IP-Sec and VPN: Secure Internet Protocol and Virtual Private Networks

Last updated: Wednesday, May 25, 2005

© Prof. Amir Herzberg
Computer Science Department, Bar Ilan University
http://AmirHerzberg.com
Sources

- Very partial, but readable, coverage in Stalling’s book, `Cryptography and Network Security`

- IP-Sec documents from IP-sec working group of Internet Engineering Task Force (IETF):
  - Architecture – RFC 2401
  - Authentication Header (AH) – RFC 2402
  - Encapsulating Security Payload (ESP) – RFC 2406
  - IKE – Internet Key Exchange – RFC 2409
    - Now working on version 2 – currently draft standard
    - Crypto based on SIGMA protocols by Krawczyk
Outline

- Virtual Private Networks (VPN)
- IPsec Architecture and modes
- ESP and AH headers
- IKE
- IPsec and NAT/NAPT
(Virtual) Private Networks

- Private Network (Intranet): network / internet owned and operated by single organization.
  - Local area networks, routers, leased lines
- Virtual Private Network: set of networks and routers, with multiple owners of hosts, where:
  - Hosts of each organization communicate `as if` they are on private network / Intranet
  - Security, Quality Of Service
    - Sometimes: use of non-IP LAN protocols, e.g. IPX, over internet (use Layer 2 Tunneling Protocol – L2TP)
  - Extranet: same, but also across organizations
- Secure VPN: secure tunnel over insecure links
  - E.g. over the Internet
  - Can’t ensure QOS but can prevent eavesdropping, spoofing
Secure Connection (Tunnel): End-to-End vs. Hop-by-Hop

- Crypto protects traffic over insecure link/Net
- Link layer: one `hop` (e.g. wireless link)
- IP Layer (IP-Sec): transparent to application
- Transport Layer (SSL/TLS): easy, widely used
- Application Layer (PGP, S/MIME)
IP-Sec Implementation Options

1. **Native**: implement IP-sec as part of IP implementation in Operating System
   - E.g. in Windows 2000, XP

2. **BITS** (Bump In The Stack) – intercept IP traffic to/from network driver
   - Implementations on host w/o changing OS
     - E.g. Checkpoint’s firewall implementation

3. **BITW** (Bump In The Wire) – intercept IP traffic by tunneling via *security gateway*
   - Single gateway can protect multiple hosts
   - Only `tunnel mode` of IP-Sec…
IP-Sec transforms the Net to a Secure Virtual Private Network

Symbols

- Security gateway (BITW)
- Secure host (native or BITS)

Untrusted internet

Gateway to gateway

Host to gateway

Host to host

Single company: Secure VPN
Many companies: Secure Extranet
Secure VPN: use SSL/TLS or IP-Sec?

- Pros of IP-Sec:
  - Protection against DOS (clogging)
  - Protect all applications, data
    - Even (limited) support for multicast
  - Implemented by operating systems, Routers, ...

- Cons of IP-Sec:
  - Hard to implement
  - Often not available at destination
    - Compatible algorithms
    - Key management
  - No explicit session setup/teardown
    - Lose packets till key set, keep connection after done
Outline

- Virtual Private Networks (VPN)
- IPsec Architecture and modes
- ESP and AH headers
- IKE
- IPsec and NAT/NAPT
IP-Sec: IP Security Protocol

■ Characteristics
  - Connectionless, Unreliable
  - IP addresses can be easily spoofed
  - Routers and gateway might be sniffed

■ Requirements
  - Data source (IP address) authentication
  - Integrity protection
  - Replay protection
  - Access control (who can
  - Confidentiality and privacy
  - Denial Of Service (Clogging, DOS) prevention (Availability)
  - Support for IPv4, IPv6 (mandatory in IPv6)
IPsec Layers (RFC2401)

- Two **separate** layers
- IKE – Internet Key Exchange
  - At application layer
- IP-Sec record sub-layer: traffic encapsulation & protection
  - **AH** – Authentication Header (no secrecy)
  - **ESP** – Encapsulating Security Payload
  - Signal to IKE when detecting traffic that requires IP-sec but without established IP-sec connection
IP-Sec Modes of Operation

- **Tunnel Mode**
  - IP-Sec adds its own IP header
    - Original IP header is encapsulated (usually encrypted)
  - IP-Sec encapsulation/decapsulation either by hosts or by gateways along the route
    - IP addr of gateways in external header → may replace NAT

