

Security Tunneling

Cyber Security

Spring 2006

Reading Material

- IPsec and SSL are covered in Bishop's Computer Security Art and Science
 - RFC's ultimate source for both
- IPsec overview -
<http://www.alliancedatacom.com/IPsec-overview.asp>
- SSL/TLS overview -
http://httpd.apache.org/docs/2.0/ssl/ssl_intro.htm
!
- VLAN Security Paper –
 - http://www.cisco.com/warp/public/cc/pd/si/casi/ca6000/prodlit/vlnwp_wp.htm

What is a tunnel?

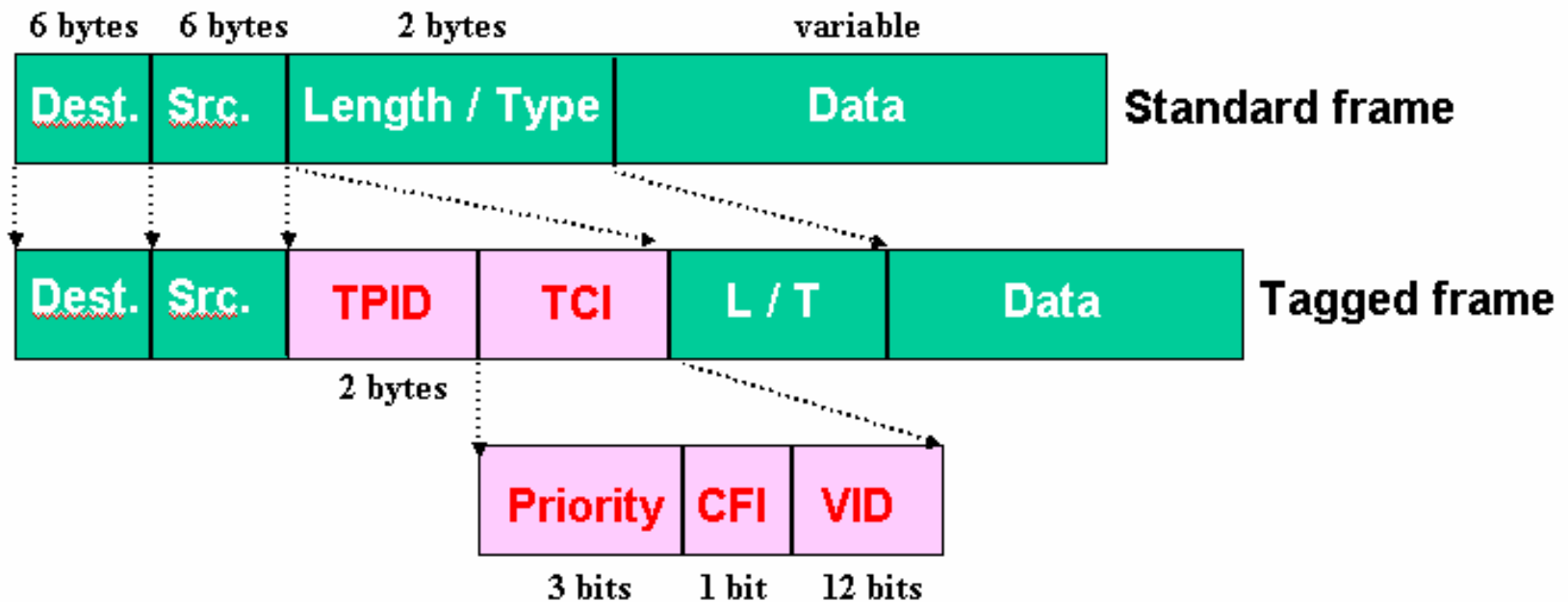
- A tunnel identifies packets in a data stream
 - Identify by encapsulation (new header possibly new trailer)
 - Identify by labeling.
- Entry into a tunnel gives the data stream different characteristics
 - E.g., Privacy, authentication, different routing characteristics
 - Security is not always the goal of the tunnel
- Also called virtual private networks (VPNs) in many situations

Tunnel Protocols for all Levels

- Layer 2
 - 802.1Q VLANs – labels ethernet frames for traffic separation
 - Proprietary link encryption
- Layer 3
 - IPSec
 - IPv6 in IPv4 – Carry IPv6 traffic over IPv4 networks
 - Generic Routing Encapsulation (GRE)
 - Multiprotocol Label Switching (MPLS) – uses labels to implement circuit switching at layer 3
- Layer 4
 - SSL/TLS
- Layer 7
 - SMIME
 - DNSSec

