IPsec and IKE
IPsec?

- Network-layer security protocol (requires only changes to sender/receiver)

Why?

- SSL does not protect against certain attacks, e.g.:
  - Enemy sends forged packet with RST bit set (tears down connection)
  - Enemy sends bogus data for connection
    SSL detects that, but cannot recover, since TCP has accepted the data
- SSL cannot (easily) protect UDP
- Lack of transparency
History

- **SP3:** Layer 3 security protocol for SDNS
- **NLSP:** OSIified version of SP3
  incomprehensible spec.
- **swIPe:** UNIX impl. by Ioannidis and Blaze
- **IPsec:** Many years of design in IETF
  recently revised
IP Security Scenario

- Virtual private networks
- “Phone home” for laptops, telecommuters
- General Internet security?
**IPsec structure**

- Network-layer
- Two modi: host-to-host, „firewall-to-firewall“
- Option for user-granularity keying

<table>
<thead>
<tr>
<th>Application</th>
<th>TCP</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Application</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>TCP</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPsec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application with IPsec (transport mode)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>TCP</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPsec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application with IPsec (tunnel mode)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IPsec transport mode

- IPsec datagram emitted and received by end-system.
- Protects upper level protocols
IPsec – tunneling mode

- Gateways (firewalls) are IPsec aware.
  - Internal traffic not protected
- Hosts need not be.
Tunnel mode illustration

IPsec protects communication on the insecure part of the network.
IPsec services

- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality

- Transport mode protects end-to-end
- Tunnel mode – much more common – is used for VPNs, etc.
  - Inner IP header can have site-local addresses
Security associations (SAs)

- Flows of IPsec data are organized into SAs
- SA is unidirectional from sender to receiver
- Security parameter index (SPI): 32 bits
- Handled via Security Association Database

IP is connectionless; IPsec is connection-oriented!
Main components/protocols:

- Authentication Header (AH):
  - Provides integrity and authenticity
- Encapsulated Security Payload (ESP):
  - Encrypts data
- Internet Key Exchange (IKE) protocol:
  - Allows peers (end systems, gateways, routers, firewalls) to agree on methods, algorithms, keys
AH protocol

- Support for data integrity, source authentication, and against replay attacks
  - But not encryption
- Authentication uses MAC with shared secret key
  - HMAC with either MD5-96 or SHA-1-96
- Approach:
  - Use IKE protocol to establish shared secret key
    (Alternative: preconfigured keys)
  - Establish SAs in both directions
  - AH protocol receives incoming IP datagrams from one SA
  - AH protocol sends outgoing IP datagrams via the other SA
AH protocol framing

Before applying AH

| orig IP header | TCP header | data |

Transport mode

| orig IP header | AH hdr | TCP header | data |

Tunnel mode

| new IP header | AH hdr | orig IP header | TCP header | data |

In each case, the protocol field in the left-most IP header is 51, indicating that the AH protocol is used.

Question: How does destination peer determine upper-layer protocol to which it passes the payload?
## Authentication header

<table>
<thead>
<tr>
<th>Bit:</th>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Header</td>
<td>Payload Length</td>
<td>RESERVED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Parameters Index (SPI)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authentication Data (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Next header**: indicates if datagram carries TCP, UDP, ICMP, etc.
- **Payload length**: length of authentication header
- **SPI (Security parameter index)**: identifies SA (algorithm, policy, rules)
- **Sequence number**: incremented for each datagram in SA
- **Authentication data**: contains MAC for this packet
AH: sequence numbers

- **Goal:**
  - Prevent attacker from sniffing and replaying a packet
    - Duplicate, authenticated IP packets may disrupt service

- **Method:**
  - Destination checks for duplicates
  - Does not keep track of ALL received packets
  - Use sliding window instead

![Sliding Window Diagram]

**Diagram Notes:**
- **Fixed window size** $W$
- **Advance window if valid packet to the right is received**
- **$N - W$** marked if valid packet received
- **$N + 1$** unmarked if valid packet not yet received
AH: transport mode

Before applying AH

| orig IP header | TCP header | data |

Transport mode

| orig IP header | AH hdr | TCP header | data |

Mac calculated over:
+ IP header fields that do not change in transit (not TTL)
+ AH header excluding Authentication Data field (the MAC itself)
+ entire upper-layer payload (e.g., TCP segment)

Source IP address = source host address
Destination IP address = destination host address
AH: tunnel mode

Before applying AH

<table>
<thead>
<tr>
<th>orig IP header</th>
<th>TCP header</th>
<th>data</th>
</tr>
</thead>
</table>

Tunnel mode

<table>
<thead>
<tr>
<th>new IP header</th>
<th>AH hdr</th>
<th>orig IP header</th>
<th>TCP header</th>
<th>data</th>
</tr>
</thead>
</table>

**MAC calculated over:**
+ original IP datagram
+ new IP header fields that do not change in transit
  (not TTL or checksum fields)
+ AH header excluding Authentication Data field (the MAC itself)

Original datagram is carried in the payload of the new datagram – fields do not change in payload.