- **Transport Mode**
  - IP-Sec uses existing IP header, just changes protocol field to IP-Sec
  - Less overhead
  - Source/dest known (for access controls, quality of service)
  - End-to-end – IP-Sec encapsulation by source host, decapsulation by destination host (receiver)

- In both: IP-Sec adds its own header
**IPsec Tunnel Mode**

- Can be applied by Security gateways
  - But also by hosts (at one or both ends)
  - Traffic may be IP-Sec protected already (nested)
- Entire IP packet is payload to IPSEC
- If provided by gateway, transparent to host
- If encryption used, hides hosts’ IP address
Recall: TCP/IP Encapsulation

**Encapsulation**: receive & deliver payload (from/to higher layer), and add header and trailer

Source port, dest port, …

Source IP-addr, Dest IP-addr,…

Source net addr, Dest* net addr,…

(Encoding): receive & deliver payload (from/to higher layer), and add header and trailer
IP-Sec Encapsulation - tunnel mode

protocol=IP-Sec

Internet (IP)

Link (Ethernet)

Encrypted packet (incl. IP header)

IP header

IPsec header
IPsec Transport Mode

- Supplies end to end security services
- Modifies IP Header, Payload
  - Does not add another header!
- Requires IP-Sec support by both hosts: native or (at least) BITS (bump in the stack)
IP-Sec Encapsulation with transport mode

Application

msg

Transport (TCP)

TCP header

msg

IP-Sec sub-layer

IPsec header

Encrypted TCP segment

protocol=IP-Sec

Internet (IP)

IP header

IPsec header

Encrypted TCP segment

Link (Ethernet)

header

IP header

IPsec header

Encrypted TCP segment
Outline

- Virtual Private Networks (VPN)
- IPsec Architecture and modes
- ESP and AH headers
- IKE
- IPsec and NAT/NAPT
**IPsec Layers (RFC2401)**

- **IKE** – Internet Key Exchange
  - Negotiate cipher suite, options
  - Establish `Security Association`
  - Compare to SSL Handshake protocol; application layer
  - Run once per `IP-sec connection` - not `real-time`

- **IP-Sec record layer: traffic encapsulation & protection**
  - Compare to SSL record protocol
  - Between IP and Transport layers
  - **AH** – Authentication Header (no secrecy)
  - **ESP** – Encapsulating Security Payload
  - Signal to IKE when detecting traffic that requires IP-sec but without established IP-sec connection
IP-Sec Record Sub-Layer

- AH – Authentication Header (no confidentiality)
- ESP – Encapsulating Secure Payload – encryption and authentication (both may be `null`)
  - Do not use encryption w/o authentication [Bellovin96]
- Both use sequence numbers for FIFO, no-replay
- Both support multiple security associations (SA)
  - SA=Security parameters: keys, algorithms, counters…
  - Multiple SA between same peers – different protocols/ports
    - Separate SA for each user to prevent chosen text attacks!
  - Simplifies key update – just change SA
  - Identify SA by Security Parameter Index (SPI)
    - Each party selects its SPI for each security association
    - Send recipient’s SPI in each packet (32 bit)
Data Structures & Security Associations (SA)

- IPsec is using two data structures (define in the IPsec architecture RFC): SAD and SPD

  - **Security Association Data (SAD):** contains all the active Security Associations (SAs)
    - Incoming: access via SPI in packet
      - SPI in each direction selected by recipient (for efficiency)
    - Outgoing: access via `selectors` in packet
      - IP addresses, TCP/UDP ports and more
    - Built manually or by key management (IKE)

  - **Security Policy Data (SPD):** contains user defined policy. The user defines which security services, at which level are offered to each IP datagram
The Security Policy Data (SPD)

Contains a list of rules: \(<\text{TrafficSelector}, \text{action}>\)

- **Traffic Selectors**: IP addresses and/or ports (specific or range), protocol (TCP/UDP/?); optionally more

- **Actions**:
  - Discard
  - Bypass IPsec
  - Apply IPsec, specifying either (or both):
    - Security services, protocol, and algorithms
    - Pointer to the entry of matching active SA in the SAD

- **Packet-filtering (firewall) functionality**
  - Simple: integrate IP-Sec into firewall
  - Harder: end-to-end IP-Sec with firewall, NAT

