IC3 - Network Security

M.Sc. in Information Security
Royal Holloway, University of London

IC3 - Network Security

Lecture 5
Secure Protocols – IPSec

Objectives of Lecture

- Revisit the “secure channel” concept from Lecture 4.
- Understand the pros and cons of providing security at different network layers.
- Investigate how IPSec provides security at the Internet layer.
- Study major applications of IPSec in Virtual Private Networking and secure remote access.

Contents

5.1 The “Secure Channel” Concept

- We need to guarantee the confidentiality, authenticity and integrity of data travelling over insecure networks
- Applications:
  - Branch office connectivity
  - Connecting to business partners at remote site
  - Remote access for employees
  - Remote administration of network devices and servers
  - E-commerce: protecting credit card numbers in transactions
  - E-government: electronic voting, filing tax returns

The “Secure Channel” Concept

- We achieve this by building a “secure channel” between two end points on an insecure network
- Typically this channel will offer:
  - Data origin authentication
  - Data integrity
  - Confidentiality
- But usually not:
  - Non-repudiation
  - Any security services once data has been received

The "Secure Channel" Concept

- Secure channels are usually constructed as follows:
- An authenticated key establishment protocol
  - During this one or both parties is authenticated.
  - A fresh, shared secret is established.
  - Optional features: anti-DoS, identity-protection, perfect forward secrecy,…
  - May use asymmetric (public key) or symmetric cryptography, or a combination of the two
- Key derivation phase
  - MAC & bulk encryption keys are derived from shared secret
- And then further traffic protected using derived keys
  - MAC gives data integrity mechanism and data origin authentication
  - Encryption gives confidentiality
  - Use symmetric cryptography for speed
  - Optional optimizations: Session re-use, fast re-keying, …

Typical Cryptographic Primitives Used

- Symmetric encryption algorithms
  - Almost universally used for performance reasons
- MAC algorithms
  - Usually built from hash functions, block ciphers, or possibly a dedicated design
  - Moderate to low computational complexity
- Asymmetric encryption and signature algorithms, Diffie-Hellman
  - For entity authentication and key exchange (as in Lecture 4)
- (Keyed) pseudo-random functions
  - For key derivation
  - Generally built from hash functions

Other Common Techniques Used

- Sequence (and Lamport clocks) numbers are widely used to prevent replay attacks and ensure correct data ordering
  - These need to be cryptographically protected
- Nonces and timestamps used to provide freshness in entity authentication exchanges

5.2 Security and Network Layers

- Where to place security functionality in the OSI protocol stack?
- Security can be applied at any of the network layers except layer 1 (Physical layer).
  - Even this is sometimes possible, e.g. spread spectrum techniques can provide limited (traffic flow) confidentiality
- What are the pros and cons of applying security at each of these layers?

Security and Network Layers

- Data Link (Network Interface) layer:
  - ✓ Covers all traffic on that link, independent of protocols above.
    - ✓ e.g. link level encryptor (Lecture 2)
    - ✓ Cannot be compromised even if communicating hosts are
  - ✓ Typically runs at line-speed of link
  - ✗ Protection only for one "hop" (point-to-point)
    - ✗ Doesn't scale well, but sometimes it's the only option
  - ✗ Usually implemented using moderate to high cost special-purpose hardware

Security and Network Layers

- Network (Internet) layer:
  - ✓ Covers all traffic carried by IP
  - ✓ Can be end-to-end (or not!)
  - ✓ Transparent to applications
  - ✓ Cost of authentication/key exchange protocols can be amortized over many applications
  - ✗ Little application control over security that gets applied.
    - ✗ Application has no visibility of Internet layer
    - ✗ Security provided may be overkill or "underkill"
  - ✗ May be unnatural place to apply security
    - ✗ Network layer is stateless and unreliable
    - ✗ Detecting and preventing replays therefore technically impossible, without maintaining extra state or via a layer violation
    - ✗ Unreliability of IP makes provision of availability a challenge
    - ✗ Order of data in secure channel may be crucial; difficult to maintain if IP datagrams are dropped, re-ordered,…

Security and Network Layers

- Transport layer:
  - ✓ End-to-end protocol
  - ✓ Covers all traffic using the protected transport protocol
  - ✓ Applications can control when it's used
    - ✓ Application can choose to select secure transport layer or not
  - ✓ Transport layer may be naturally stateful (TCP)
    - ✓ Makes provision of some security services easier
  - ✗ Each application must be modified or proxied to take advantage of the security provided by secure transport layer option
    - ✗ Compromised or misconfigured applications and systems may disable or weaken security mechanism
  - ✗ May be protocol-specific
    - ✗ E.g. SSL/TLS only implemented over TCP, not UDP

Security and Network Layers

- Application layer:
  - ✓ Security can be tuned to application requirements
    - ✓ Different applications may have radically different needs
      - e.g. VoIP applications vs. sensitive data transfer
  - ✓ Easy access to user credentials (e.g. private keys)
  - ✓ Possible to provide non-repudiation services at application level
    - ✓ May not make sense at lower layers
  - ✗ But no leveraging effect
    - ✗ Every application must handle its own security
    - ✗ Plenty of room for errors, redundancy, and security holes

5.3 IPSec

- IPSec basic features
- IPSec transport and tunnel modes
- AH – authentication and data integrity
- ESP – confidentiality
- IPSec policy and Security Associations
- Combining Security Associations
- Key management in IPSec: IKE

IPSec Basic Features

- IPSec provides security at network (Internet) layer
  - So all IP datagrams can be covered
  - No re-engineering of applications required
  - Transparent to users (apart from some key management aspects)
- Mandatory for next-generation IPv6, optional retro-fit for current-generation (IPv4)
- Core definition in IETF RFCs 4301-4309 (2005)
  - Revisions of original specifications in RFCs 2401–2412 (1998)
  - Warning: Not necessarily for the faint-hearted
  - Consult "IPSec" by N. Doraswamy and D. Harkins, 2nd ed. (Prentice Hall, 2003).
  - Further RFCs define later enhancements (new cipher suites, etc.)