If both peers are gateways, then source and dest IP addresses are those of the gateways.
AH: Summary

- Authentication
  - Incorporates shared secret into MAC

- Message integrity
  - MAC appended to message

- Counters playback attack
  - Sequence numbers
ESP protocol

- ESP protocol provides confidentiality through encryption
- Symmetric encryption with shared secret key
- IKE protocol can be used to establish shared secret key
- Uses SPIs as did AH

- Opt. authentication services similar to AH protocol
- Can be used with null authentication or null encryption
## Transport mode

- Authentication does not include fields in original IP header (as with AH)

- Attacker can perform traffic analysis:
  - Examine protocol numbers
  - Ultimate destination
  - Other IP fields
Tunnel mode

- Ultimate destination is hidden
- Can counter traffic analysis

- Hosts do not have to do encryption
  - done by gateways/firewalls.
    - When in routers/firewalls, key distribution is easier.
Padding: encryption algorithm often require specific input sizes (e.g., a multiple of a block in a block cipher).
ESP example

1. Remote host prepares datagram with dest. address of server.
2. ESP trailer appended to datagram; result encrypted.
3. ESP header added; MAC across all of it appended
4. Datagram header added. SA = remote host, DA = firewall
5. Firewall receives datagram and verifies MAC; using SPI in ESP header, firewall decrypts to get plaintext inner datagram. Firewall sends inner datagram to server.
Possible encryption algorithms

- DES
- 3DES
- AES
- RC5
- IDEA
- 3-IDEA
- 3-IDEA
- CAST
- Blowfish
- ....
SA: Security Association

- Think of it as IPsec connection: defines
  - Protocol used (AH, ESP)
  - Mode (transport, tunnel)
  - Encryption or hashing algorithm to be used
  - Negotiated keys, key lifetime, lifetime of this SA
  - ... plus other info

- Both sides have to agree on SA
- More than one SA per packet!
IPsec vs. SSL

- SSL is in application; IPsec in OS
  - IPsec protects all apps

- SSL is susceptible to a DoS attack:
  - Attacker inserts bogus TCP segment into packet stream: with correct TCP checksum and seq #s
  - TCP acks segment and sends it up to SSL.
  - SSL will discard since integrity check is bogus
  - Real segment arrives: TCP rejects since it has wrong seq #
  - SSL never gets real segment: closes connection

- What if attacker inserts a bogus IPsec datagram?
  - IPsec receiver drops datagram as integrity check fails
  - Real segment arrives, passes integrity check, is passed to TCP – no problem!
Virtual Private Networks (VPN)

- ESP is often used to implement a VPN
  - Packets go from internal network to a gateway with TCP / IP headers for address in another network
  - Entire packet hidden by encryption
    - Including original headers
  - Receiving gateway decrypts packet forwards original IP packet to receiving address in protected network

- Known as VPN tunnel
  - Secure communication between parts of the same organization over public untrusted Internet
IPsec and firewalls

- Encryption is not authentication or authorization!
- Access controls may be needed to encrypted traffic
- Source IP address is only authenticated if somehow bound to a certificate
- Encrypted traffic can use different firewall; however, co-ordination of policies may be needed!
ESP together with AH

- AH and ESP are often combined
- End-to-end AH in transport mode
  - Authenticate packet sources
- Gateway-to-gateway ESP in tunnel mode
  - Hide packet contents and addresses on insecure part of network
- Significant cryptographic overhead
  - Even with AH
Realization issues

- IPsec often relies on DNS
  - IP addresses
  - User specified hostnames
  - Attacker can try to subvert mapping

- DNSSEC may not meet some organizational security standards
  - E.g.: DNSSEC uses its own certificates, not X.509

- How to enforce authorization and cryptography?
  - How do apps request cryptographic protection? How do they verify its existence?
  - How do administrators mandate cryptography between host or network pairs?
Secure key establishment