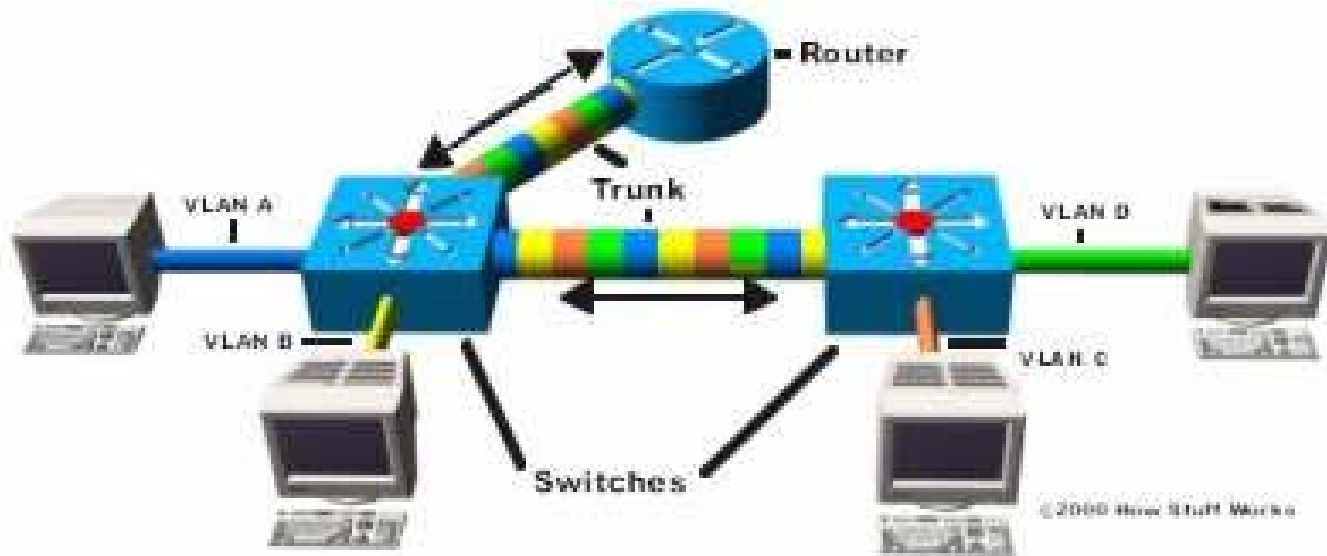
802.1Q VLAN

- Supported by many switches
- Augments ethernet frame with tag



VLAN Trunking

- Enables multiple VLANs to be carried over a single physical link between switches



VLAN used in Siebel

- Using VLANs in the lab configuration to create virtual wires between firewalls, hosts, and the outside world
- CS Department uses VLAN trunking to virtually connect machines
- VLAN trunking will provide lab access to a virtual devices running on a VMWare server in a far distant machine room.

VLAN Security Issues

- Classic case of security being an after thought
 - Designed for traffic separation, not security!
- Cisco white paper on VLAN security
 - http://www.cisco.com/en/US/products/hw/switches/ps708/products_white_paper09186a008013159f.shtml

VLAN 1

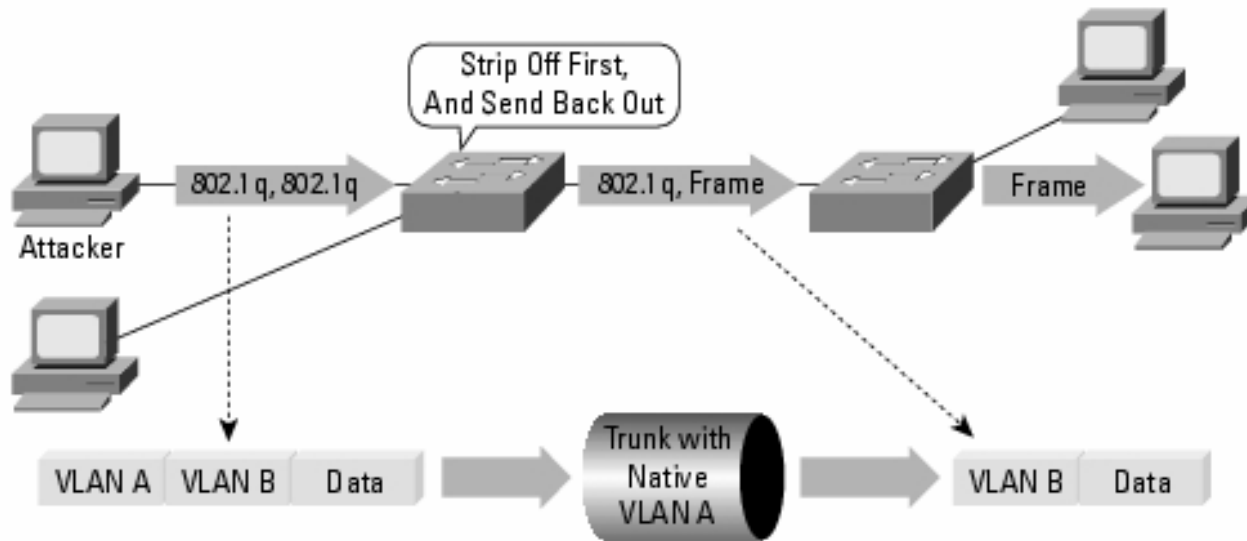
- By default unmarked packets go into VLAN 1
 - Means VLAN 1 tends to appear on multiple switches
 - Bad activity on VLAN 1 will affect the entire network
- Understand where VLAN 1 is used and prune back unnecessary uses

Differentiate Trusted and Untrusted Ports

- Reduce protocols on untrusted ports
 - Limit points of attack
- For example, VLAN Trunking Protocol (VTP) or Dynamic Trunking Protocol (DTP)
 - Cisco proprietary protocol that allows for automatic propagation of VLAN configuration across the network
 - If VTP could be co-opted by bad guy can reconfigure the network.

Native VLANs

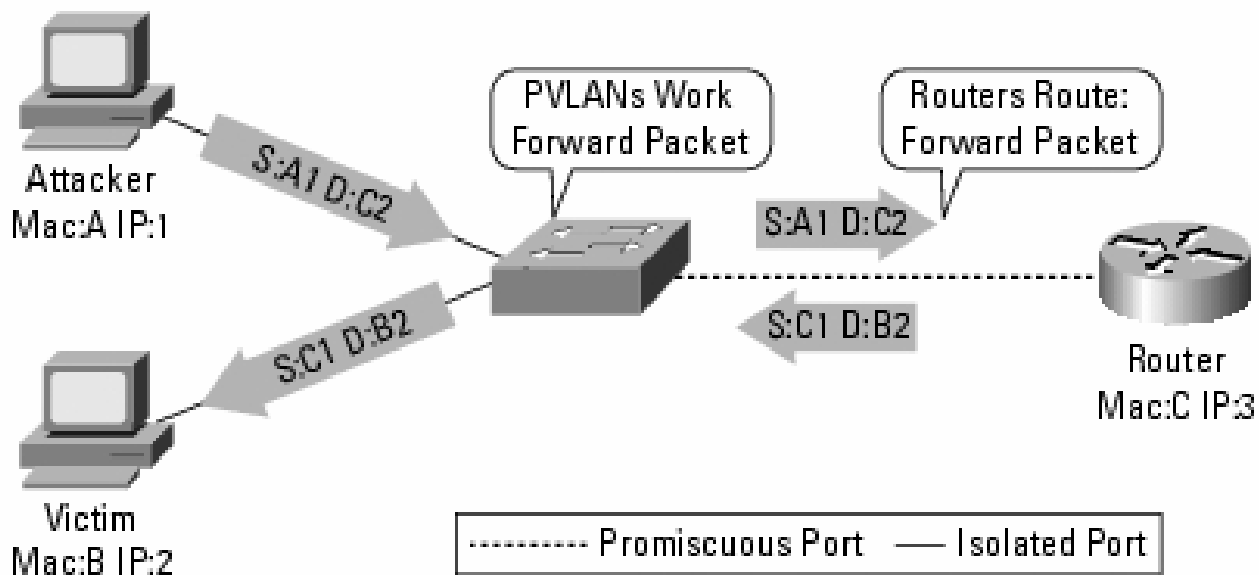
- Created for backwards compatibility
 - One of the VLANs associated with port can be native
 - All untagged packets to with the native VLAN
 - All tagged packets in native VLAN get stripped