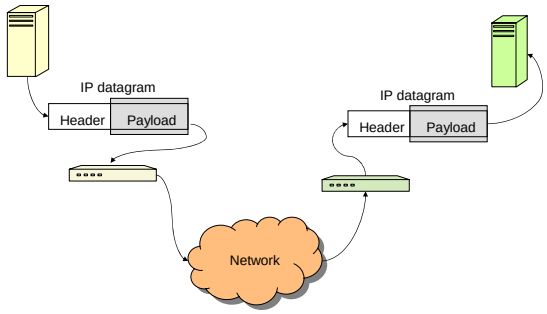
IPSec Basic Features

- IPSec provides two basic modes of use:
  - Transport mode: for IPSec-aware hosts as endpoints
  - Tunnel mode: for IPSec-unaware hosts, established by intermediate gateways or host OS
- IPSec provides authentication and/or confidentiality services for data
  - AH and ESP protocols
- AH and ESP can each be applied multiple times (in tunnel or transport mode) to a given datagram
  - IPSec policies will define how and when this is done
- IPSec provides (overly?) flexible set of key establishment methods:
  - IKE, derived from Oakley and SKEME protocols
  - Operating within ISAKMP framework
  - IKEv2 (RFC 4306, Dec. 2005) not yet widely deployed
  - Copious interoperability headaches owing to vague and fuzzy layered specifications leaving plenty of room for misinterpretation

IPSec Transport Mode

- Protection for upper-layer protocols
- Protection covers IP datagram payload (and selected header fields)
  - Could be TCP packet, UDP, ICMP message,….
- Host-to-host (end-to-end) security:
  - IPSec processing performed at endpoints of secure channel
  - So endpoint hosts must be IPSec-aware for transport mode

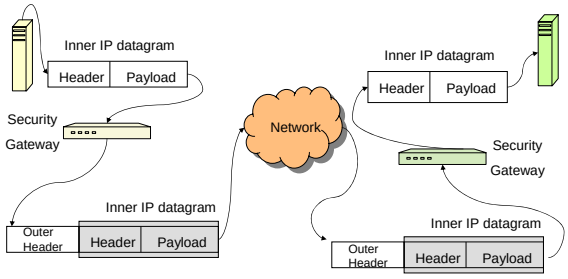
IPSec Transport Mode



IPSec Tunnel Mode

- Protection for entire IP datagram
- Entire datagram plus security fields treated as new payload of "outer" IP datagram
- So original "inner" IP datagram is *encapsulated* within "outer" IP datagram
- IPSec processing is performed at *security gateways* on behalf of endpoint hosts
  - Gateway could be perimeter firewall or router (e.g. also with hardware support for IPSec offloading)
  - Gateway-to-gateway rather than end-to-end security
  - Hosts need not be IPSec-aware
- Intermediate routers have no visibility of inner IP datagram
  - Even orginal source and destination addresses encapsulated and so "hidden"

IPSec Tunnel Mode



AH Protocol

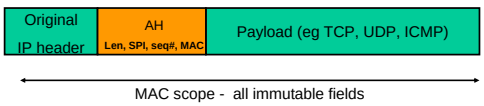
- AH = Authentication Header (RFC 4302)
- Provides data origin authentication and data integrity services
- AH authenticates whole payload and most of header
- Prevents IP address spoofing
  - Source IP address is authenticated
- Creates stateful channel
  - Use of sequence numbers
- Prevents replay of old datagrams
  - AH sequence number is integrity protected
  - Recipient tracks sequence numbers of arriving packets
  - Reject repeats and packets that are too old
- Uses MAC and symmetric key shared between endpoints

AH Protocol

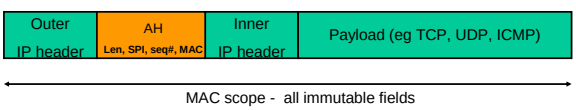
- AH specifies a header added to IP datagrams
- Fields in header include:
  - Payload length
  - SPI = Security Parameters Index
    - Identifies which algorithms and keys are to be used for IPSec processing (more later)
  - Sequence number
  - Authentication data (the MAC value)
    - Calculate over immutable IP header fields (so omit TTL) and (payload or inner IP datagram)

AH Protocol – Transport and Tunnel

AH in transport mode:



AH in tunnel mode:



ESP Protocol

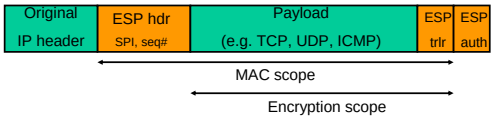
- ESP = Encapsulating Security Payload (RFC 4303)
- Provides one or both of:
  - Confidentiality
    - Protection for payload in transport mode and inner datagram in tunnel mode
    - Sequence number is not protected by encryption
  - Authentication/integrity protection
    - Protection for payload in transport mode and inner datagram in tunnel mode
    - But header fields (original header or outer header) are unprotected
- Gives limited traffic-flow confidentiality in tunnel mode
- Uses symmetric encryption and MACs based on secret keys shared between endpoints

ESP Protocol

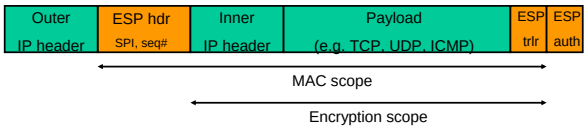
- ESP specifies a header and trailing fields to be added to IP datagrams
- Fields in header include:
  - SPI
  - Sequence number
- Fields in trailer include:
  - Any padding needed for encryption algorithm (may also help disguise payload length)
  - Padding length
  - Authentication data (if any) – the MAC value

ESP Protocol – Transport and Tunnel

ESP in transport mode:



ESP in tunnel mode:



AH and ESP Algorithms

- IPSec supports the use of a number of algorithms for ESP and AH
  - Standard was designed to be flexible and extensible
- ESP:
  - DES, three-key triple DES, AES, Blowfish, etc.
  - Each algorithm needs its own RFC
  - E.g. use of DES for ESP defined in RFC 2405 (since deprecated in RFC 4308); while RFC 2410 specifies the "null encryption algorithm"!
  - New RFCs summarize some mandatory cipher suites (RFC 4308)
- AH:
  - HMAC-MD5-96, HMAC-SHA-1-96,…

Integrity Protection in AH and ESP

- Separate existence of authentication/integrity protection in both AH and in ESP for performance, backwards-compatibility, and political (!) reasons
  - Original version of ESP (RFC 1827) had no integrity protection mechanism
  - So two IPSec processing steps needed to provide both confidentiality and integrity protection services
  - IETF decided to incorporate integrity protection directly into second version of ESP RFC (RFC 2406) for efficiency reasons
- Integrity protection has different scope in ESP and AH

Sequence Numbers in IPSec

- Both ESP and AH use sequence numbers to provide an anti-replay service
- Sequence numbers are 32 bits long
  - Initialised to zero
  - Increment on datagram-by-datagram basis
  - Overflow results in auditable event and re-keying
- Protected by MACs in AH and ESP
  - But no protection afforded to sequence numbers when ESP (confidentiality only) is used
- Recipient uses "sliding window" to track datagram arrivals

