://www.owasp.org/index.php/Threat_Risk_Modeling
DREAD

- **Damage** Reproducibility Exploitability Affected users Discoverability (DREAD) is a classification scheme to assess and compare the risk presented by each evaluated threat.
- Risk_DREAD = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5
- Damage Potential (how much damage can it cause?)
  - e.g., 0 = nothing, 5 = some, 10 = complete
- Reproducibility (how easy is it to reproduce the threat?)
  - e.g., 0 = impossible, 5 = few steps, need authentication, 10 = simple, no authentication needed.
- Exploitability (what is needed to exploit this threat?)
  - e.g., 0 = advanced tools and knowledge, 4 = using public attack tools, 10 = just a web browser
- Affected users (how many users will be affected?)
  - e.g., 0 = none, 5 = some, 10 = all users
- Discoverability (how easy is it to discover this threat?)
  - e.g., 0 = very hard, 5 = need monitoring, 9 = documented publicly, 10 = visible in the address bar.
Securing communications
Objective

- Construct a communication mechanism where Alice and Bob can exchange messages such that
- only Alice and Bob can generate messages
- nobody else than Alice or Bob can read messages
- nobody can alter messages
Steps

- fill me
- fill me
- fill me
Hash function

- Validate that a message has not been altered on its way between Alice and Bob
- Hash functions map arbitrary large numbers of variable length to fixed-length numbers
  - $h = H(m)$, $h$ is called hash or digest
  - e.g., MD5, SHA-1, SHA-256
- Good hash functions for cryptography must be such that
  - $H(m)$ is not complex to compute
  - but finding a $m_2$ such that $H(m_2) = H(m)$ is complex,
  - $H(m)$ is deterministic,
  - $H$ output must be evenly distributed over the output set
- Example
  - SHA-1 maps messages its input space on a 160-bits output
    - $\text{SHA-1}(\text{Message to validate}) = 5e06ee754bda0d33cf65ec305ffec779404e66029$
    - $\text{SHA-1}(\text{Message to validate}) = b1c306f8cb792fa14d4d1fdcf6f37d86c2fe6bb9$
Is that enough?

Alice  Trudy  Bob
Is that enough?

$\text{Alice} \quad \text{Trudy} \quad \text{Bob}$

\[ d = H(\text{msg}) \]
Is that enough?

Alice

msg
d = H(msg)

Trudy

msg, d

Bob
Is that enough?

msg
d = H(msg)

valid as d = H(msg)
Is that enough?

Alice

\[ d = H(\text{msg}) \]

msg

Trudy

\[ \text{msg, } d \]

Bob

\[ \text{valid as } d = H(\text{msg}) \]

\[ d_2 = H(\text{msg}_2) \]
Is that enough?

Alice
- msg
- $d = H(msg)$

Trudy
- msg, $d$

Bob
- valid as $d = H(msg)$

msg$_2$
- $d_2 = H(msg_2)$

msg$_2$, $d_2$
Is that enough?

- Alice
  - msg
  - $d = H(msg)$
  - $msg_2$
  - $d_2 = H(msg_2)$

- Trudy
  - msg, $d$
  - $msg_2, d_2$
  - msg
  - $d_3 = H(msg_3)$

- Bob
  - valid as $d = H(msg)$
Is that enough?

Alice

\[ d = H(\text{msg}) \]

\[ \text{msg} \]

\[ d_2 = H(\text{msg}_2) \]

Trudy

\[ \text{msg}, d \]

\[ \text{msg}_2, d_2 \]

Bob

\[ \text{msg}, d \]

\[ \text{msg}_3, d_3 \]

valid as \( d = H(\text{msg}) \)
Is that enough?

\[
\text{Alice} \quad \text{Trudy} \quad \text{Bob}
\]

\[
\text{msg} \quad \text{msg}, d \quad \text{valid as } d = H(\text{msg})
\]

\[
\text{d} = H(\text{msg}) \\
\text{msg}_2 \quad \text{msg}_2 \quad \text{msg}_2, d_2 \quad \text{msg}_2, d_2 \quad \text{msg}_2, d_2
\]

\[
\text{d}_2 = H(\text{msg}_2) \\
\text{d}_3 = H(\text{msg}_3) \\
\text{msg}_3 \quad \text{msg}_3, d_3 \quad \text{valid as } d_3 = H(\text{msg}_3)
\]

\[
\text{msg}_3
\]

\[
\text{d}_3
\]
Hash function with salt

- Hash functions are deterministic
- Add a salt such that the output of the hash function is a function of the message and the salt
  - $h = H(m, K)$ where $K$ is the salt or key of the hash function
- As long as Trudy does not know the salt, she can’t forge a valid digest
Hash function with salt (contd.)

Alice
K

Trudy

Bob
K
Hash function with salt (contd.)

$\text{Alice}$

$K$

$\text{msg}$

$d = H(\text{msg}, K)$

$\text{Trudy}$

$
\text{Bob}$

$K$
Hash function with salt (contd.)

\[ d = H(\text{msg}, K) \]
Hash function with salt (contd.)

\[ d = H(\text{msg}, K) \]
Hash function with salt (contd.)

Alice

K
msg
d = H(msg, K)

msg₂
d₂ = H(msg₂, K)

Trudy

msg, d

Bob

K

valid as d = H(msg, K)
Hash function with salt (contd.)

\[
d = H(msg, K)
\]

Alice

\[
K
\]

\[
msg
\]

\[
d = H(msg, K)
\]

Trudy

Bob

\[
K
\]

\[
msg, d
\]

valid as \( d = H(msg, K) \)

\[
msg_2
\]

\[
d_2 = H(msg_2, K)
\]

\[
msg_2, d_2
\]
Hash function with salt (contd.)

Alice

\( K \)

\( \text{msg} \)

\( d = H(\text{msg}, K) \)

Bob

\( K \)

valid as \( d = H(\text{msg}, K) \)

Trudy

\( \text{msg}, d \)

**msg_2, d_2**

**msg_3**

\( d_2 = H(\text{msg}_2, K) \)

\( d_3 = H(\text{msg}_3) \)
Hash function with salt (contd.)