Note: Only Works if Trunk Has the Same Native VLAN as the Attacker.

Private VLAN Attack

- Private VLAN
 - An escape to let routed traffic pass between L2 constraints
 - L2 Proxy



Other Layer 2 Attacks

- MAC Flooding
- ARP Spoofing
- 802.1Q tagging attack
 - Attacker creates DTP packets. Trick port into going into trunk mode.
- Spanning Tree Protocol (STP) Attacks
 - Broadcast protocol to agree on a tree of bridges to avoid broadcast loops
 - Attacker attempts to insert packets claiming he is new root bridge

IPSec Operational Architecture

- IPSec Security Architecture, RFC 2401
- Designed by the Security Working Group of the IETF.
 - <http://ietf.org/html.charters/ipsec-charter.html>
- Motivated from IPv6 design
 - Add arbitrary number of extension headers to store information about the security protocols
 - First IPv4 implementations around '97

Security Association (SA)

- Records on the endpoints that store operational information
 - E.g., encryption protocol, keying information, traffic stream filters
- One SA per endpoint to represent a simplex connection
 - Two pairs of SAs to represent duplex connectivity
- The SA memory footprint can be a limiting factor in the number of tunnels
 - Smaller routers cannot support very many simultaneous SAs
- Must know the ID of your peer's SA to communicate
 - Addressed by the Security Parameters Index (SPI)
 - SPI identified in the security protocol headers
 - SPI + Peer address + security protocol will uniquely identify a SA

IPSec Protocols

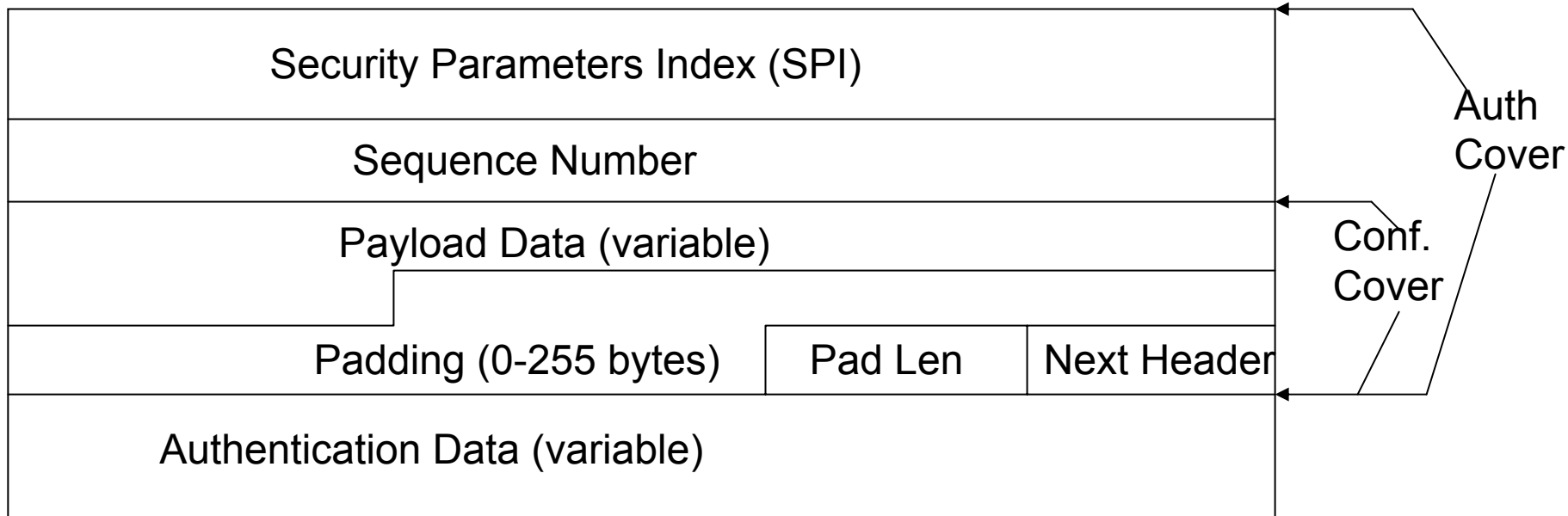
- The IPSec framework describes how a number of different IPSec security protocols can be applied to a tunnel
- Two protocols implemented
 - Encapsulating Security Payload (ESP) – provides privacy (encryption) and message authentication (detection of change)
 - Authentication Header (AH) – provides authentication (detection of change)

ESP

- RFC 2406
- Initially ESP only provided confidentiality not message authentication
 - You were supposed to use AH get authentication
 - People argued that ESP as not useful without authentication, so it was added in as an option
 - Now AH is not so valuable, since you can use a null encryption in ESP to get essentially the same thing