- What properties are needed?
  - Authentication (know identity of other party)
  - Secrecy (generated key not known to any others)
  - Forward secrecy (compromise of one session key does not compromise keys in other sessions)
  - Prevent replay of old key material
  - Prevent denial of service
  - Protect identities from eavesdroppers
  - ???
Key management in IPsec

- Manual key management
  - Keys and parameters exchanged offline (e.g., by phone), security associations established by hand

- Pre-shared symmetric keys
  - New session key derived for each session by hashing pre-shared key with session-specific nonces
  - Standard symmetric-key authentication and encryption

- Online key establishment
  - Internet Key Exchange (IKE) protocol
  - Use Diffie-Hellman to derive shared symmetric key
Diffie-Hellman key exchange

\[ g^a \mod p \]
\[ g^b \mod p \]

Authentication? No
Secrecy? Only against passive attacker
Replay attack? Vulnerable
Forward secrecy? Yes
Denial of service? Vulnerable
Identity protection? Yes

Participants can’t tell \( g^x \mod p \) from a random element of \( G \): send them garbage and they’ll do expensive exponentiations.
IKE genealogy

- **Diffie-Hellman**
  - 1976
  - + authentication, identity protection

- **Station-to-Station**
  - Diffie, van Oorschot, Wiener 1992
  - + defense against denial of service

- **ISAKMP**
  - NSA 1998
  - “generic” protocol for establishing security associations
  - + defense against replay

- **Photuris**
  - Karn, Simpson 1994-99
  - + compatibility with ISAKMP

- **IKE**
  - Cisco 1998

- **IKEv2**

- **Internet standard December 2005**
Design objectives for key exchange

- Shared secret
  - Create and agree on a secret

- Authentication
  - Participants need to verify each other’s identity

- Identity protection
  - Eavesdropper should not be able to infer participants’ identities by observing protocol execution

- Protection against denial of service
  - Malicious participant should not be able to exploit the protocol to cause the other party to waste resources
Ingredient 1: Diffie-Hellman

A → B: \( g^a \)
B → A: \( g^b \)

- Shared secret is \( g^{ab} \), compute key as \( k = \text{hash(rand, } g^{ab}) \)
  - Diffie-Hellman guarantees perfect forward secrecy
- Authentication
- Identity protection
- DoS protection
Ingredient 2: Challenge-Response

A → B: m, A
B → A: n, sig_B(m, n, A)
A → B: sig_A(m, n, B)

- Shared secret
- Authentication
  - A receives his own number m signed by B’s private key and deduces that B is on the other end; similar for B
- Identity protection
- DoS protection
DH + Challenge-Response

ISO 9798-3 protocol:

A → B:  \( g^a, A \)
B → A:  \( g^b, \text{sig}_B(g^a, g^b, A) \)
A → B:  \( \text{sig}_A(g^a, g^b, B) \)

- Shared secret: \( g^{ab} \)
- Authentication
- Identity protection
- DoS protection

\[ m := g^a \]
\[ n := g^b \]
**Ingredient 3: Encryption**

Encrypt signatures to protect identities:

- $A \rightarrow B$: $g^a$, $A$
- $B \rightarrow A$: $g^b$, $\text{Enc}_K(\text{sig}_B(g^a, g^b, A))$
- $A \rightarrow B$: $\text{Enc}_K(\text{sig}_A(g^a, g^b, B))$

- Shared secret: $g^{ab}$
- Authentication
- Identity protection (for responder only!)
- DoS protection

$k=\text{hash}(g^{ab})$
**DoS prevention**

- Denial of service due to resource clogging
  - If responder opens a state for each connection attempt, attacker can initiate thousands of connections from bogus or forged IP addresses
- **Cookies** ensure that responder is stateless until initiator produced at least 2 messages
  - Responder’s state (IP addresses and ports) is stored in an unforgeable cookie and sent to initiator
  - After initiator responds, cookie is regenerated and compared with the cookie returned by the initiator
  - Cost: 2 extra messages
Ingredient 4: Anti-DoS Cookie

"Almost-IKE" protocol:

A → B: \( g^a, A \)
B → A: \( g^b, \text{hash}_{K_b}(g^b, g^a) \)
A → B: \( g^a, g^b, \text{hash}_{K_b}(g^b, g^a), \text{Enc}_{K}(\text{sig}_A(g^a, g^b, B)) \)
B → A: \( g^b, \text{Enc}_{K}(\text{sig}_B(g^a, g^b, A)) \)

- Shared secret: \( g^{ab} \)
- Authentication
- Identity protection
- DoS protection?