Alice

K
msg
d = H(msg, K)

msg

Trudy

msg, d

msg2, d2

Bob

K

valid as d = H(msg, K)

d2 = H(msg2, K)

msg3

d3 = H(msg3)

msg3, d3
Hash function with salt (contd.)

Alice

- $K$
- $msg$
- $d = H(msg, K)$

Trudy

- $msg, d$

Bob

- $K$

valid as $d = H(msg, K)$

invalid as $d_3 \neq H(msg_3, K)$

$K$

$msg$

$d = H(msg, K)$

$msg_2$, $d_2$

$H(msg_2, K)$

$K$

$msg_3$, $d_3$

$H(msg_3)$
Problem solved?

- fill me
- fill me
- fill me
Problem solved?

- fill me
- fill me
- fill me
- fill me

How can Alice and Bob agree on K?
Diffie-Hellman key exchange

- How can Alice and Bob agree on a secret number and be sure that Eve will not discover it?

- Principle
  - do not exchange the secret number but other numbers that are use to build up the secret
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

Alice → Eve → Bob
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

Alice

Eve

Bob

a, g, m
Working on finite group and positive integers

\[ A \equiv g^a \mod m \]
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

Alice

Eve

Bob

\[ A \equiv g^a \mod m \]

\[ A, g, m \]
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

Alice

\( a, g, m \)

\( A \equiv g^a \mod m \)

Eve

\( A, g, m \)

Bob

\( b \)
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

\[ A \equiv g^a \mod m \]

Alice

Eve

Bob

\[ B \equiv g^b \mod m \]
Working on finite group and positive integers

\[
\begin{align*}
A &\equiv g^a \mod m \\
B &\equiv g^b \mod m \\
K &\equiv A^b \mod m
\end{align*}
\]
Working on finite group and positive integers

Diffie-Hellman key exchange (contd.)

- Alice
  - a, g, m
  - \( A \equiv g^a \mod m \)

- Eve
  - A, g, m
  - B

- Bob
  - b
  - \( B \equiv g^b \mod m \)
  - \( K \equiv A^b \mod m \)
Working on finite group and positive integers

\[ A \equiv g^a \mod m \]

\[ K \equiv B^a \mod m \]

\[ B \equiv g^b \mod m \]

\[ K \equiv A^b \mod m \]
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Eve</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>a, g, m</td>
<td>a, g, m</td>
<td>b</td>
</tr>
<tr>
<td>A ≡ g^a mod m</td>
<td>A, g, m</td>
<td>B ≡ g^b mod m</td>
</tr>
<tr>
<td>K ≡ B^a mod m</td>
<td></td>
<td>K ≡ A^b mod m</td>
</tr>
</tbody>
</table>

K ≡ A^b mod m ≡ (g^a mod m)^b mod m ≡ g^{ba} mod m ≡ (g^b mod m)^a mod m ≡ B^a mod m ≡ K
```
Why can’t Eve guess K if she knows A, B, g, and m?
- discrete exponentiation is linear with the size of the argument
  - easy to compute $x \equiv y^z \mod p$
- but for some discrete groups, no efficient algorithm is known to compute discrete logarithm
  - hard to determine natural $z$ that ensures $x \equiv y^z \mod p$
- Eve knows A, B, g, and m but can’t determine neither $a$ nor $b$ that are absolutely necessary to compute K
  - $K \equiv A^b \mod m \equiv (g^a \mod m)^b \mod p \equiv g^{ba} \mod m \equiv (g \mod m)^a \mod m \equiv B \mod m$
Trudy can break Diffie-Hellman key exchange (contd.)
Diffie-Hellman key exchange (contd.)

- How can we protect Diffie-Hellman from Trudy?
- Principle
  - Alice and Bob sign the messages exchanged in Diffie-Hellman (?!)

82
Asymmetric cryptography

In asymmetric cryptography (aka public-key cryptography), two keys are used:

- **public key**
  - publicly available to anybody (even attackers)
  - used to encrypt a message

- **private key**
  - known only by the legitimate owner of the public key
  - used to decrypt a message

- **e.g., RSA, PGP, Diffie-Hellman**

- Public-key cryptography is 10 to 100 times slower than symmetric-key cryptography
  - seldom (never?) used to encrypt communications
Asymmetric cryptography (contd.)

Eve cannot determine the message

\[
m \rightarrow c = \text{crypt}(m, \text{Public}_B) \rightarrow c \rightarrow \text{decrypt}(c, \text{Private}_B) = m
\]
Trudy can send a forged message

Asymmetric cryptography (contd.)

\[
\begin{align*}
\text{Alice} & \quad \text{Trudy} \quad \text{Bob} \\
\text{Public}_B & \quad \text{Public}_B & \quad \text{Public}_B, \text{Private}_B \\
\text{m} = \text{crypt}(m, \text{Public}_B) & \quad c & \quad \text{decrypt}(c, \text{Private}_B) = m \\
\text{m}_2 & \quad \text{c}_2 & \quad \text{decrypt}(\text{c}_3, \text{Private}_B) = \text{m}_3 \\
\text{c}_2 = \text{crypt}(\text{m}_2, \text{Public}_B) & \quad \text{c}_3 & \quad \text{c}_3 = \text{crypt}(\text{m}_3, \text{Public}_B)
\end{align*}
\]
Eve can read the message

\[ s = \text{sign}(m, \text{Private}_A) \]

\[ \text{check}(m, s, \text{Public}_A) \]
How to build sign and check?

- $s = \text{sign}(H(m), k) = \text{crypt}(H(m), k)$
- $\text{check}(m, s, K) = (H(m)==\text{decrypt}(s, K))$

  - where $k$ is the private key of the signer and $K$ is the public key

- Asymmetric cryptography is slow and $m$ can be large
  - encrypting $m$ would be too costly
  - solution: consider the digest of $m$ while signing
Public key infrastructure

- How to safely obtain Bob’s public key?

Alice  Trudy  Bob

Public_B, Private_B
Public key infrastructure

How to safely obtain Bob’s public key?

Alice → Trudy → Bob

What is your public key?

Public_B, Private_B
How to safely obtain Bob’s public key?

Alice  Trudy  Bob

What is your public key?

Public_B
Public key infrastructure

- How to safely obtain Bob’s public key?

Alice  Trudy  Bob

- What is your public key?
- PublicB

PublicB, PrivateB
Public key infrastructure (contd.)

- Trudy can send a forged key

```
Alice  Trudy  Bob

Public_T, Private_T

What is your public key?