- Defined by administrator or via (proprietary!) API
IPsec Replay Protection

- Sequence number zero when SA established
- Increment per outgoing packet
- Receiver identify replay by repeated seq #
  - Pass to higher layer out-of-order packets, but flag them in window (at least 32 positions) – since IP does not preserve order
- Sequence number field is sent and included in the MAC computation
  - In SSL/TLS it is not sent (why?)
- Must not wrap during a single key lifetime
  - Keys must be changed after $2^{32}$ packets
Authentication Header (AH)

- Inserted after the IP header
- AH protocol number (in IP header) is 51

![Diagram of AH header]

**Protocol=51**

- **IP Header**
  - Next protocol
  - Payload length
  - Reserved
  - SPI (Security Parameters Index)
  - Sequence number
  - Authentication data (variable)

**Original protocol (now protocol: AH)**

**Replay Protection…**

**MAC over entire IP packet (IP header, other headers, data…)**

- 32 bits
ESP - Encapsulating Security Payload

32 bits

- SPI
- Payload data
- Padding
- (0-255) bytes
- Pad length
- Next protocol
- Authentication data (variable)
- Initialization Vector (if needed for encryption)
- Sequence number
- Why MAC the SPI?
- Simpler…and why not?
- Why MAC the SPI?
- Why MAC the SPI?

IP/UDP/TCP…

(c) Amir Herzberg, http://AmirHerzberg.com
Outline

- Virtual Private Networks (VPN)
- IPsec Architecture and modes
- ESP and AH headers
- IKE
- IPsec and NAT/NAPT
Recall: IPsec Layers

- **IP-Sec record layer**
  - Traffic encapsulation & protection
  - Between IP and Transport layers
  - AH – Authentication Header (no secrecy)
  - ESP – Encapsulating Security Payload

- **IKE – Internet Key Exchange**
  - Application protocol to establish and manage SA (security association, e.g. keys) for IP-Sec
    - Compare to SSL Handshake protocol
    - Negotiate cipher-suite, options
  - Invoked by IP-Sec record layer when rules say packet should be secured, but no SA in place
  - Run once per `IP-Sec connection` - not `real-time`
Introduction to IKE

- IKE is a mature, complex protocol for securely setting up keyed sessions, in particular IP-Sec Security Associations (SA)
- IKE evolved over several years from multiple proposals; IKEv2 is now `draft standard`
  - Discussion relate to v1 and v2, except when noted
- IKE runs over UDP (port 500; detect NAT: 4500)
  - One IKE message per UDP datagram
- Uses (only) exchanges (request/response)
  - *Initiator (Alice)* makes request, *Responder (Bob)* responses
  - Initiator (only) retransmits/aborts for reliability
  - Not necessarily client/server! But usually Alice is client.
IKE `advanced` features (design goals)

- Cryptographic negotiation
  - Efficient, secure, robust, flexible

- Identity Hiding
  - IKEv1: of Alice (usually client);
  - IKEv2: of Bob against MITM, both against spoofing
  - Or: hide both – only if using manually shared keys

- Robustness against Denial Of Service

- NAT/NAPT-friendly

- Strong (Perfect) Forward Secrecy (PFS)
  - What’s this?
Strong (Perfect) Forward Security (PFS)

- Protect traffic of period $i$ from exposure of all keys of all periods $j \neq i$, as long as exposure happens after (refresh phase of) period $i+1$.
- Active adversary - can always inject/eavesdrop etc.
- Motivation: attacker may eventually expose some old keys, by cryptanalysis, reading erased data,…

![Diagram showing the timeline of periods and refresh phases with an attack time point]
IP-Sec Setup: IKE (Internet Key Exchange) ver1

- Two phases
  - 1\textsuperscript{st} phase: setup ISAKMP SA (Internet Security Association and Key Management Protocol)
    - Algorithms, keys, etc. – to be used by IKE (not AH/ESP!)
    - Perfect forward secrecy (PFS): exposure of all keys does not expose past traffic [using Diffie-Hellman]
  - 2\textsuperscript{nd} phase: Generate IP-Sec SA
    - Protected using the ISAKMP SA
    - Many 2\textsuperscript{nd} phases may share ISAKMP SA (1\textsuperscript{st} phase)
      - E.g. one 1\textsuperscript{st} phase for gateways, then many 2\textsuperscript{nd} phase for each pair of hosts using these gateways
    - More efficient than 1st phase; PFS optional
Why derive many session keys?