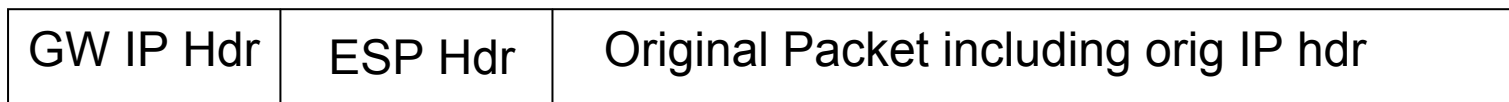
ESP Header

- Both confidentiality and message authentication cover part of the header
- Payload data can include things like initialization vectors (IVs)
- Sequence number is used to avoid replay attacks

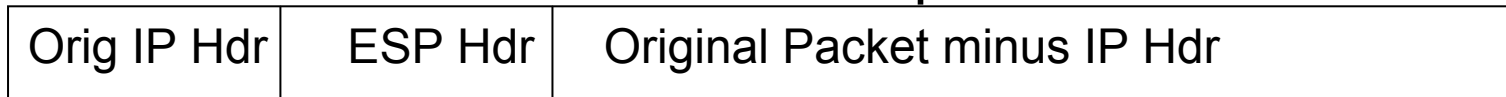


Tunnel and Transport Modes

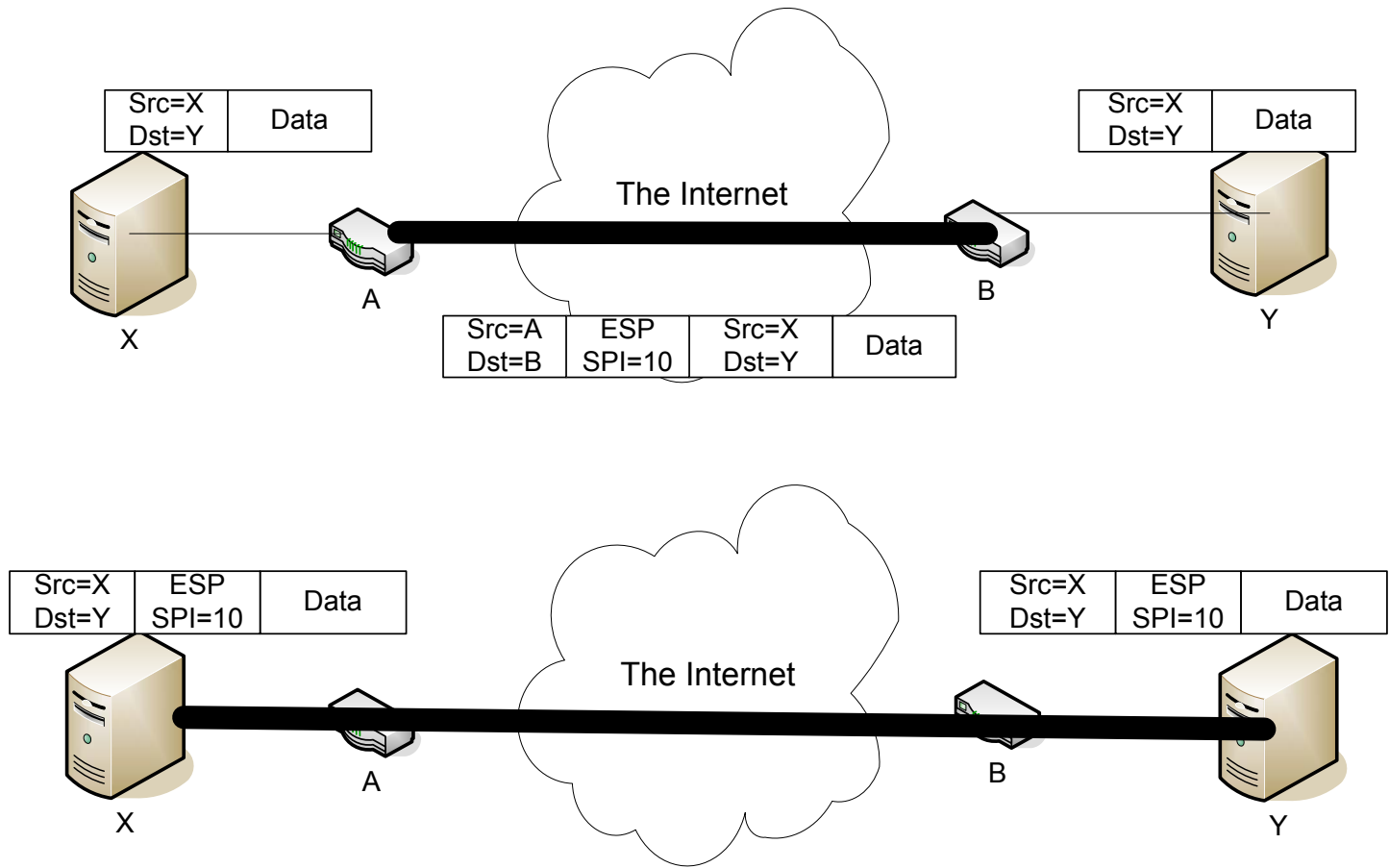
- IPSec tunnels can be set up in two modes
- Tunnel mode
 - Creates a new IP header and encapsulates the original
 - Used by gateways



- Transport mode
 - Just encapsulates the transport layer and beyond
 - Can be used if the source and destination of the traffic are also the tunnel endpoints



Tunnel and Transport Modes



IPSec Startup Negotiations

- To start a tunnel need to have the endpoints agree on certain things
 - Keying material
 - Protocols to use on which types of traffic
 - E.g., use ESP with 3DES on HTTP traffic
 - Belief that the peer is who he says he is
- Some of this can be hard coded in the endpoint configuration
 - All of it was initially using manual keying
- Now much of it can be negotiated with Internet Key Exchange (IKE) Protocol

Internet Key Exchange (IKE)