Public_T

Public_T
```

89
Alice and Bob trust a third party (e.g., Trent) for authentication.
Public key infrastructure (contd.)

- Practically, Bob sends a certificate (e.g., X.509), not only its public key and signature

- A certificate provides many information to be able to correctly identify and authenticate its subject (e.g., Bob)
  - the subject name and organization
  - the subject public key (and type)
  - the issuer name and organization
  - the certificate validity time (valid not before and not after)
  - the certificate signature and type, signature made by the issuer of the certificate
  - ...

...
Public key infrastructure (contd.)

- Certificates are issued once and valid during a given time period, whatever the number of time it is used
- What if the subjects leaves its organization? The private key of the subject is stolen? The private key of the issuer is stolen?
- When a certified key is compromised, the certificate is revoked
  - the issuer maintains the list of revoked certificates
  - that should be checked by the client.
Diffie-Hellman key exchange (the return)

- Trudy cannot perform her attack anymore

\[
\begin{align*}
A &\equiv g^a \mod m \\
s_A &\equiv \text{sign}((A, g, m), \text{Private}_A) \\
B &\equiv g^b \mod m \\
s_B &\equiv \text{sign}(B, \text{Private}_B) \\
K &\equiv B^a \mod m
\end{align*}
\]
Problem solved?

- fill me
- fill me
- fill me
Problem solved?

- fill me
- fill me
- fill me

Replay attacks are still possible!
Trudy can replay a message
Nonce

- Trudy can replay a message

Alice

Public$_A$, Private$_A$

m = “open door”
s = sign(m, Private$_A$)

Trudy

Public$_A$

Bob

Public$_A$

m = “open door”
s = sign(m, Private$_A$)
Trudy can replay a message

Alice

Public\textsubscript{A}, Private\textsubscript{A}

m = “open door”
s = sign(m, Private\textsubscript{A})

Trudy

Public\textsubscript{A}

m, s

Bob

Public\textsubscript{A}
Trudy can replay a message

\[ m = \text{“open door”} \]
\[ s = \text{sign}(m, \text{Private}_A) \]

Remember \((m, s)\)
Trudy can replay a message

Alice: $m = \text{"open door"}$
\[ s = \text{sign}(m, \text{Private}_A) \]

Trudy: $m, s$

Remember $(m, s)$

Bob: $\text{check}(m, s, \text{Public}_A)$

Door is open
Trudy can replay a message

- Alice
  - Public$_A$, Private$_A$
  - $m = \text{"open door"}$
  - $s = \text{sign}(m, \text{Private}_A)$

- Trudy
  - $m$, $s$
  - remember $(m, s)$

- Bob
  - Public$_A$
  - check$(m$, $s$, Public$_A)$
  - door is open

- Trudy can replay a message

- $m_2 = \text{"close door"}$
- $s_2 = \text{sign}(m_2, \text{Private}_A)$
Trudy can replay a message

\[ m = \text{“open door”} \]
\[ s = \text{sign}(m, \text{Private}_A) \]
\[ m, s \]

\[ m_2 = \text{“close door”} \]
\[ s_2 = \text{sign}(m_2, \text{Private}_A) \]

check\((m, s, \text{Public}_A)\)
door is open
Trudy can replay a message

Trudy can replay a message

\( m = \text{"open door"} \)
\( s = \text{sign}(m, \text{Private}_A) \)

\( m_2 = \text{"close door"} \)
\( s_2 = \text{sign}(m_2, \text{Private}_A) \)

check\((m, s, \text{Public}_A)\)
\( \text{door is open} \)

check\((m_2, s_2, \text{Public}_A)\)
\( \text{door is closed} \)
Nonce

- Trudy can replay a message

Alice

Publicₐ, Privateₐ

m = “open door”

s = sign(m, Privateₐ)

m₂ = “close door”

s₂ = sign(m₂, Privateₐ)

Trudy

Publicₐ

m, s

remember (m, s)

Bob

Publicₐ

check(m, s, Publicₐ)

door is open

m₂, s₂

check(m₂, s₂, Publicₐ)

door is closed

m, s
Nonce

- Trudy can replay a message

Trudy can replay a message by pretending to be Alice and sending a message that Alice has already sent. This can be done by signing a message with a private key and sending it to Bob, who then verifies it with the corresponding public key.

Example:

- Alice sends a message: "open door" with signature: $s = \text{sign}(m, \text{Private}_A)$
- Trudy intercepts the message and pretending to be Alice, sends the same message: $m$, $s$
- Bob verifies the signature: $\text{check}(m, s, \text{Public}_A)$
- Bob assumes the message is from Alice
- Trudy can now use the signature to sign a new message: $m_2 = \text{"close door"}$, $s_2 = \text{sign}(m_2, \text{Private}_A)$
- Trudy sends $m_2$, $s_2$ to Bob
- Bob verifies the signature: $\text{check}(m_2, s_2, \text{Public}_A)$
- Bob assumes the message is from Alice
- Trudy can now claim that the door is closed, which is a replay of the original message signed by Alice.
Nonce (contd.)

- A nonce is a number used only once
- Three general methods to create nonces
  - sequential number
    - increment after each use
    - keep it in non-volatile storage in case of reboot
  - timestamp
    - current time of the nonce generation
    - be sure clock is not going backward (e.g., winter time)
  - random number
    - low collision probability if the pseudo random number generator is good and random number is big enough (e.g., more than 128 bits)
- Nonce alone is rarely enough to have a good protection
  - not robust to eavesdropping or man-in-the-middle attack
Nonce (contd.)

- Each message is made unique thanks to the nonce
Nonce (contd.)

- Each message is made unique thanks to the nonce