- Why not establish and use one `master key`?
- Ensure reliable, secure separation of sessions
  - In particular prevent IP spoofing in ESP/Transport
- Restrict use of a single key
  - Make cryptoanalysis harder
    - Less available ciphertext
    - Some sessions may be easier to attack (chosen/known plaintext)
  - Restrict damage of known key attack: session key exposure does not expose past or future messages, session keys, or master key
- Strong (Perfect) Forward Secrecy (PFS)
Why Two IKE Phases?

- To fulfill the PFS requirement, every phase I exchange, performs a DH exchange
- In phase II, DH execution is optional – phase II and the IPsec keys can be derived from phase I exchange
- Phase II is more efficient
- Many phase II exchanges can use the same set of phase I keys
- Why derive different keys and not
Identification of Peer in IKE

- 1\textsuperscript{st} phase – identify using one of:
  - Manually pre-shared secret key
  - Exchanged public key certificates, and…
    - Public key signatures, or
    - Public key encryption of challenge (two variants) – only in IKE version 1.

- 2\textsuperscript{nd} phase – trusts identification in 1\textsuperscript{st} phase
  - Uses identities from SPD
IKE Denial Of Service Attacks

- IKE DOS Attack: flood victim with IKE requests (fake source IP addr) $\Rightarrow$ victim performs expensive computations in vain
- Solution: before performing expensive computations (e.g. DH), verify that the other party is indeed located in the IP address that appears in the header
- How? Cookies mechanism… (next)
- Note: requires the `main mode` of IKEv1 (6 flows, cf. to `aggressive mode of 3 flows), also optional exchange in IKEv2; usually invoked only on attack
The Cookies Mechanism

- The recipient sends a pseudo random string (Cookie) to the other party
- The other party return the cookie, proving it can receive from its IP address
- Compute cookie by MAC with local secret key $k$ of $x=IP$ address, UDP ports, date and hour
  - Secure assuming hash is a true random function
- Efficient generation, memory less verification
- Expensive calculations will be performed, and state kept, only if valid cookie is received
IKEv2 Exchanges

- **Initial Exchanges (cf. phase I)**
  - IKE_SA_Init exchange: Negotiate crypto-suites, exchange keys (DH) and nonces
  - IKE_Auth exchange: authenticate IKE_SA_Init exchange, exchange identities, certificates and traffic selectors, and establish 1st child SA
- **Create_Child_SA exchange (cf. phase II)**
  - Create_Child_SA exchange: Identify new SA (and, if rekeying, existing SA); Exchange nonces, and optionally keys (DH) and traffic selectors

Traffic Selector: IP addresses and/or ports (specific or range), protocol (TCP/UDP/*…)

(c) Amir Herzberg, http://AmirHerzberg.com
IKEv2: IKE_SA_Init exchange

- Negotiate crypto-suites
- Exchange $g^i$, $g^r$ (Diffie-Hellman public values)
- Exchange nonces
- Identities (and certificates) not exposed yet!

Diagram:
- HDR, SAi1, KEi, Ni
- SPIi, Version, flags,…
- `cipher suites`
- DH-group IDs, $g^i \text{ mod } p$
- `$g^r$
- initiator’s nonce
- responder’s nonce
- HDR, SAR1, KEr, Nr, [CERTREQ]
- SPIr, Version, flags,…
- `cipher suites`
- Trust anchors (root CA’s)
Keys Derivation in IKE

IKE_SA_Init exchange:
public values $g^i, g^r$ and nonces $n_i, n_r$

Derive `master key`:
$$SKEYSEED = PRF(N_i, N_r, g^{ir})$$

Derive IKE keys from `master key`:
$$\{SK_d, SK_{ai}, SK_{ar}, SK_{ei}, SK_{er}\} = prf+(SKEYSEED, N_i, N_r, SPI_i, SPI_r)$$

Derive child SA keys from `master key`:
$$\text{KeyMaterial} = prf+(SK_d, g^{ir}, N_i, N_r, SPI_i, SPI_r)$$

$SK$: secret keys:
_d: Derive child SA keys
_ai, _ei: initiator MAC, encrypt
_ar, _er: respond. MAC, encrypt

Why use public values as PRF key? [next foil]
Deriving `master key` \( k \) (in spec: SKEYSEED)