Sequence Numbers in IPSec

- Sliding windows:
  - Window indicates which sequence numbers have already been seen
  - Each newly arrived sequence number is compared to the entries in the current window
  - If new sequence number to left of window or already received, reject the packet (too old or replay)
  - If new sequence number to right of current window, move the window to the right to accommodate it and accept the packet
  - Otherwise, mark the corresponding entry in the window and accept the packet
- Recommended window width is 64.
  - Datagrams can be dropped if delayed too long (by network latency or deliberately)

IPSec Security Policy

- How does IPSec determine what security processing is to be applied to IP datagrams?
- IPSec-aware host has a Security Policy Database (SPD)
  - A required part of IPSec implementations
  - SPD is populated by network adminstrators
- The SPD is consulted for each out-bound and in-bound datagram
- Fields in IP datagram compared to fields in SPD entries to find matches
  - Match can be based on source and dest addresses (and ranges of addresses), transport layer protocol, transport layer port numbers,…
- Each match identifies a *Security Association* (SA) or group of SAs (or the need for a new SA)

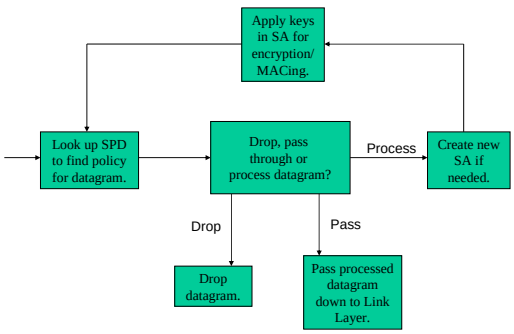
IPSec Security Associations (SAs)

- Each SA defines a set of algorithms, mode (tunnel or transport) and keys to be used to process a datagram
- An SA is a one-way (simplex) relationship between a sender and receiver
  - Specifying some cryptographic processing to be applied to *this* datagram from *this* sender to *this* receiver
- SAs are held in the SA database (SADB)
  - Collection of active SAs
  - A required part of IPSec implementations

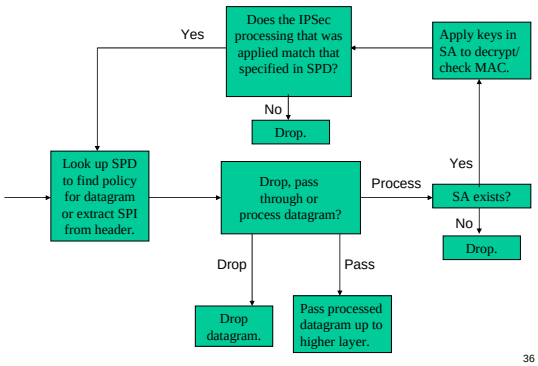
IPSec Security Associations (SAs)

- SAs are established manually or as needed by IKE (see later)
- Each SA is identified by a unique SPI (32 bit value carried in AH and ESP headers)
  - Allows recipient to determine how to process received datagrams
- Each SA contains:
  - Sequence number counter/sliding window
  - AH/ESP info: algorithms, IVs, keys, key lifetimes
  - SA lifetime (soft and hard, bytes processed and/or time)
  - Protocol mode: tunnel or transport
  - Tunnel destination for tunnel mode
  - Path Maximum Transfer Unit (PMTU)

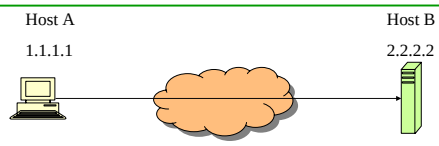
IPSec Out-bound Processing



IPSec In-bound Processing



SPDs and SAs in Action



A's SPD:

From	To	Protocol	Port	Policy	SADB pointer
1.1.1.1	2.2.2.2	TCP	80	Transport ESP with 3DES	

A's Outbound SADB:

From	To	Protocol	SPI	SA record
1.1.1.1	2.2.2.2	ESP	10	3DES key

SPDs and SAs in Action

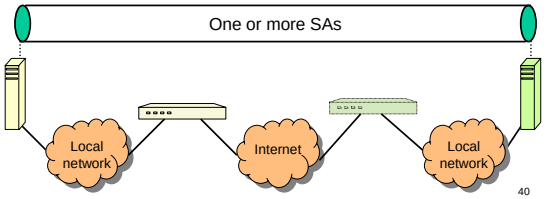
- IPSec processing is evidently complex and may have impact on network throughput
  - Realistic IPSec policy can lead to nested levels of IPSec processing
  - Outbound processing requires matching of traffic selectors to entries in SPD, then SADB look-ups
  - In-bound processing can use SPIs to index directly to SADB
- Careful design and implementation of SPD and SADB is then necessary
  - Need efficient look-up mechanisms for matching traffic to SPD
  - Use cache of pointers to SADB in SPD
  - Use cache of pointers to SPD entries in socket data structure for connection-oriented communications (TCP)

Combining SAs

- Often, we want security services provided by both ESP and AH, and may want to provide them at different points in network
  - ESP only allows MAC after encryption; we may desire reverse
  - May desire AH in transport host-to-host tunnelled inside ESP gateway-to-gateway for Virtual Private Network (VPN)
  - Using encryption without some form of authentication/integrity protection in IPSec is extremely dangerous
- SAs can be combined using either:
  - **Transport adjacency**: more than one SA applied to same IP datagram without tunnelling
    - Essentially AH + ESP
  - **Iterated tunnelling**: multiple levels of nesting of IPSec tunnels; each level with its own SA
    - Each tunnel can begin/end at different IPSec site along route

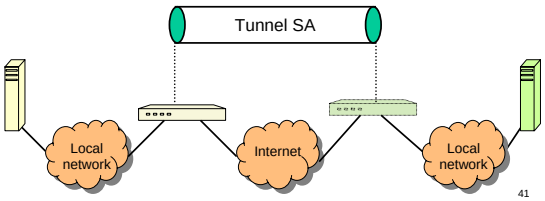
Required SA Combinations

1. End-to-end application of IPSec between IPSec-aware hosts:
   - One or more SAs, one of the following combinations:
     - AH in transport
     - ESP in transport
     - AH followed by ESP, both transport
     - Any of the above, tunnelled inside AH or ESP



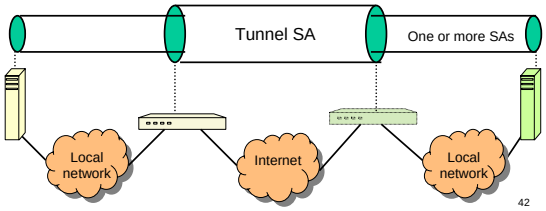
Required SA Combinations

2. Gateway-to-gateway only:
   - No IPSec at hosts
   - Simple Virtual Private Network (VPN)
   - Single tunnel SA supporting any of AH, ESP (confidentiality only) or ESP (confidentiality + authenticity)