```
Alice
Publicₐ, Privateₐ
m
n = nonce
s = sign((m, n), Privateₐ)

Trudy
Publicₐ

Bob
Publicₐ
```
Nonce (contd.)

- Each message is made unique thanks to the nonce

```
Alice

Public_A, Private_A
m
n = nonce
s = sign((m, n), Private_A)

Trudy

Public_A
m, n, s

Bob

Public_A
```

97
Nonce (contd.)

- Each message is made unique thanks to the nonce

Alice

Public\textsubscript{A}, Private\textsubscript{A}

\( m \)

\( n = \text{nonce} \)

\( s = \text{sign}((m, n), \text{Private}_A) \)

Bob

Public\textsubscript{A}

Public\textsubscript{A}

Trudy

Public\textsubscript{A}

\( m, n, s \)

remember \( (m, n, s) \)
Nonce (contd.)

- Each message is made unique thanks to the nonce

Alice

Public\textsubscript{A}, Private\textsubscript{A}

\begin{align*}
m &= \text{nonce} \\
n &= \text{nonce} \\
s &= \text{sign}((m, n), \text{Private}_{\text{A}})
\end{align*}

Trudy

Public\textsubscript{A}

remember \((m, n, s)\)

Bob

Public\textsubscript{A}

check\(((m, n), s, \text{Public}_{\text{A}})\)

nonces = \{n\}
Nonce (contd.)

- Each message is made unique thanks to the nonce

---

Alice

Public_A, Private_A

m
n = nonce
s = sign((m, n), Private_A)

m2
n2 = nonce
s2 = sign((m2, n2), Private_A)

Trudy

Public_A

m, n, s

Bob

Public_A

check((m, n), s, Public_A)
nonces = {n}

remember (m, n, s)
Nonce (contd.)

- Each message is made unique thanks to the nonce

Alice

Public$_A$, Private$_A$

$m$

$n = \text{nonce}$

$s = \text{sign}((m, n), \text{Private}_A)$

$m_2$

$n_2 = \text{nonce}$

$s_2 = \text{sign}((m_2,n_2),\text{Private}_A)$

Trudy

$m, n, s$

remember $(m, n, s)$

Bob

Public$_A$

$m_2, n_2, s_2$

check($(m, n), s, \text{Public}_A)$

nonces = {n}
Nonce (contd.)

- Each message is made unique thanks to the nonce

\[
m = \text{nonce} \\
s = \text{sign}((m, n), \text{Private}_A) \\
m_2 = \text{nonce} \\
s_2 = \text{sign}((m_2, n_2), \text{Private}_A)
\]

\[
\text{Alice} \\
\text{Public}_A, \text{Private}_A \\
m \\
n = \text{nonce} \\
s = \text{sign}((m, n), \text{Private}_A) \\
m_2 \\
n_2 = \text{nonce} \\
s_2 = \text{sign}((m_2, n_2), \text{Private}_A)
\]

\[
\text{Trudy} \\
\text{Public}_A \\
m, n, s \\
\text{remember} (m, n, s)
\]

\[
\text{Bob} \\
\text{Public}_A \\
\text{check}((m, n), s, \text{Public}_A) \\
\text{nonces} = \{n\} \\
\text{check}((m_2, n_2), s_2, \text{Public}_A) \\
\text{nonces} = \{n, n_2\}
\]
Nonce (contd.)

- Each message is made unique thanks to the nonce

\[
m = \text{nonce} \quad s = \text{sign}((m, n), \text{Private}_A) \quad \text{check}((m, n), s, \text{Public}_A)
\]

\[
m_2 = \text{nonce} \quad s_2 = \text{sign}((m_2, n_2), \text{Private}_A) \quad \text{check}((m_2, n_2), s_2, \text{Public}_A)
\]
Nonce (contd.)

- Each message is made unique thanks to the nonce

\[
m, n, s = \text{sign}((m, n), \text{Private}_A)
\]

\[
m_2, n_2, s_2 = \text{sign}((m_2, n_2), \text{Private}_A)
\]

Alice

- Public$_A$, Private$_A$
- \( m \)
- \( n = \text{nonce} \)
- \( s = \text{sign}((m, n), \text{Private}_A) \)
- \( m_2 \)
- \( n_2 = \text{nonce} \)
- \( s_2 = \text{sign}((m_2, n_2), \text{Private}_A) \)

Trudy

- Public$_A$
- \( m, n, s \)
- remember \((m, n, s)\)

Bob

- Public$_A$
- \( m, n, s \)
- check\((m, n), s, \text{Public}_A\)
- nonces = \{n\}

- check\((m_2, n_2), s_2, \text{Public}_A\)
- nonces = \{n, n_2\}

- check\((m, n), s, \text{Public}_A\)
- nonce already used: skip
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

Alice  Bob  Chuck
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

```
Alice
m = “abcd”

Bob

Chuck
```
TCP sequence number does not protect against segment injection attacks in TCP

Alice

m = “abcd”

Bob

m, seq=x

Chuck
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

m = “abcd”
m, seq=x

Alice → Bob

“abcd”

Bob → Chuck

m = “abcd”
TCP sequence number does not protect against segment injection attacks in TCP

- \( m = \text{“abcd”} \)
- \( m, \text{seq} = x \rightarrow \text{“abcd”} \)
- \( \text{ack} = x+4 \)
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

\[\text{Alice} \quad \text{Bob} \quad \text{Chuck}\]

- \(m = \text{"abcd"}\)
- \(m, \text{seq}=x\)
- \(\text{ack} = x+4\)
- \(\text{mc} = \text{"123456789"}\)

\(m = \text{"abcd"}\)
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

```
Alice
m = “abcd”
```

```
Bob
m, seq=x
```

```
Chuck
mc = “123456789”
```

```
“abcd”
```

```
ack = x+4
```

```
m_c, seq=x
```

```
m_c = “123456789”
```
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

Alice

- m = "abcd"
- ack = x+4

Bob

- m, seq=x
- "abcd"

Chuck

- mc, seq=x
- "abcd56789"
- mc = "123456789"

m, seq=x
TCP sequence number does not protect against segment injection attacks in TCP

Alice
m = “abcd”

Bob
m, seq=x
ack = x+4
“abcd”
m_c, seq=x
“abcd56789”

Chuck
m_c = “123456789”
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP

```
Alice

m = “abcd”
m2 = “ef”

m, seq=x

m2, seq=x+4

ack = x+4

Bob

“abcd”

m2, seq=x+4

“abcd56789”

Chuck

mc, seq=x

mc = “123456789”
```
TCP sequence number does not protect against segment injection attacks in TCP

Nonce (contd.)

- m = “abcd”
- m2 = “ef”
- m, seq=x
- m2, seq=x+4
- ack = x+4
- ack = x+9
- “abcd”
- “abcd56789”
- mc = “123456789”
TCP sequence number does not protect against segment injection attacks in TCP

- Alice
  - $m = "abcd"
  - $m_2 = "ef"
  - ack = $x+9$

- Bob
  - m, seq=$x$
  - ack = $x+4$
  - $m, seq=x+4$
  - $m_2, seq=x+4$
  - “abcd”
  - “abcd56789”

- Chuck
  - $m_c = "123456789"
  - $m_c, seq=x$
  - “abcd56789”

Nonce (contd.)
TCP sequence number does not protect against segment injection attacks in TCP

- m = "abcd"
- m2 = "ef"
- m, seq=x
- m2, seq=x+4
- ack = x+4
- ack = x+9
- "abcd"
- "abcd56789"
- "abcd56789"
- mc = "123456789"

Nonce (contd.)
TCP sequence number does not protect against segment injection attacks in TCP

Alice

m = “abcd”

m2 = “ef”

Bob

m, seq=x

m2, seq=x+4

ack = x+4

ack = x+9

ack = x+9

Chuck

“abcd”

“abcd56789”

mc = “123456789”

“abcd56789”

“abcd56789”

“abcd56789”

98

m2, seq=x+4

mc, seq=x
Problem solved?

- fill me
- fill me
- fill me
Problem solved?

- fill me
- fill me
- fill me

DoS attacks are still possible!
Denial of Services

- Resources are always limited
  - e.g., processor, memory, link capacity
- The easiest way of leading a DoS is to overwhelm CPUs, memory, or links of the target
- A more complicated way is to manage an intrusion and neutralize the target
- imagine you gain administrative access to border router of your network!
Danger of state

- Establishment and maintenance of session requires state
  - often maintained in “tables” with predefined capacity
- An attacker can saturate state tables by initiating multiple sessions
- Principle
  - require attacker to maintain state before maintaining state yourself
  - in general it is too costly for an attacker to maintain state
TCP relied on a state machine started upon reception of a SYN packet.
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet

Alice

Bob

Chuck

(src=IP_A:port_A, dst=IP_B:port_B, SYN, seq_A=x)
TCP relied on a state machine started upon reception of a SYN packet

Danger of state (contd.)

- Alice
  - SYN.received: \{src=IP_A:port_A, 
    dst=IP_B:port_B, 
    seq_A=x, 
    seq_B=y\}

- Bob

- Chuck
  - (src=IP_A:port_A, 
    dst=IP_B:port_B, 
    SYN, 
    seq_A=x)
TCP relied on a state machine started upon reception of a SYN packet

Alice

SYN.received:
{src=IP_A:port_A,
dst=IP_B:port_B,
seq_A=x,
seq_B=y}

Bob

SYN+ack,
seq_B=y

Chuck

(src=IP_A:port_A,
dst=IP_B:port_B,
SYN,
seq_A=x)
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet

Alice

SYN.received: 
{src=IP_A:port_A, 
dst=IP_B:port_B, 
seq_A=x, 
seq_B=y}

Bob

SYN+ack, 
seq_B=y

Chuck

(src=IP_A:port_A, 
dst=IP_B:port_B, 
SYN, 
seq_A=x)

When to remove state?
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice → Bob → Chuck
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice

Bob

Chuck

(src=IP_A:port_A, dst=IP_B:port_B, SYN, seq=x)
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice

Bob

Chuck

\[(\text{src}=IP_A:port_A, \text{dst}=IP_B:port_B, \text{SYN}, \text{seq}=x)\]

No state created

\[y=H(IP_A, Port_A, \text{secret})\]
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice

Bob

Chuck

(src=IP_A:port_A,
 dst=IP_B:port_B,
 SYN,
 seq=x)

SYN+ack,
 seq_B=y

No state created
 y=H(IP_A, Port_A, secret)
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