- RFC 2409
- Uses the ISAKMP SA framework (RFC 2408) and the Oakley key negotiation protocol (RFC 2412)
- Performs mutual endpoint authentication
 - Shared key authentication
 - Certificate authentication
 - Plus a few other
- Protocol (transform) agreement
- Key exchange and re-keying

Security Boot Stapping

- Endpoints must share some sort of information to start communicating
 - Shared secret with peer
 - Knowledge of the peer's certificate
- Extended authentication mode (Xauth) allows the human input of authentication data that is validated against a Radius server
- If you weren't using IKE, you could use a manual key for all communication

Oakley Key Determination

- Optionally uses Diffie-Hellman to provide perfect forward secrecy
 - If private key is compromised, previously captured data is not at risk
 - Must re-key often and cannot simply exchange data keys by encrypting with the main private key

Diffie-Hellman Key Exchange

- By examining the communication of a pair of parties you cannot derive their agreed upon scheme unless you can compute a discrete log
- Alice and Bob agree on large prime n and a g that is primitive mod n (g is often 2)
 - Good n 's and g 's are published. Oakley defines 6 well-known groups.
- Alice chooses a large random number x and computes $X = g^x \text{ mod } n$ and sends X to Bob
- Bob chooses a large random number y , computes $Y = g^y \text{ mod } n$ and sends Y to Alice
- Alice computes $k = Y^x \text{ mod } n$
- Bob computes $k' = X^y \text{ mod } n$
- $k = k'$ so they now have a shared key
- Diffie-Hellman exchanges can be quite computationally expensive

ISAKMP

- ISAKMP negotiations take place over a fixed SA
- Divided into two phases
 - First phase negotiates the security of communication over the ISKMP SA itself
 - Second phase negotiates security attributes of the target SA
- The results of the first phase can be used over multiple second phase negotiations

ISAKMP anti-clogging

- Uses cookies for simple denial of service avoidance
 - Goal is to prevent simple IP spoofing from causing endpoint to perform many computationally intensive calculations
 - Each end selects a value that includes information about the endpoint addresses and ports, time, and a secret value
 - Each end can determine if it's cookie is stale to avoid simple DOS
 - Still need to aggressively cleanup requests that end up being bogus

Transform Negotiation

- The initiator provides a list of security protocols and transforms it is willing to use on the negotiated SA. The proposals are ordered by preference
- The responder selects from one or rejects the negotiation if none of the proposals are match that peer's capabilities or policy requirements

Main Mode and Aggressive Mode

- ISAKMP can be run in two modes
- Main mode uses more message exchanges
 - Exchanges minimal information each round trip
 - Enables protection of sensitive data like identity protection
- Aggressive mode reduces number of messages exchanged
 - At the cost of not being able to protect as much data during the exchanges

Main Mode Example

- ->Initiator: Hdr; SA; nonce
- <-Responder: Hdr; SA; nonce. Now peers agree on a SA. The SA negotiation involves agreeing on a set of protocols, e.g. ESP with 3DES vs ESP with DES
- ->Initiator: Hdr; KE
- Responder generates a key. Sends back Hdr; KE
- Initiator uses the KE information to generate its copy of the shared secret. Now can communicate using this new key
- ->Initiator: Hdr; IDii; Auth
- Responder verifies initiators identity. Returns Hdr; IDir; Auth
- Initiator verifies responder's identity. Exchange complete in 3 round trips

Aggressive Mode Example

- ->Initiator: Hdr; SA; KE; Nonce; IDii
- <-Responder: Hdr; SA; KE; Nonce; IDir; Auth
- ->Initiator: Hdr*;Auth First protected traffic

NAT Transparent IPSec

- Initially IPSec could not handle address translation in the middle
 - RFC 3715 describes the problems
 - AH includes the addresses in the outer IP header in its authentication calculation
 - Changes to the IP addresses affect the TCP/UDP checksums, which are encrypted in ESP
 - Addresses and ports encrypted or authenticated
 - For remote users this was a big use case
- Introduced NAT-traversal extensions RFC 3947
- Detect NAT during IKE
 - Move from standard IKE port on 500 to negotiate on port 4500
 - Encapsulate the IPSec traffic using UDP to preserve the original headers from NAT
- One endpoint must fixup the translated addresses after untunneling the traffic

IPSec challenges

- **Scaling**
 - Numerous security associations eat up too much memory for small routers
 - Configurations on the hub in a hub and spoke network grow n^2 in the number of spokes
 - Dynamic Multipoint VPN (DMVPN)
- **Performance**
 - Even symmetric encryption can be too much for high bandwidth environments
- **Symmetry**
 - Both sides must have a means to prove identity to each other
 - Implies the need for a PKI or other broad identity proof mechanism

SSL

- Transport layer security
 - Provides confidentiality, integrity, authentication of endpoints
 - Developed by Netscape for WWW browsers and servers
- Internet protocol version: TLS
 - Compatible with SSL
 - Standard [rfc2712](#)

Working at Transport Level

- Data link, Network, and Transport headers sent unchanged
- Original transport header can be protected if tunneling

Ethernet Frame Header	IP Header	TCP Header	TCP data stream Encrypted/authenticated Regardless of application
-----------------------------	--------------	---------------	-------------------------------------------------------------------------