- **Goal:** derive `master key` from \( g^i, g^r, n_i, n_r \)
  - Nonces allow parties to reuse same exponents \( i, r \)
  - Also: nonces are used to extract random key… How/Why?
- If \( k = g^{ir} \) was random… we could have used \( prf_k(n_i|n_r) \)
  - But \( g^{ir} \) is not random!
- Heuristic: use \( h(g^{ir}) \rightarrow \)random oracle\` security
- Better: \( uh_n(g^{ir}) \) where \( uh \) is a universal hash [CW79] and \( n \) is random (nonce)

- **IKE version:**
  - Use \( k = prf_{n_i\|n_r}(g^{ir}) \) instead of \( uh_n(g^{ir}) \)
  - Using \( prf \) instead of \( uh \): heuristic to simplify coding
  - Using \( n_i\|n_r \): should be random since the parties should use random nonces (see also the notes to this foil).
IKEv2: IKE_Auth exchange

- Authenticate IKE_SA_Init exchange
- Exchange identities and certificates (encrypted for privacy – but client identity has weaker protection)
- Exchange traffic selectors
- Establish 1st child SA
- Encrypted and authenticated (MAC) using SK \{ \}
  - Like in ESP: encrypt then MAC; use keys SK\_[a/e][i/r].
Outline

- Virtual Private Networks (VPN)
- IPsec Architecture and modes
- ESP and AH headers
- IKE
- IPsec and NAT/NAPT
NAT & IP-Sec

- NA(P)T connects `private` internet to the Internet, with separate address spaces
  - NAT: temporary mapping from private addresses to Internet addresses
  - NAPT: temporary mapping from private <IP, port> to Internet <IP, port>, using random ports
- Mapping could `break` IP-Sec traffic
- NA(P)T are common in home, corporate and temporary (hotel, café) networks
- Interoperability necessary for IPsec VPNs!!
NAT & NAPT
(Network Address (& Port) Translation)

rest of Internet

local network (e.g., company network) 10.0.0.0

138.76.29.7

10.0.0.4

Datagrams with source or destination in this network have 10.0.0.* address for source, destination (as usual)

All datagrams to/from local network have Internet source IP address, e.g.: 138.76.29.7.

NAT: from pool of IP addresses
NAPT: one/few IP addresses; different source port numbers identify sender, port

(c) Amir Herzberg, http://AmirHerzberg.com
IP-Sec and NAT/NAPT Issues

1. AH MAC includes IP header, changed by NA(P)T ➔ MAC verification fails
2. UDP/TCP checksum & Transport mode: TCP and UDP checksum cover the IP addresses; NA(P)T devices recalculate checksum, but can’t after IP-Sec ➔ receiving TCP/UDP drop the packet
   ✷ Not in Tunnel mode, which does not change IP header
3. IKEv1 insisted on using port 500 both ways.
   ✷ Some NAPT try to support by not `touching` this port.
Common solution to NA(P)T interoperability, invoked if NA(P)T detected (by Notify payload source/dest IP).

Use UDP with both ports=4500

IKE and ESP only (why?)
- Identify IKE by `0000`x word (in the SPI field; avoid such SPI!)

ESP: Tunnel or Transport

IKE uses port 4500 (to avoid special treatment of port 500)

RFC 3948: UDP Encapsulation of IPsec Packets
Few exercises

- An organization connects to the Internet from multiple offices, but concerned about:
  - Denial of service attacks from the Internet
  - Protect data on few key applications (mostly web) from unauthorized exposure
  - Efficiency and cost of solution
- IP-Sec uses connections. In what ways are these connections not reliable?
- Consider extranet btw 3 companies, using mostly web services. Present and compare SSL and IPSec designs.
- Few companies create extranet (shared VPN over Internet) using IP-sec. How can they (securely and efficiently)…
  - Prevent network attacker from counting CEO-to-CEO messages
  - Prevent insiders from eavesdropping on CEO-to-CEO messages
Conclusion

- IP Security protocol protects all Internet traffic
- Tunnel mode allows gateways to protect many hosts
- Transport mode allows efficient host-to-host security
  - Possible interoperability problems w/ NAT (can tunnel over UDP)
- Support for authentication (AH/ESP) and encryption (ESP)
- Tunneling of IP-sec protected traffic is possible
  - E.g. to hide identities of source/destination hosts behind gateway
- Flexible policy for security
  - Block, allow (unprotected) or protect traffic
  - Defined on host/port basis – not per user!
- Resiliency to clogging (in IKE via cookies)
- Requires no change in applications
- Hard to implement, interoperate