```
Alice  Bob  Chuck
(src=IP_A:port_A, dst=IP_B:port_B, SYN, seq=x)
SYN+ack, seq_B=y
ACK(seq=x+1,ack=y+1)
No state created
y=H(IP_A, Port_A, secret)
```
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice: (src=IP_A:port_A, dst=IP_B:port_B, SYN, seq=x)

Bob: SYN+ack, seq_B=y

ACK(seq=x+1,ack=y+1)

Chuck: No state created

\( y = H(\text{IP}_A, \text{Port}_A, \text{secret}) \)

\( \text{check ack= 1 + H(\text{IP}_A, \text{Port}_A, \text{secret})} \)

create state
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)

Alice

\[
\text{src=IP}_A:\text{port}_A,
\text{dst=IP}_B:\text{port}_B,
\text{SYN},
\text{seq}=x)
\]

Bob

\[
\text{SYN+ack,}
\text{seq}_B=y
\]

\[
\text{ACK(seq=x+1,ack=y+1)}
\]

Chuck

\[
\text{check ack= 1 + H(IP}_A, \text{Port}_A, \text{secret)}
\]

\[
\text{create state}
\]

\[
\text{Cannot force state at Bob without creating local state}
\]
Danger of complexity

- Protection mechanism can be complex and can require important processing power
- An attacker can overwhelm her target CPU by triggering protection mechanisms

Principle
- require attacker to perform more processing than yourself
- in general an attacker does not want to have to do heavy computation
Danger of complexity (contd.)

- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
      - e.g., Bob asks Alice to find a J such that the K lowest order bits of H((N,J)) are zeros. N is a nonce and K sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don’t want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA
Danger of complexity (contd.)

- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
      - e.g., Bob asks Alice to find a $J$ such that the $K$ lowest order bits of $H((N,J))$ are zeros. $N$ is a nonce and $K$ sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don’t want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA
Link overloading

- Messages are sent to Bob by traversing links
- If an attacker can send packets at a high enough rate, she can saturate links toward Bob and make him unavailable
- Unfortunately, Bob cannot make anything to block packets before they reach him
- Principle
  - tweak the network to not suffer too much of such attacks
Example of Distributed Denial of Service (DDoS) attack
Attacks are often to random destinations or with random sources

backscatter traffic to a sink-hole that can receive a lot of traffic attack without impacting the network
Link overloading (contd.)

- Use the sink-hole to attract bizarre packets
Link overloading (contd.)

- Use the sink-hole to protect the target

IBGP:
- prefix: Bob/32
- nexthop: sink-hole
- NO_EXPORT
Link overloading (contd.)

- A first parade is to filter illicit traffic before it can harm the target
  - e.g., firewall, access lists
- A set of rules is specified a priori, if the traffic does not match the rules, it is discarded
  - always block everything but what is acceptable
Link overloading (contd.)

- Filtering based on origin
  - useful to avoid spoofing
    - e.g., block any packet which source address does not belong to the customer cone of a BGP neighbor
  - does not work so well as it depends on every network between the origin and the target

- Filtering based on traffic pattern
  - analyze the traffic and if it deviates from what is normal, drop it
    - e.g., drop malformed packets, rate limit a source if it sends too much SYN packets, ignore mails from well known SPAM servers, block any flow initiated by the outside if there is no server in the network
Network Intrusion Detection System (NIDS)

- An NIDS aims at discovering non-legitimate operations.
- The NIDS analyses the traffic to detect abnormal patterns.
- Upon anomaly detection, the NIDS triggers an alert with a report on the anomaly.
- NOC follows procedures upon detection.
Network Intrusion Detection System (contd.)

- Signature based detection
  - a database of abnormal behavior is maintained to construct a signature for each attack
  - if the traffic corresponds to a signature in the database, trigger an alarm
  - risk of false negative (0-day attack)
  - e.g., Snort, Bro, antivirus

- Outlier detection
  - the anomaly detector learns what is the normal behavior of the network
  - went an outlier is detected, an alarm is triggered
  - risk of false positive and false negative
  - e.g., cluster analysis, time series analysis, spectral analysis
if antivirus(self) == BAD:
  skip
else:
  I am bad
Problem solved?

- fill me
- fill me
- fill me
Problem solved?

- fill me
- fill me
- fill me
- fill me

Relay attacks are still possible!
Relay attack

- In a relay attack, Chuck does not contact Alice directly but goes via Bob.
- If the traffic from Bob to Alice is bigger than the traffic from Chuck to Bob, the attack is called amplification attack.
- As for DoS, hard to protect correctly against relay attacks:
  - use filters (e.g., deactivate ICMP)
  - authentication of the source
  - but correct spoofing protection that doesn’t open a relay attack door is very hard to deploy in practice as it requires messages in both directions between parties.
What did we miss?
What did we miss?

- To terminate the session!
- with the same care as the opening of the session
- this is often neglected
Perfect Forward Secrecy

- With perfect forward secrecy (PFS), Eve cannot decrypt messages sent between Alice and Bob.
  - even if she captures every message.
  - even if she breaks into Alice and Bob after the communication to steal their secrets (e.g., private keys).
Perfect Forward Secrecy (contd.)

- PFS is provided using ephemeral keys
  - the ephemeral key is generated and used only during the session
  - the session key is not stored after the communication
  - the session key is independent of stored information (e.g., good PRNG)
  - for long sessions, change the session key regularly
Perfect Forward Secrecy (contd.)

1. Initiate the communication between Alice and Bob
   - authenticity proven with public/private key pairs
2. Alice and Bob agree on a secret K
   - use Diffie-Hellman
     - authenticate DH messages with public/private key pairs
3. Encrypt/Decrypt messages with symmetric cryptography using K as the key
   - no need to sign as it is encrypted
   - be sure a nonce is used to avoid replay
4. If session is too long, back to 2.
5. Close the session correctly and be sure K is not stored anywhere
Privacy
Sharing secrets

- Context
  - $n$ students work on a top-secret project