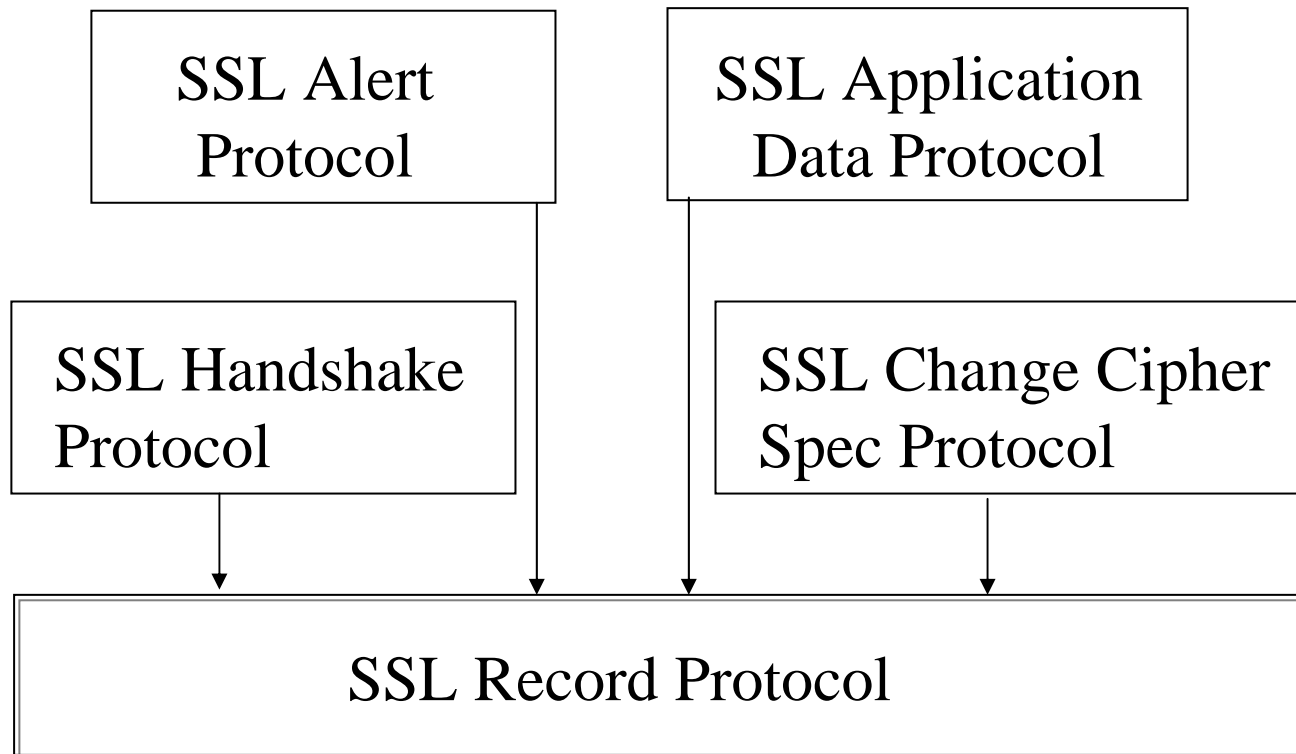
SSL Session

- Association between two peers
 - May have many associated connections
 - Information for each association:
 - Unique session identifier
 - Peer's X.509v3 certificate, if needed
 - Compression method
 - Cipher spec for cipher and MAC
 - "Master secret" shared with peer
 - 48 bits

SSL Connection

- Describes how data exchanged with peer
- Information for each connection
 - Random data
 - Write keys (used to encipher data)
 - Write MAC key (used to compute MAC)
 - Initialization vectors for ciphers, if needed
 - Sequence numbers

Structure of SSL



Supporting Crypto

- All parts of SSL use them
- Initial phase: public key system exchanges keys
 - Classical ciphers ensure confidentiality, cryptographic checksums added for integrity
 - Only certain combinations allowed
 - Depends on algorithm for interchange cipher
 - Interchange algorithms: RSA, Diffie-Hellman, Fortezza
 - AES added in 2002 by [rfc3268](#)

RSA: Cipher, MAC Algorithms

<i>Interchange</i>	<i>Classical cipher</i>	<i>MAC Algorithm</i>
RSA, <i>cipher</i> key \leq 512 bits	<i>none</i>	MD5, SHA
	RC4, 40-bit key	MD5
	RC2, 40-bit key, CBC	MD5
	DES, 40-bit key, CBC	SHA
RSA	<i>none</i>	MD5, SHA
	RC4, 128-bit key	MD5, SHA
	IDEA, CBC mode	SHA
	DES, CBC mode	SHA
	DES, EDE mode, CBC	SHA

mode

Diffie-Hellman: Types

- Diffie-Hellman: certificate contains D-H parameters, signed by a CA
 - DSS or RSA algorithms used to sign
- Ephemeral Diffie-Hellman: DSS or RSA certificate used to sign D-H parameters
 - Parameters not reused, so not in certificate
- Anonymous Diffie-Hellman: D-H with neither party authenticated
 - Use is “strongly discouraged” as it is vulnerable to attacks

D-H: Cipher, MAC Algorithms

<i>Interchange cipher</i>	<i>Classical cipher</i>	<i>MAC</i>
Diffie-Hellman, DSS Certificate	DES, 40-bit key, CBC mode	SHA <i>Algorithm</i>
	DES, CBC mode	SHA
	DES, EDE mode, CBC	SHA
Diffie-Hellman, key ≤ 512 bits RSA Certificate	DES, 40-bit key, CBC mode	SHA
	DES, CBC mode	SHA
	DES, EDE mode, CBC mode	SHA

Ephemeral D-H: Cipher, MAC Algorithms

<i>Interchange cipher</i>	<i>Classical cipher</i>	<i>MAC Algorithm</i>
Ephemeral Diffie-Hellman, DSS Certificate	DES, 40-bit key, CBC mode	SHA
	DES, CBC mode	SHA
	DES, EDE mode, CBC mode	SHA
Ephemeral Diffie-Hellman, key ≤ 512 bits, RSA Certificate	DES, 40-bit key, CBC mode	SHA
	DES, CBC mode	SHA
	DES, EDE mode, CBC mode	SHA

Anonymous D-H: Cipher, MAC Algorithms

<i>Interchange</i>	<i>Classical cipher</i>	<i>MAC Algorithm</i>
Anonymous D-H, DSS Certificate	RC4, 40-bit key	MD5
	RC4, 128-bit key	MD5
	DES, 40-bit key, CBC	SHA
	DES, CBC mode	SHA
	DES, EDE mode, CBC mode	SHA