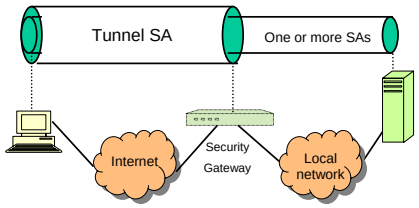
Required SA Combinations

3. A combination of 1 and 2 above:
   - Gateway-to-gateway tunnel as in 2 carrying host-to-host traffic as in 1
   - Gives additional, flexible security on local networks (between gateways and hosts)
   - e.g. ESP in tunnel mode carrying AH in transport mode



Required SA Combinations

4. Remote host support:
   - Single gateway (typically firewall)
   - Remote host uses Internet to reach firewall, then gain access to server behind firewall
   - Traffic protected in inner tunnel to server as in case 1 above
   - Outer tunnel protects inner traffic over Internet



IPSec Key Management

- IPSec is a heavy consumer of symmetric keys:
  - One key for each SA
  - Potentially, different SAs for every combination from:
    {ESP,AH} x {tunnel,transport} x {sender, receiver} x {protocol} x {port}
- Where do these SAs and keys come from?
- Two sources:
  - Manual keying
    - Fine for small number of nodes and testing purposes
    - Hopeless for reasonably sized networks of IPSec-aware hosts
  - IKE: Internet Key Exchange, RFC 2409 (v1), RFC 4306 (v2).
    - RFC documentation can be a bit hard to follow
    - IKE is an adaptation of other protocols (Oakley and SKEME) within the framework of another specification (ISAKMP)
    - Protocols have many options and parameters
  - IKEv2 can be assumed to have similar lengthy teething problems
    - Intended to address problems and complexities of IKEv1, but…

IKE Security Goals (IKE v1)

- Entity authentication of participating parties
- Establishment of a fresh, shared secret
  - Shared secret used to derive further keys
  - For confidentiality and integrity protection of IKE management channel
  - For SAs for general use
- Limited resistance to Denial-of-Service attacks
  - Using cookie mechanism
- Secure negotiation of all parameters and algorithms
  - Authentication method, key exchange method, Diffie-Hellman group, algorithms for encryption and MAC, hash algorithms
- Options for Perfect Forward Secrecy, Deniable Authentication and Identity Protection

IKE Phases

IKE operates in two phases:

- **Phase 1**: Negotiate a special SA, the IKE SA, along with keying information
  - IKE SA specifies encryption and MAC algorithms for use in constructing a secure channel used in Phase 2
  - IKE SA also specifies authentication method and Diffie-Hellman parameters to be used in Phase 1
    - Authenticated Diffie-Hellman key exchange used to establish keying information for use in Phase 2 secure channel
  - Collection of algorithms and data called a *protection suite*
  - IKE SA is bi-directional and contains somewhat different information to "normal" IPSec SAs

IKE Phases

- **Phase 2**: SAs for general IPSec use are negotiated
  - Phase 2 uses a secure channel to perform further SA negotiation
  - Algorithms for this secure channel are defined by the IKE SA agreed in Phase 1
  - Keys are derived from the Diffie-Hellman exchange in Phase 1
  - Phase 2 can also be used for secure transport of error and management traffic
  - Many Phase 2 runs allowed for each run of Phase 1; multiple SAs can be negotiated per run
  - The result is fast and cheap negotiation of IPSec SAs in Phase 2

IKE Phase 1

- Phase 1 is the heavyweight exchange to establish a secure channel for Phase 2; two variants:
  - "Main mode": slow (6 messages), more cautious, hides details of credentials used, provides (limited) anti-DoS service
  - "Aggressive mode": less negotiation, only 3 messages, more information disclosed
- Each of main and aggressive mode allows 4 different authentication mechanisms:
  - Signature, public-key encryption, revised public-key encryption, pre-shared key (symmetric)
  - Nonces for freshness
  - Certificates for authenticity of public keys
- Chosen mechanism used to authenticate a Diffie-Hellman key exchange
  - In one of 5 different fixed groups or using "new group mode"

IKE Phase 1 Main Mode Example

We illustrate Phase 1 main mode using "authentication with signatures" (simplified!)
(i=Initiator, r=Responder, […]=optional field)

1. I→R: HDR_i, SA_i
2. R→I: HDR_r, SA_r
3. I→R: HDR_i, KE_i, N_i [,Cert_Req]
4. R→I: HDR_r, KE_r, N_r [,Cert_Req]
5. I→R: HDR_i, $\{ID_i, [Cert_i,] Sig_i\}_{SKEYIDe}$
6. R→I: HDR_r, $\{ID_r, [Cert_r,] Sig_r\}_{SKEYIDe}$

Explanation

Messages 1 and 2:

- I and R exchange cookies CKY-I, CKY-R (contained in HDR_i, HDR_r fields)
  - Cookies provide limited anti-DoS measure (details later)
- I and R also exchange lists of preferred/accepted IKE SAs (in SA_i, SA_r fields), these are also known as protection suites
  - These specify algorithms for use in Phase 2 and authentication methods and Diffie-Hellman parameters for use in remainder of Phase 1

Messages 3 and 4:

- I and R exchange Diffie-Hellman values ($KE_i = g^x$, $KE_r = g^y$) and nonces (N_i, N_r), request certificates
- I and R also re-exchange cookies to complete anti-DoS feature

Explanation

- Messages 5 and 6:
  - I and R exchange identities, certificates, and signatures
  - These exchanges are encrypted by a key $SKEYID_e$ derived from Diffie-Hellman values and nonces
  - Signatures are on hash of string formed by concatenating Diffie-Hellman values, nonces, SA_i, SA_r,…)
  - Signatures on fresh values (nonces) provide mutual entity authentication
- Compare this protocol with Station-to-Station protocol from Lecture 4

Features of Main Mode

- Identity protection
  - ID_i, ID_r and Certs only ever transported in encrypted form.
- Anti-Denial of Service via CKY-I and CKY-R
  - I and R do not perform expensive computations until an exchange of cookies has taken place
  - Prevents rudimentary DoS based on address spoofing
  - Attacker spoofing I's IP address will not receive cookie from R in message 2 and cannot guess correct response for CKY-R in message 3
  - Likewise, attacker spoofing R's address will not possess correct CKY-I for inclusion in message 2
- Secure negotiation of protection suites
  - SA_i and SA_r included in signatures
  - Prevents attacker spoofing messages to force I and R to agree on weakest common algorithms