  - They cannot trust each other
  - The project is in a digital safe
  - To open the digital safe, at least $k$ out of the $n$ students must be present
A polynomial of degree $k-1$ is uniquely identified with $k$ points.
\( (k=4, n=7) \)
\((k=4, n=7)\)
\((k=4, n=7)\)
\( f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \)

\((k=4, n=7)\)
$f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3$

$k=4, n=7$
\[ f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \]

\((k=4, n=7)\)
\( f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \)
\( f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \)

\((k=4,n=7)\)
(k=4, n=7)

\[ f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \]
\( f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \)
\[ f(x) = a_1 + a_2 \cdot x + a_3 \cdot x^2 + a_4 \cdot x^3 \]

\[
 f(x) = \sum_{i=1}^{k} y_i \left( \prod_{j=1, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \right)
\]

\( (k=4, n=7) \)
(k,n) threshold scheme

- $D = [x_1, \ldots, x_n]$ is a data composed of $n$ pieces
- When at least $k$ pieces $x_i$ of $D$ are known
  - $D$ can be computed
- otherwise $D$ remains undetermined
(k,n) threshold scheme

- $D = [x_1, \ldots, x_n]$ is a data composed of $n$ pieces

A polynomial of degree $k-1$ is uniquely identified with $k$ points

- $D$ can be computed
- otherwise $D$ remains undetermined
Shamir’s (k,n) Threshold Scheme

- Let $D$ be our secret (an integer), decomposed in $n$ pieces.
- Let $p$ be a prime number $p > \max(D, n)$.
- Generate $k-1$ random number $a_i$
  
  \[
  \forall i \in [1; k - 1] | a_i \in [0; p[
  \]
- Define the polynomial of degree $k-1$
  
  \[
  g(x) = D + a_1 \cdot x^1 + \cdots + a_{k-1} \cdot x^{k-1}
  \]
- Note that $g(0) = D$
Shamir’s (k,n) Threshold Scheme (contd.)

- Generate $n$ fragments of the secret
  
  $D_1 = g(1) \mod p, D_2 = g(2) \mod p, \ldots D_n = g(n) \mod p$

- Distribute $(x_i, D_i)$

- Recompute $D$ from $k$ fragments $(x_j, D_j)$ among $n$ using Lagrange polynomial interpolation

\[
g(0) = \sum_{i=1}^{k} D_i \left( \prod_{j=1, j \neq i}^{k} \frac{-x_j}{x_i - x_j} \right)
\]

\[
D \equiv g(0) \mod p
\]
Example $k=3$, $n=5$

- $p = 997$
- Make 5 groups
  - group 1: (1, 547)
  - group 2: (2, 629)
  - group 3: (3, 394)
  - group 4: (4, 839)
  - group 5: (5, 967)
Example $k=3$, $n=5$

- $p = 997$
- Make 5 groups
  - group 1: (1, 547)
  - group 2: (2, 629)
  - group 3: (3, 394)
  - group 4: (4, 839)
  - group 5: (5, 967)

Collaborate with 2 other groups to compute the secret $D$
Example k=3, n = 5
(contd.)

Group 1, 3, 4

\[ g(0) = 547 \left( \frac{-3}{1 - 3} \frac{-4}{1 - 4} \right) + 394 \left( \frac{-1}{3 - 1} \frac{-4}{3 - 4} \right) + 839 \left( \frac{-1}{4 - 1} \frac{-3}{4 - 3} \right) \]

\[ g(0) = 547 \times 2 - 394 \times 2 + 839 = 1145 \]

\[ g(0) \mod 997 = 148 \]
Example k=3, n =5 (contd.)

To compute it, we took $D = 148$, $p = 997$ a prime number, and the polynomial $p=997$ (prime), $a_1=59$ (random), $a_2=340$(random) 
$g(x)=148 + 59x + 340x^2$

Such that
$D_1 = g(1) \mod 997 = 547$
$D_2 = g(2) \mod 997 = 1626 \mod 997 = 629$
$D_3 = g(3) \mod 997 = 3385 \mod 997 = 394$
$D_4 = g(4) \mod 997 = 5824 \mod 997 = 839$
$D_5 = g(5) \mod 997 = 8943 \mod 997 = 967$
Shamir’s (k,n) Threshold Scheme (contd.)

- The size of each fragment does not exceed the size of the secret
  - as long as $p$ is chosen of the same order as the secret
- Possible to generate new fragments at any time, without altering the others
- Possible to construct hierarchies by attributing more or less fragments
  - the boss has $k$ fragments, the subaltern has $k/2$, …
- No assumption as opposed to cryptographic functions
Anonymity

- Alice wants to send a message to Bob
- Communications are unsecured
- Nobody can know who is the sender (not even Bob)
- Nobody can know who is the receiver
- Nobody else than Bob can retrieve the message
Mix

- Objectives of a mix
  - Hide correspondences between incoming and outgoing messages
  - Not possible to map a source and an outgoing message (apart for the mix)
  - No possible to map a receiver and an incoming message (apart for the mix)
Mix (contd.)

- If the mix cannot be fully trusted, use a cascade of mixes
- It works as long as untrusted mixes do not collaborate all together
Chaum-net

- Allow to send a sealed message via a cascade of mixes
- In an overlay, each participant has a private/public key pair
- Alice randomly chooses a few of them (e.g., 3) to be mixes
- Alice recursively encrypts the message with the public key of each mixes she selected
Allow to send a sealed message via a cascade of mixes

In an overlay, each participant has a private/public key pair

Alice randomly chooses a few of them (e.g., 3) to be mixes

Alice recursively encrypt the message with the public key of each mixes she selected
Chaum-net example

Alice

Bob
Chaum-net example

Alice

Bob

m
Chaum-net example

Alice

A

B

Bob

m
Chaum-net example

Alice

A

B

Bob

m
Chaum-net example
Chaum-net example

\[ K_{\text{Bob}}(R_0, m) \]
Chaum-net example

$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$
Chaum-net example

$$K_{Bob}(R_0, m)$$

$$K_B(Bob, R_1, K_{Bob}(R_0, m))$$

$$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$$
Chaum-net example

\[ K_{Bob}(R_0, m) \]
\[ K_B(Bob, R_1, K_{Bob}(R_0, m)) \]
\[ K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m))) \]
Chaum-net example

$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$
Chaum-net example

Alice

A

B

Bob

Bob: m

$K_{Bob}(R_0, m)$

$K_B(Bob, R_1, K_{Bob}(R_0, m))$

$K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))$
Chaum-net example

Alice → A → B → Bob

\[ K_{Bob}(R_0, m) \]
\[ K_B(Bob, R_1, K_{Bob}(R_0, m)) \]
\[ K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m))) \]
Chaum-net example

\[ K_{Bob}(R_0, m) \]
\[ K_B(Bob, R_1, K_{Bob}(R_0, m)) \]
\[ K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m))) \]
Chaum-net example

\[ K_{Bob}(R_0, m) \]
\[ K_B(Bob, R_1, K_{Bob}(R_0, m)) \]
\[ K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m))) \]
Cool, I am anonymous!

\[ K_{Bob}(R_0, m) \]
\[ K_B(Bob, R_1, K_{Bob}(R_0, m)) \]
\[ K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m))) \]
Alice

A

B

Bob

Chaum-net example

Are you sure?

\[
\begin{align*}
K_{Bob}(R_0, m) \\
K_B(Bob, R_1, K_{Bob}(R_0, m)) \\
K_a(B, R_2, K_B(Bob, R_1, K_{Bob}(R_0, m)))
\end{align*}
\]
Social behavior
"If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place."