Doesn’t quite work: B must remember his DH exponent \( b \) for every connection

\[ k = \text{hash}(g^{ab}) \]
Medium-term secrets and nonces

- Idea: use the same Diffie-Hellman value $g^{ab}$ for every session, update every 10 minutes or so
  - Helps against denial of service
- To ensure keys are different for each session, derive them from $g^{ab}$ and session-specific nonces
  - Nonces guarantee freshness of keys for each session
  - Re-computing $g^a, g^b, g^{ab}$ is costly, generating nonces (fresh random numbers) is cheap
- More efficient and helps with DoS
- But no longer guarantees forward secrecy (why?)
(Simplified) Photuris

[Karn and Simpson]

Cookie\textsubscript{I}, Cookie\textsubscript{R}, offer crypto

Cookie\textsubscript{I}, Cookie\textsubscript{R}, \( g^a \mod p \), select crypto

Cookie\textsubscript{I}, Cookie\textsubscript{R}, \( g^b \mod p \)

switch to \( K = h(g^{ab} \mod p) \)

Cookie\textsubscript{I}, Cookie\textsubscript{R}, \( K = h(g^{ab} \mod p) \)

Alice is called “Initiator” for consistency with IKE terminology

Bob is called “Responder”
IKE Genealogy Redux

Diffie-Hellman 1976
+ authentication, identity protection

Station-to-Station
Diffie, van Oorschot, Wiener 1992
+ defense against denial of service

ISAKMP NSA 1998
“generic” protocol for establishing security associations
+ defense against replay

Photuris Karn, Simpson 1994-99
+ compatibility with ISAKMP

IKE Cisco 1998

IKEv2 Orman 1998

Internet standard December 2005
Cookies in Photuris and ISAKMP

- Photuris cookies are derived from local secret, IP addresses and ports, counter, crypto schemes
  - Same (frequently updated) secret for all connections
- ISAKMP requires unique cookie for each conn.
  - Add timestamp to each cookie to prevent replay attacks
  - Now responder needs to keep state

- Inherent conflict: to prevent replay, need to remember values that you’ve generated or seen before, but keeping state allows denial of service
IKE overview

- Goal: create security association between 2 hosts
  - Shared encryption and authentication keys, agreement on crypto algorithms
- Two phases:
  1st: establishes security association (IKE-SA)
    - Via authenticated Diffie-Hellman
  2nd: creates actual security association (child-SA)
    - Use phase 1 key to create a fresh keys via new nonces
Why two-phase design?

- Expensive 1st phase creates “main” SA
- Cheap 2nd phase allows to create multiple child SAs (based on “main” SA) between same 2 hosts
  - Example: one SA for AH, another SA for ESP
  - Different conversations may need different protection
    - Some traffic only needs integrity protection or short-key crypto
    - Too expensive to always use strongest available protection
  - Avoid multiplexing several conversations over same SA
    - For example, if encryption is used without integrity protection (bad idea!), it may be possible to splice the conversations
  - Different SAs for different classes of service
IKE: Phase One

Instead of running 2nd phase, "piggyback" establishment of child-SA on initial exchange.

Initiator reveals identity first
Prevents "polling" attacks where attacker initiates IKE connections to find out who lives at an IP addr.

Optional: refuse 1st message and demand return of stateless cookie.
IKE: Phase Two (Create Child-SA)

After Phase One, I and R share key $K$

$\text{Enc}_K(\text{proposal, } N_i, [g^a \mod p], \text{traffic})$

Optional re-key using old DH value and fresh nonces

IP address range, ports, protocol id

Crypto suites, protocol (AH, ESP or IPcomp)

$\text{Enc}_K(\text{proposal, } N_r, [g^b \mod p], \text{traffic})$

Can run this several times to create multiple SAs
Other Aspects of IKE

- Interaction with other network protocols
  - How to run IPsec through NAT (Network Address Translation) gateways?

- Error handling
  - Very important! Bleichenbacher attacked SSL by cryptanalyzing error messages from an SSL server

- Protocol management
  - Dead peer detection, rekeying, etc.

- Legacy authentication
  - What if one of the parties doesn’t have a public key?
Current State of IPsec

- Best currently existing VPN standard
  - For example, used in Cisco PIX firewall, many remote access gateways

- IPsec has been out for a few years, but wide deployment has been hindered by complexity
  - ANX (Automotive Networking eXchange) uses IPsec to implement a private network for the Big 3 auto manufacturers and their suppliers