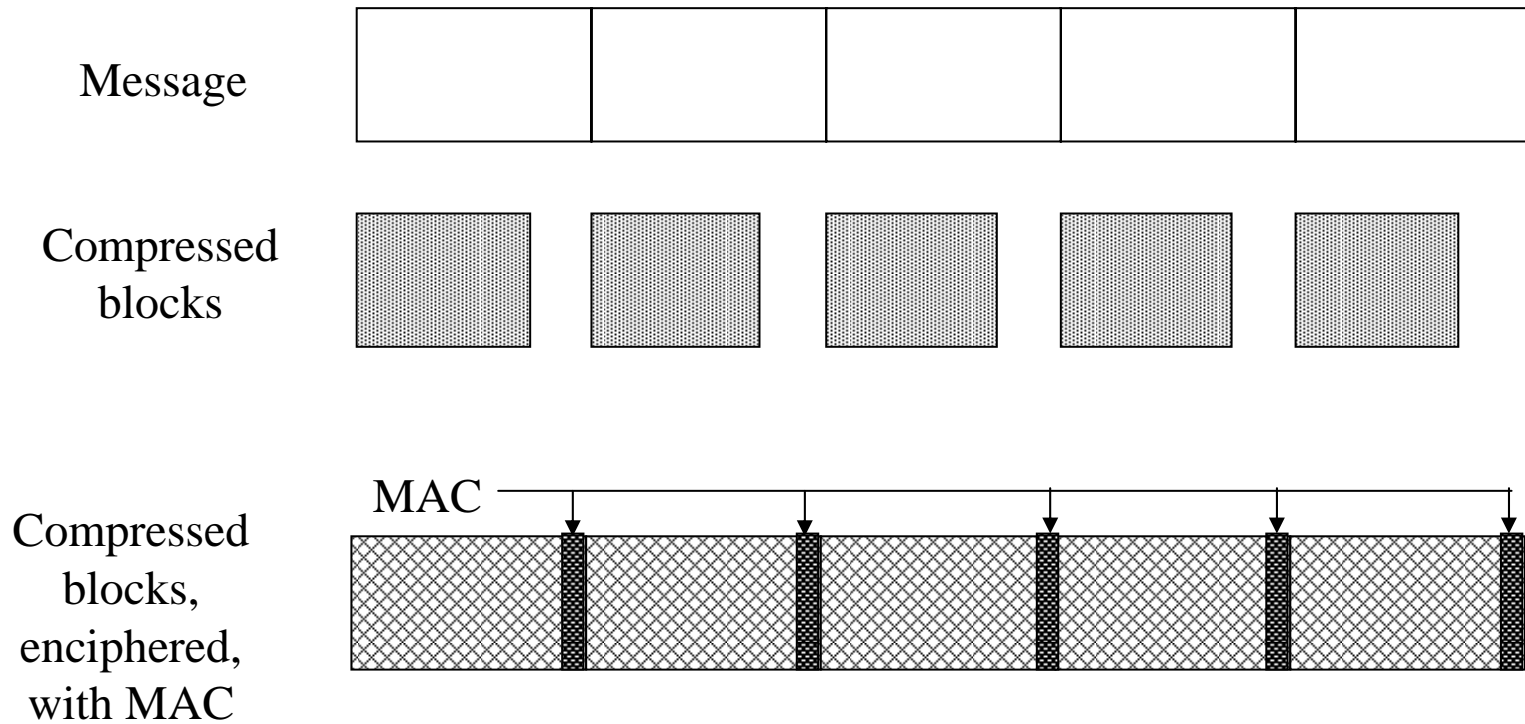
Fortezza: Cipher, MAC Algorithms

<i>Interchange</i>	<i>Classical cipher</i>	<i>MAC Algorithm</i>
Fortezza ^{<i>cipher</i>} key exchange	<i>none</i>	SHA
	RC4, 128-bit key	MD5
	Fortezza, CBC mode	SHA

Digital Signatures

- RSA
 - Concatenate MD5 and SHA hashes
 - Sign with public key
- Diffie-Hellman, Fortezza
 - Compute SHA hash
 - Sign appropriately

SSL Record Layer



Record Protocol Overview

- Lowest layer, taking messages from higher
 - Max block size 16,384 bytes
 - Bigger messages split into multiple blocks
- Construction
 - Block b compressed; call it b_c
 - MAC computed for b_c
 - If MAC key not selected, no MAC computed
 - b_c , MAC enciphered
 - If enciphering key not selected, no enciphering done
 - SSL record header prepended

SSL MAC Computation

- Symbols

- h hash function (MD5 or SHA)
- k_w write MAC key of entity
- $ipad = 0x36$, $opad = 0x5C$
 - Repeated to block length (from HMAC)
- seq sequence number
- SSL_comp message type
- SSL_len block length

- MAC

$h(k_w || opad || h(k_w || ipad || seq || SSL_comp || SSL_len || block))$

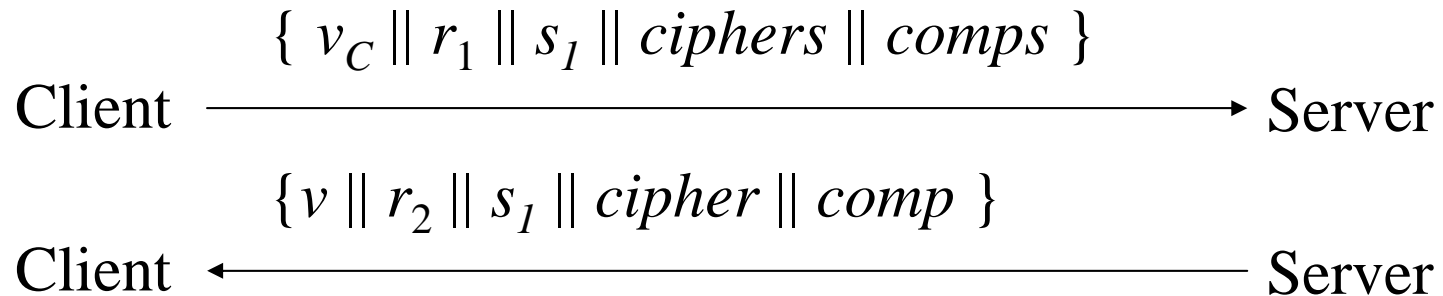
SSL Handshake Protocol

- Used to initiate connection
 - Sets up parameters for record protocol
 - 4 rounds
- Upper layer protocol
 - Invokes Record Protocol
- Note: what follows assumes client, server using RSA as interchange cryptosystem