Aggressive Mode

- Aggressive mode sacrifices identity protection, flexibility in protection suite negotiation and anti-DoS feature to gain faster execution
  - 3 messages instead of 6
  - I provides list of protection suites, identity, Diffie-Hellman value and nonce in first message
  - R selects one suite, and responds with choice together with his identity, Diffie-Hellman value and nonce. Also includes authentication payload (e.g. a signature)
  - I responds with his authentication payload in third message

Deriving Keys From Phase 1

- Phase 1 agrees Diffie-Hellman key g^{xy}
- Further keys derived from this keying material:
  - $SKEYID = prf(N_i | N_r, g^{xy})$ (for signature-based authentication)
  - $SKEYID_d = prf(SKEYID, g^{xy} | CKY\text{-}I | CKY\text{-}R | \text{“0”})$
  - $SKEYID_a = prf(SKEYID, SKEYID_d | g^{xy} | CKY\text{-}I | CKY\text{-}R | \text{“1”})$
  - $SKEYID_e = prf(SKEYID, SKEYID_d | g^{xy} | CKY\text{-}I | CKY\text{-}R | \text{“2”})$
- $SKEYID_a$: key for MAC in Phase 2
- $SKEYID_e$: key for encryption in Phase 2
- $SKEYID_d$: also used to derive further keys for IPSec SAs exchanged in Phase 2

IKE Phase 2

- Only one form for Phase 2, also called Quick Mode
- Either I or R can initiate Phase 2 protocol run
- Uses algorithms and keys agreed in Phase 1 to protect IPSec SA exchanges in Phase 2
  - Can have many Phase 2 runs over this secure channel
  - Can propose/accept multiple SAs in one Phase 2 protocol run
  - Spreads cost of heavy-weight Phase 1
  - Uses only symmetric techniques (MAC and encryption algorithms)

IKE Phase 2

- Uses fresh nonces to provide entity authentication
- Uses $SKEYID_d$ from Phase 1 to define keys for exchanged IPSec SAs
  - Option to include new Diffie-Hellman exchange in Phase 2 runs for higher security
  - This provides property of perfect forward secrecy (PFS), but slower to execute
  - PFS: even if Phase 1 secrets later compromised, keys in IPSec SAs exchanged in Phase 2 will still be secure

IKE Phase 2

Basic structure of IKE Phase 2:

- I sends list of proposed SAs, nonce and optional fields: DH value; identity information
- R responds with accepted SAs, nonce and optional fields: DH value; identity information
  - Both flows integrity protected and encrypted
  - Identity information provides traffic selectors for populating SADBs of I and R
- I closes with message providing entity authentication to R
  - Essentially a MAC on R's nonce

Further IKE Exchanges

- IKE information exchange
  - For transmission of status and error messages
  - Example: notify a peer that an SA has been deleted
  - Carried in single, unacknowledged message
- IKE new group exchange
  - Allows peers to negotiates private parameter sets for Diffie-Hellman key exchanges
  - In addition to the 5 pre-defined groups
  - Protected by the IKE SA
  - Two message protocol: Propose and accept

Changes and Improvements in IKEv2

- Reduced complexity and some clarifications
- Support for NAT traversal (using UDP encapsulation with ESP SPI value of zero)
- Improvements in the DoS cookie mechanism
  - Limited source authentication to reduce likelihood of DoS attacks from spoofed source addresses
- SA lifetimes were negotiated in IKEv1, in IKEv2 the lifetimes can be chosen more or less arbitrarily by each party to the exchange
- IKEv1 and v2 are not interoperable, but are sufficiently different in header fields that they can run over the same port

Final Notes on IPSec

- IKE is carried over UDP (port 500); hence unreliable and blocked by some firewalls
- IPSec and firewalls have problems working together
  - Authentication of source IP addresses in AH is the issue
  - Some firewalls change these addresses on out-bound datagrams (masquerading or NAT)
- IPSec support for ICMP is somewhat complicated
- Managing IPSec policy and deployments is tricky
  - Getting it wrong can mean losing connectivity, e.g. by making exchanges of routing updates unreadable
  - Getting it wrong can mean loss of security
  - Many, many IPSec options, rather poor documentation

Final Notes on IPSec

- Microsoft started supporting IPSec with Windows XP, replacing PPTP; it is also part of most other Unix and Unixoid operating systems (usually also with IPv6 support)
- IPSec adopted in UMTS standards to provide secure communications for core network infrastructure
- Many vendor-specific hardware implementations
  - Typically integrated with firewall/router to provide general purpose security gateway
  - But IPSec VPN products are being severely challenged in the marketplace by SSL-based products

6.1 SSL/TLS

- SSL/TLS overview and basic features
- SSL Record Protocol
- SSL Handshake Protocol
- Other SSL Protocols
- SSL and TLS differences
- SSL applications

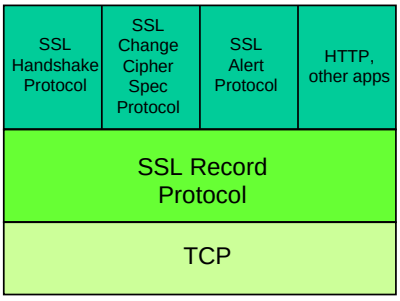
SSL/TLS Overview

- SSL = Secure Sockets Layer.
  - unreleased v1, flawed but useful v2, good v3.
- TLS = Transport Layer Security.
  - TLS1.0 = SSL3.0 with minor tweaks (see later).
  - Defined in RFC 2246.
  - Open-source implementation at http://www.openssl.org/.
- SSL/TLS provides security 'at TCP layer'.
  - Uses TCP to provide reliable, end-to-end transport.
  - Applications need some modification.
  - In fact, usually a thin layer between TCP and HTTP.

SSL/TLS Basic Features

- SSL/TLS widely used in Web browsers and servers to support 'secure e-commerce' over HTTP.
  - Built into Microsoft IE, Netscape, Mozilla, Apache, IIS,…
  - Use indicated by presence of browser lock.
- SSL architecture provides two layers:
  - SSL Record Protocol
    - Provides secure, reliable channel to upper layer.
  - Upper layer carrying:
    - SSL Handshake Protocol, Change Cipher Spec. Protocol, Alert Protocol, HTTP, any other application protocols.