"If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place."

Eric Schmidt, directeur général de Google, 2009
Je n’ai rien à cacher!
Je n’ai rien à cacher!

Les définitions de lois et moralité ne sont pas universelles
Le site Facebook autorise-t-il les photos de mères en train d’allaiter ?

Oui. Nous reconnaissons la beauté et le caractère naturel de l’allaitement, et nous sommes ravis de savoir qu’il est important pour les mères de partager leurs expériences avec autrui sur Facebook. La plupart de ces photos respectent nos règlements.

Veuillez noter que les photos que nous examinons nous sont presque toutes signalées par d'autres membres qui se plaignent de leur partage sur Facebook.
Restrictions sur le contenu en France

En France, nous avons restreint l’accès à du contenu signalé dans le cadre de lois interdisant la négation de la Shoah et l’apologie du terrorisme, ainsi que 32 100 cas d’images uniques liés aux attaques terroristes de novembre 2015 à Paris, qui, selon l’OCLCTIC, constituaient des infractions présumées aux lois françaises de protection de la dignité humaine.

Nombre d'éléments de contenu restreint

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>37,695</td>
</tr>
</tbody>
</table>
Je suis invisible sur Internet
Je suis invisible sur Internet

Mais je l’utilise tout le temps et partout
Je suis invisible sur Internet

Mais je l’utilise tout le temps et partout
L’Internet a beaucoup changé
L’Internet a beaucoup changé
de 4 à plus 1 milliard de terminaux
En principe l’Internet est décentralisé
En principe l’Internet est décentralisé

En pratique il est contrôlé par quelques géants...
En principe l’Internet est décentralisé

En pratique il est contrôlé par quelques géants...
... chez qui il faut s’enregistrer
... chez qui il faut s’enregistrer

Cliquez ici pour accepter
Bienvenue dans les règles de confidentialité de Google

Données que nous collectons

Lorsque vous utilisez nos services, vous nous fixez certaines données. Nous y ajoutons celles que nous ne pouvons pas contrôler ou que nous ne pouvons pas contrôler. Nous vous donnons des informations et des conseils sur la façon de gérer les données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler.

Règles de confidentialité

Date de la dernière modification: 23 mars 2016 (voir les versions antérieures)

Comment nous utilisons les données que nous collectons

Les données que nous collectons deviennent notre patrimoine et nous devons savoir les utiliser de manière responsable. Nous ne nous engageons pas à collecter des informations qui sont inutiles ou qui sont utilisées de manière incorrecte. Nous garantissons que nous ne pouvons pas contrôler les données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler.

Données que nous partageons

Nous ne partageons pas de nos informations personnelles avec d'autres personnes, d'organisations ou d'autres individus à moins qu'ils aient votre consentement.

Pratiques spécifiques à certains produits

Google utilise des informations pour améliorer ses services. Nous remercions nos utilisateurs pour faire des commentaires qui nous aident à améliorer nos services. Nous ne pouvons pas contrôler les données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler, des données que nous pouvons contrôler.
Données que nous partageons

Nous ne communiquons vos données personnelles à des entreprises, des organisations ou des personnes tierces que dans les circonstances suivantes :

- Avec votre consentement

Nous ne communiquons des données personnelles vous concernant à des entreprises, des organisations ou des personnes tierces qu'avec votre consentement. Nous demandons toujours votre autorisation avant de communiquer à des tiers des données personnelles sensibles.
Données que nous partageons

Nous ne communiquons vos données personnelles à des entreprises, des organisations ou des personnes tierces que dans les circonstances suivantes :

- Avec votre consentement
- Pour des raisons juridiques

Nous ne partagerons des données personnelles avec des entreprises, des organisations ou des personnes tierces que si nous pensons en toute bonne foi que l’accès, l’utilisation, la protection ou la divulgation de ces données est raisonnablement justifiée pour :
  - se confronter à des obligations légales, réglementaires, judiciaires ou administratives ;
  - faire appliquer les conditions d’utilisation en vigueur, y compris pour constater d’éventuels manquements à celles-ci ;
  - déceler, éviter ou traiter des activités frauduleuses, les atteintes à la sécurité ou tout problème d’ordre technique ;
  - se prémunir contre toute atteinte aux droits, aux biens ou à la sécurité de Google, de ses utilisateurs ou du public, en application et dans le respect de la loi.

Pour des raisons juridiques

Nous ne partageons des données personnelles avec des entreprises, des organisations ou des personnes tierces que si nous pensons en toute bonne foi que l’accès, l’utilisation, la protection ou la divulgation de ces données est raisonnablement justifiée pour :
  - se confronter à des obligations légales, réglementaires, judiciaires ou administratives ;
  - faire appliquer les conditions d’utilisation en vigueur, y compris pour constater d’éventuels manquements à celles-ci ;
  - déceler, éviter ou traiter des activités frauduleuses, les atteintes à la sécurité ou tout problème d’ordre technique ;
  - se prémunir contre toute atteinte aux droits, aux biens ou à la sécurité de Google, de ses utilisateurs ou du public, en application et dans le respect de la loi.
... et qui son intégrés à tous les sites
... et qui sont intégrés à tous les sites

Cliquez ici pour partager
Je n’ai pas de compte
Je n’ai pas de compte

Je me déconnecte
Risque pour votre vie privée
Risque pour votre vie privée

Je leur fait confiance
Qui utilise Skype?
Qui utilise Skype?

Logiciel de téléphonie par Internet composé
- d’un annuaire téléphonique publique;
- d’un protocole d’échange de paquets audio sur IP.
Qui utilise BitTorrent?
Qui utilise BitTorrent?

Logiciel de partage de fichiers composé d’un protocole d’échange de paquets de données sur IP.
Qui utilise BitTorrent et Skype?
Qui utilise BitTorrent et Skype?

A tout moment il est possible de connaître l’adresse IP
- d’un utilisateur de Skype;
- de machines impliquées dans un téléchargement BitTorrent.
On peut dire qui télécharge quoi/dépuis où!
On peut dire qui télécharge quoi/depuis où!

Depuis chez soi