Overview of Rounds

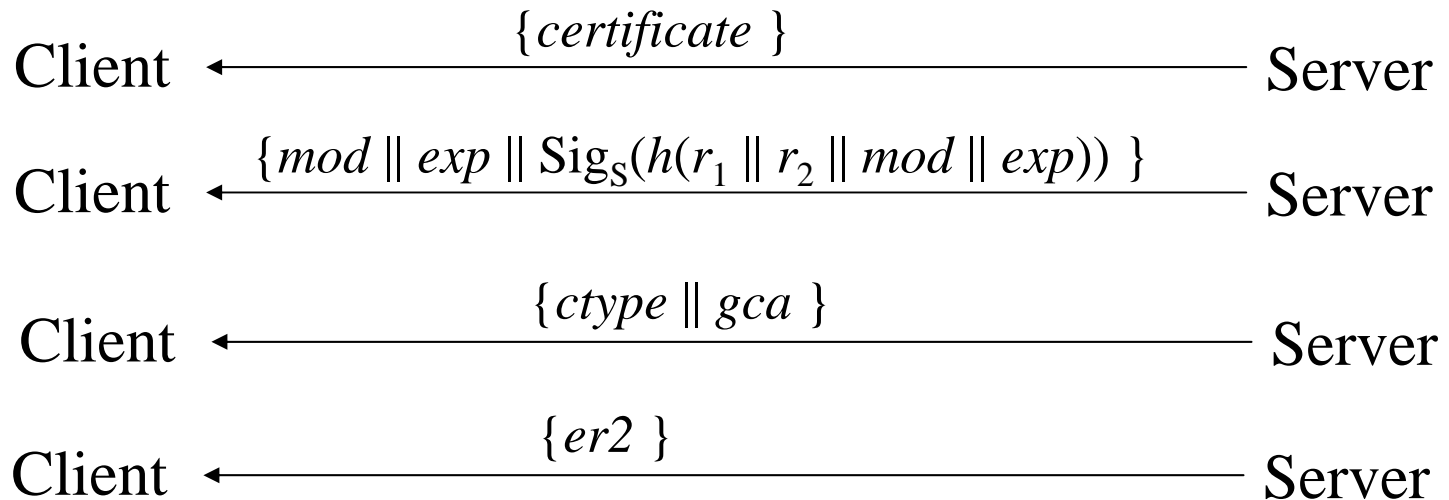
1. Create SSL connection between client, server
2. Server authenticates itself
3. Client validates server, begins key exchange
4. Acknowledgments all around

Handshake Round 1



v_C	Client's version of SSL
v	Highest version of SSL that Client, Server both understand
r_1, r_2	nonces (timestamp and 28 random bytes)
s_1	Current session id (0 if new session)
$ciphers$	Ciphers that client understands
$comps$	Compression algorithms that client understand
$cipher$	Cipher to be used
$comp$	Compression algorithm to be used

Handshake Round 2



Note: if Server not to authenticate itself, only last message sent; third step omitted if Server does not need Client certificate

k_S Server's private key

\textit{ctype} Certificate type requested (by cryptosystem)

\textit{gca} Acceptable certification authorities

$\textit{er2}$ End round 2 message

Handshake Round 3

Client $\xrightarrow{\{ client_cert \}}$ Server

Client $\xrightarrow{\{ pre \} Pub_S}$ Server

Both Client, Server compute master secret *master*:

$$\begin{aligned} master = & MD5(pre \parallel SHA('A' \parallel pre \parallel r_1 \parallel r_2) \parallel \\ & MD5(pre \parallel SHA('BB' \parallel pre \parallel r_1 \parallel r_2) \parallel \\ & MD5(pre \parallel SHA('CCC' \parallel pre \parallel r_1 \parallel r_2)) \end{aligned}$$

Client $\xrightarrow{\{ h(master \parallel opad \parallel h(msgs \parallel master \parallel ipad)) \}}$ Server

msgs Concatenation of previous messages sent/received this handshake

opad, ipad As above

Handshake Round 4

Client sends “change cipher spec” message using that protocol

Client \longrightarrow Server

$\{ h(\textit{master} // \textit{opad} // h(\textit{msgs} // 0x434C4E54 // \textit{master} // \textit{ipad})) \}$

Client \longrightarrow Server

Server sends “change cipher spec” message using that protocol

Client \longleftarrow Server

$\{ h(\textit{master} // \textit{opad} // h(\textit{msgs} // \textit{master} // \textit{ipad})) \}$

Client \longleftarrow Server

msgs Concatenation of messages sent/received this handshake in
previous rounds (does not include these messages)

opad, ipad, master As above

SSL Change Cipher Spec Protocol

- Send single byte
- In handshake, new parameters considered “pending” until this byte received
 - Old parameters in use, so cannot just switch to new ones

SSL Alert Protocol

- Closure alert
 - Sender will send no more messages
 - Pending data delivered; new messages ignored
- Error alerts
 - Warning: connection remains open
 - Fatal error: connection torn down as soon as sent or received

SSL Alert Protocol Errors

- Always fatal errors:
 - unexpected_message, bad_record_mac, decompression_failure, handshake_failure, illegal_parameter
- May be warnings or fatal errors:
 - no_certificate, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown

SSL Application Data Protocol

- Passes data from application to SSL Record Protocol layer