SSL Protocol Architecture



SSL Record Protocol

- Provides secure, reliable channel to upper layer.
- Carries application data and SSL 'management' data.
- Session concept:
  - Sessions created by handshake protocol.
  - Session state defined by session ID and set of cryptographic parameters (encryption and hash algorithm, master secret, certificates) negotiated in handshake protocol.
  - Each session can carry multiple sequential *connections*.
- Connection concept:
  - Keys for multiple connections derived from master secret created during single run of handshake protocol.
  - New nonces used with master secret to derive keys for each new connection.
  - These nonces are exchanged in a lightweight version of handshake protocol.
  - Avoids repeated use of expensive handshake protocol

SSL Record Protocol

SSL Record Protocol provides:

- Data origin authentication and integrity.
  - MAC using algorithm similar to HMAC.
  - Based on MD-5 or SHA-1 hash algorithms.
  - MAC protects 64 bit sequence number for anti-replay.
- Confidentiality.
  - Bulk encryption using symmetric algorithm.
    - IDEA, RC2-40, DES-40 (exportable), DES, 3DES block ciphers.
    - RC4-40 and RC4-128 stream ciphers.

SSL Record Protocol

Operation of Record Protocol:

- Data from layer above is received and partitioned into fragments (max size 2^{14} bytes).
- Optional data compression.
  - Default option is no compression.
- Calculate MAC and append to data.
- Pad to multiple of encryption algorithm block length (if needed), then encrypt.
- Prepend header.
  - Containing content type, version, length of fragment.
- Submit to TCP.
- Reverse these steps at recipient.

SSL Handshake Protocol

- Like IPSec, SSL consumes symmetric keys:
  - MAC and encryption algorithms at Record Layer.
  - Initialization vectors (IVs) for encryption algorithms.
  - Different keys and IVs in each direction.
- These keys are established by the SSL Handshake Protocol and subsequent key derivation.
- As with IKE in IPSec, the SSL Handshake Protocol is a complex protocol with many options.

SSL Handshake Protocol Security Goals

- Entity authentication of participating parties.
  - Participants are called 'client' and 'server'.
    - Reflects typical usage in e-commerce.
  - Server nearly always authenticated, client more rarely.
  - Appropriate for most e-commerce applications.
- Establishment of a fresh, shared secret.
  - Shared secret used to derive further keys.
  - For confidentiality and authentication in SSL Record Protocol.
- Secure ciphersuite negotiation.
  - Encryption and hash algorithms
  - Authentication and key establishment methods.

SSL Handshake Protocol – Key Exchange

- SSL supports several key establishment mechanisms.
- Method used is negotiated during the Handshake Protocol itself.
- Most common is RSA encryption (as in Lecture 4).
  - Client chooses `pre_master_secret`, encrypts using public RSA key of server, sends to server.
- Can also create `pre_master_secret` from:
  - Fixed Diffie-Hellman
    - Server (and possibly Client) certificate contains DH parameters.
  - Ephemeral Diffie-Hellman
    - Server and Client choose fresh Diffie-Hellman components.
  - Anonymous Diffie-Hellman
    - Each side sends Diffie-Hellman values, but no authentication.
    - Vulnerable to man-in-middle attacks.

SSL Handshake Protocol – Entity Authentication

- SSL supports several different entity authentication mechanisms for clients and servers.
- Method used is again negotiated during the Handshake Protocol itself.
- Most common server authentication method is based on RSA.
  - Ability of server to decrypt `pre_master_secret` using its private key and then generate correct MAC in `finished` message using key derived from `pre_master_secret` authenticates server to client (c.f. Lecture 4).
- Less common: DSS or RSA signatures on nonces (and other fields, e.g. Diffie-Hellman values).

SSL Key Derivation

Keys used for MAC and encryption in Record Layer derived from `pre_master_secret`:

- Derive `master_secret` from `pre_master_secret` using combination of MD5 and SHA-1 hash functions.
- Derive `key_block` from `master_secret` and client/server nonces, by repeated use of MD5 and SHA-1 in combination.
- Split up `key_block` into MAC keys, encryption keys and IVs for use in Record Protocol as needed.

SSL Handshake Protocol Run

- An illustrative protocol run follows.
- We choose the most common use of SSL.
  - No client authentication.
  - Client sends `pre_master_secret` encrypted under Server's RSA public key
  - Server public key obtained from server certificate.
  - Server authenticated by ability to decrypt to obtain `pre_master_secret`, and construct correct `finished` message.
- Other protocol runs are similar.

SSL Handshake Protocol Run

M1: C → S: **`ClientHello`**

- Client initiates connection.
- Sends client version number.
  - 3.1 for TLS.
- Sends `ClientNonce` and `SessionID`.
  - Nonce is 28 random bytes plus 4 bytes of time.
  - `SessionID` used to signal request to set up new connection for existing session or to signal completely new session.
- Offers list of ciphersuites.
  - Key exchange and authentication options, encryption algorithms, hash functions.
  - E.g. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`.

SSL Handshake Protocol Run

M2: S → C: **`ServerHello`**, `ServerCertChain, ServerHelloDone`

- Sends server version number.
- Sends `ServerNonce` and `SessionID`.
  - `SessionID` will match client's if new connection for existing session; otherwise selected by server.
- Selects single ciphersuite from list offered by client.
  - E.g. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`.

SSL Handshake Protocol Run

M2: S → C: `ServerHello`, **`ServerCertChain, ServerHelloDone`**

- Sends `ServerCertChain` message.
  - Allows client to validate server's public key back to acceptable root of trust.
- (optional) `CertRequest` message.
  - Omitted in this protocol run – no client authentication.
- Finally, `ServerHelloDone`.

SSL Handshake Protocol Run

M3: C → S: **`ClientKeyExchange`**, `ChangeCipherSpec, ClientFinished`

- `ClientKeyExchange` contains encryption of `pre_master_secret` under server's RSA public key.
- (optional) `ClientCertificate`, `ClientCertificateVerify` messages.
  - Only sent when client is authenticated.
  - `ClientCertificateVerify` message is typically a signature on nonces (and other values) exchanged in the protocol run.
  - Authentication via signature and nonce (c.f. Lecture 4).

SSL Handshake Protocol Run

M3: C → S: `ClientKeyExchange`, **`ChangeCipherSpec`, `ClientFinished`**

- `ChangeCipherSpec` indicates that client is now switching to use of ciphersuite agreed for this session.
  - Sent using SSL Change Cipher Spec. Protocol.
  - Technically, an upper layer protocol.
- Finally, `ClientFinished` message.
  - A MAC on all messages sent so far (by both sides).
  - MAC computed using `master_secret`.
  - Provides protection of ciphersuite negotiation.

SSL Handshake Protocol Run

M4: S → C: `ChangeCipherSpec`, `ServerFinished`

- `ChangeCipherSpec` indicates that server is now switching to ciphersuite agreed for this session.
  - Sent using SSL Change Cipher Spec. Protocol.
- Finally, `ServerFinished` message.
  - A MAC on all messages sent so far (both sides).
  - MAC computed using `master_secret`.
  - Server can only compute MAC if it can decrypt `ClientKeyExchange` in M3 to get `pre_master_secret`.
  - Provides server authentication and protection of ciphersuite negotiation.

SSL Handshake Protocol Run

Summary:

M1: C → S: `ClientHello`

M2: S → C: `ServerHello`, `ServerCertChain`,`ServerHelloDone`

M3: C → S: `ClientKeyExchange`, `ChangeCipherSpec`, `ClientFinished`

M4: S → C: `ChangeCipherSpec`, `ServerFinished`

SSL Handshake Protocol Run

1. Is the client authenticated to the server in this protocol run?
2. Can an adversary learn the value of `pre_master_secret`?
3. Is the server authenticated to the client?

1. No!
2. No! Client has validated server's public key; only holder of private key can decrypt `ClientKeyExchange` to learn `pre_master_secret`.
3. Yes! `ServerFinished` includes MAC on nonces computed using key derived from `pre_master_secret`.

Other SSL Handshake Protocol Runs

- Many optional/situation-dependent protocol messages:
  - M2 (S→C) can include:
    - `ServerKeyExchange` (e.g. for DH key exchange), including server's signature.
    - `CertRequest` (for client authentication).
  - M3 (C→S) can include:
    - `ClientCert` (for client authentication),
    - `ClientCertVerify` (for client authentication).
- For details, see Stallings Figure 7.6 and pp. 212-219 (SSL) or RFC 2246 (TLS).

SSL Handshake Protocol – Additional Features

- SSL Handshake Protocol supports *session resumption* and *ciphersuite re-negotiation*.
  - Allows authentication and shared secrets to be reused across multiple connections in a single session.
    - Eg, fetching next web-page from same website without re-doing full, expensive Handshake Protocol
  - Also allows re-keying and change of ciphersuite during a session.

SSL Handshake Protocol – Additional Features

Mechanism:

- Client and server run lightweight version of Handshake Protocol.
- `ClientHello` quotes existing `SessionID`, new nonce and list of ciphersuites.
- `ServerHello` repeats `SessionID`, sends new nonce and selected ciphersuite.
- Parties then exchange `ChangeCipherSpec` and `Finished` messages.
- New `key_block` is derived by both sides.
  - New keys and IVs dependent on new nonces and old `master_secret`.
- Exchange protected by existing Record Protocol.

Other SSL Protocols

- Alert protocol.
  - Management of SSL session, error messages.
  - Fatal errors and warnings.
- Change cipher spec protocol.
  - Not part of SSL Handshake Protocol.
  - Used to indicate that entity is changing to recently agreed ciphersuite.
- Both protocols run over Record Protocol (so peers of Handshake Protocol).

SSL and TLS

TLS1.0 = SSL3.0 with minor differences, including:

- TLS signalled by version number 3.1.
- Use of HMAC for MAC algorithm.
- Different method for deriving keying material (`master_secret` and `key_block`).
  - Pseudo-random function based on HMAC with MD5 and SHA-1 operating in combination.
- Additional alert codes.
- More client certificate types.
- Variable length padding.
  - Can be used to hide lengths of short messages and so frustrate traffic analysis.

SSL/TLS Applications

Secure e-commerce using SSL/TLS.

- Client authentication not needed until client decides to buy something.
- SSL then provides a secure channel for transport of, for example, credit card details, security code, billing address.
- Hence user authentication at application level protected by SSL at transport level.
- Very successful (amazon.com, on-line supermarkets, airlines,…)

SSL/TLS Applications

Secure e-commerce: some issues.

- No guarantees about what happens to client data (including credit card details) after session: may be stored on insecure server.
- Does client understand meaning of certificate expiry and other security warnings?
- Does client software properly check server certificate chain?
- Can an attacker inject root certificates into the client browser?
- Does the name in certificate match the URL of the e-commerce site? Does the user check this?
- Is the site the one the client thinks it is?
- Is the client software proposing appropriate ciphersuites?

SSL/TLS Applications

Secure electronic banking.

- Client authentication may be enabled using client certificates.
  - Issues of registration, secure storage of private keys, revocation and re-issue.
- Otherwise, SSL provides secure channel for sending client credentials.
- Similar issues to e-commerce applications.

SSL/TLS Applications

Virtual Private Networking.

- SSL provides convenient method for enabling secure, remote access to web-facing applications.
- Popular due to widespread deployment of required browser software.
  - Compare to deployment issues for IPSec.
- Vendors producing web proxying components for non-web-facing applications, further extending applicability of SSL VPNs.
- SSL VPNs now a serious competitor to IPSec VPNs.

Some SSL/TLS Security Flaws

- (Historical) flaws in random number generation for SSL.
  - Low quality random number generator leads to predictable session keys.
  - Goldberg and Wagner, Dr. Dobb's Journal, Jan. 1996.
  - http://www.ddj.com/documents/s=965/ddj9601h/

Some SSL/TLS Security Flaws

- Flaws in error reporting.
  - (differing response times by server in event of padding failure and MAC failure) + (analysis of padding method for CBC-mode) = recovery of SSL plaintext.
  - Canvel, Hiltgen, Vaudenay and Vuagnoux, Crypto2003.
  - http://lasecwww.epfl.ch/php_code/publications/search.php?ref=CHVV03
- Timing attacks.
  - analysis of OpenSSL server response times allows attacker in same LAN segment to derive server's private key!
  - Boneh and Brumley, 12th Usenix Security Symposium, 2003.
  - http://crypto.stanford.edu/~dabo/abstracts/ssl-timing.html

6.2 SSH

- SSH overview
- SSH architecture
- SSH security
- Port forwarding with SSH
- SSH applications

SSH Overview

- SSH = Secure Shell.
  - Initially designed to replace insecure rsh, telnet utilities.
  - Secure remote administration (mostly of Unix systems).
  - Extended to support secure file transfer and other functions.
  - Latterly, provide a general secure channel for network applications.
  - SSH-1: flawed ad hoc design, now largely obsolete.
  - SSH-2: better security, more flexible architecture.
- SSH provides security at application layer.
  - Only covers traffic explicitly protected.
  - Applications need modification, but port-forwarding eases some of this (see later).
  - Built on top of TCP, reliable transport layer protocol.

SSH Overview

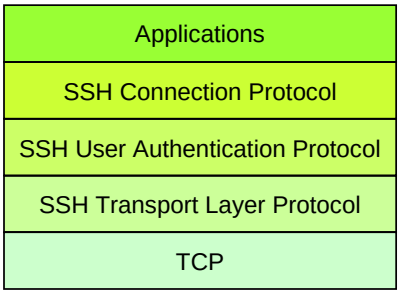
- SSH Communications Security (SCS).
  - www.ssh.com.
  - Founded by Tatu Ylonen, designer of SSH-1.
  - Their Tectia product suite implements SSH-2.
- Open source implementation of SSH-2 also available from OpenSSH.
- IETF Secure Shell (SECSH) working group.
  - Standards for SSH largely completed, long awaiting publication as RFCs.
  - www.ietf.org/html.charters/secsh-charter.html.

SSH-2 Architecture

SSH-2 adopts a three layer architecture:

- SSH Transport Layer Protocol.
  - Initial connection.
  - Server authentication (almost always).
  - Sets up secure channel between client and server.
- SSH User Authentication Protocol
  - Client authentication over secure transport layer channel.
- SSH Connection Protocol
  - Supports multiple concurrent connections over a single transport layer protocol secure channel.
  - Efficiency (session re-use) and support for multiple applications.
- Some texts consider UserAuth and Connection protocols to be peers. The IETF draft standards do not.

SSH-2 Architecture



SSH-2 Security Goals

- Server (almost) always authenticated in transport layer protocol.
  - Usually by a public key signature method.
  - Public keys supported by certificates and x.509 PKI / SPKI/ OpenPGP or manually distributed to clients.
- Client host/user usually authenticated in user authentication protocol.
  - By public key method (many methods supported).
  - Or simple password for particular application over secure channel.
  - Or via host-based method.

SSH-2 Security Goals

- Establishment of a fresh, shared secret.
  - Using Diffie-Hellman key exchange.
  - Shared secret used to derive further keys, similar to SSL/IPSec.
  - For confidentiality and authentication in SSH transport layer protocol.
- Secure ciphersuite negotiation.
  - Encryption, MAC, and compression algorithms.
  - Server authentication and key exchange methods.

SSH-2 Algorithms

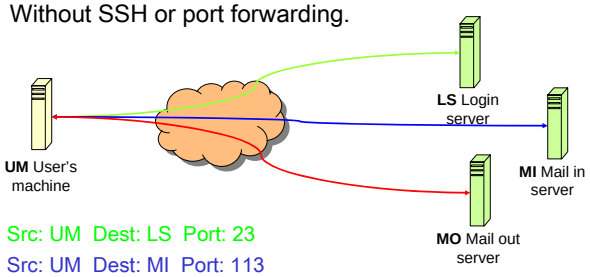
- SSH-2 requires support for particular algorithms, but also defines a DNS-style naming convention for “private” algorithms and methods.
- Typical algorithms:
  - Server authentication via RSA or DSS signatures on nonces (and other fields).
  - HMAC-SHA1 or HMAC-MD5 for MAC algorithm.
  - 3DES, AES, RC4 and many others.
  - SHA-1 hash function for key derivation.

SSH-1 Versus SSH-2

- Many vulnerabilities were found in SSH-1.
  - SSH-1 Insertion attack exploiting weak integrity mechanism (CRC-32) and unprotected packet length field.
  - SSHv1.5 session key retrieval attack (theoretical).
  - Man-in-the-middle attacks (using e.g. dsniff).
  - DoS attacks.
    - Overload server with connection requests.
    - Buffer overflows.
- SSH-1 now regarded as obsolete, but may still be widely deployed.
  - SSH-2 implementations tend to have an SSH-1 mode.
- Few SSH-2 protocol problems discovered, but plenty of vulnerabilities in implementations.

SSH Port Forwarding

Without SSH or port forwarding.



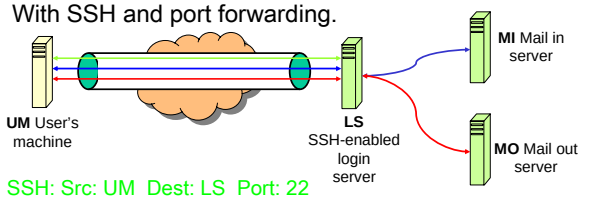
Src: UM Dest: LS Port: 23
Src: UM Dest: MI Port: 113
Src: UM Dest: MO Port: 25

SSH Port Forwarding

- Recall: TCP port number 'identifies' application.
- User on local machine:
  - Configures application to connect to selected destination port on local machine instead of normal port on remote machine.
  - Configures local SSH client to collect traffic on that port and forward it over secure SSH tunnel to remote SSH server.
- Remote SSH server:
  - Receives SSH-protected traffic and decrypts it.
  - Forwards traffic to appropriate server (based on port) using internal network.

SSH Port Forwarding

With SSH and port forwarding.



SSH: Src: UM Dest: LS Port: 22
Mail app: Src: UM Dest: UM Port: 5113
SSH client: Src: UM Dest: LS Port: 113
LS: Src: LS Dest: MI Port: 113
Mail app: Src: UM Dest: UM Port: 5025
SSH client: Src: UM Dest: LS Port: 25
LS: Src: LS Dest: MO Port: 25

SSH Applications

- Secure remote administration.
  - SysAdmin (client) sets up terminal on remote machine.
  - SysAdmin password protected by SSH transport layer protocol.
  - SysAdmin commands protected by SSH connection protocol.
- sftp:
  - Similar functionality to ftp but running over SSH.
- Guerilla Virtual Private Network.
  - E.g. use SSH + port forwarding to secure e-mail communications, web browsing, etc.
- Anonymous ftp for software updates, patches...
  - No client authentication needed, but clients want to be sure of origin and integrity of software.

6.3 Comparing IPSec, SSL/TLS, SSH

- All three have initial (authenticated) key establishment then key derivation.
  - IKE in IPSec
  - Handshake Protocol in SSL/TLS (can be unauthenticated!)
  - Authentication Protocol in SSH
- All protect ciphersuite negotiation.
- All three use keys established to build a 'secure channel'.

Comparing IPSec, SSL/TLS, SSH

- Operate at different network layers.
  - This brings pros and cons for each protocol suite.
  - Recall `Where shall we put security?' discussion.
  - Naturally support different application types, can all be used to build VPNs.
- All practical, but not simple.
  - Complexity leads to vulnerabilities.
  - Complexity makes configuration and management harder.
  - Complexity can create computational bottlenecks.
  - Complexity necessary to give both flexibility and security.

Comparing IPSec, SSL/TLS, SSH

Security of all three undermined by:

- Implementation weaknesses.
- Weak server platform security.
  - Worms, malicious code, rootkits,…
- Weak client platform security.
  - Keystroke loggers, malware,…
- Limited deployment of certificates and infrastructure to support them.
  - Especially client certificates.
- Lack of user awareness and education.
  - Users click-through certificate warnings.
  - Users fail to check URLs.
  - Users send sensitive account details to bogus websites ("phishing") in response to official-looking e-mail.

Secure Protocols – Last Words

A (mis)quote from Eugene Spafford:

"*Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.*"