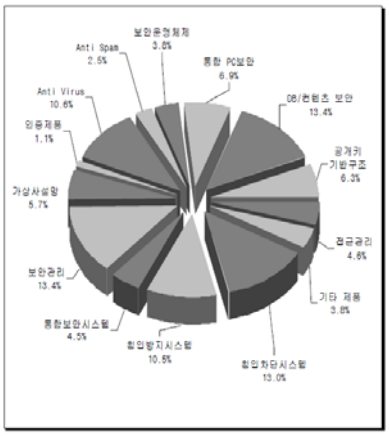
국내 지식정보보안산업 현황

(그림 4-3) 시스템 및 네트워크 정보보안 제품 매출액 (단위 : 백만원)



(그림 4-4) 시스템 및 네트워크 정보보안 제품 매출 비중

출처: 2009 국내지식정보보안 산업 시장 및 동향조사, KISA, 2009.12

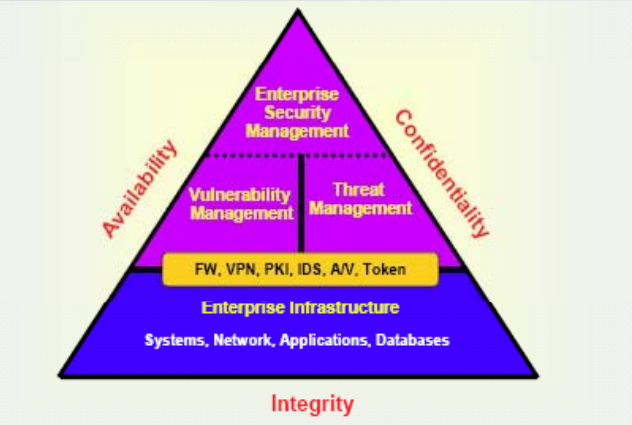
지식정보보안 산업의 정의 "정보보안산업과 물리보안산업을 결합한 산업군으로 암호, 인증. 감시 등의 보안 기술이 적용된 제품군을 생산하거나 관련 보안 기술을 활용하여 재난,재해,범죄 등을 방지하는 서비스를 제공하는 산업"(지식경제부, 2008.12.15)

❑1

VPN

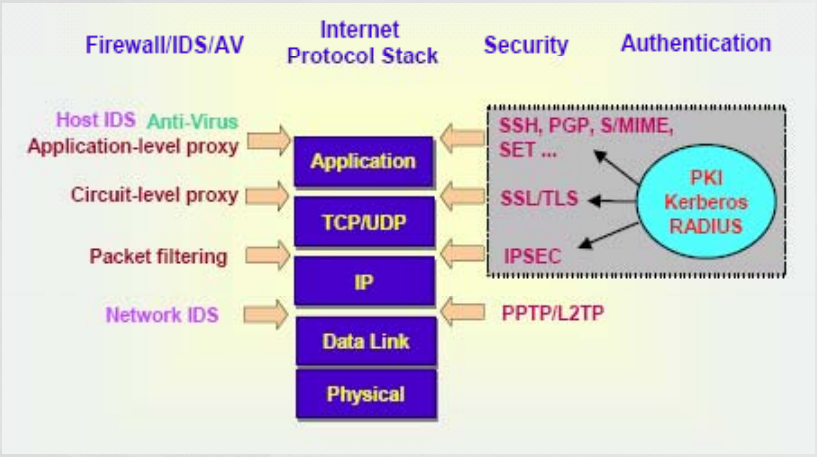
❑2

Enterprise Security Management



❑3

Major Internet Security Technologies



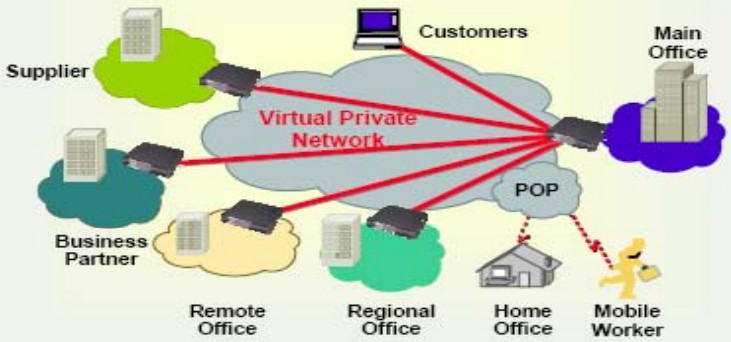
❑4

Cryptographic Primitives

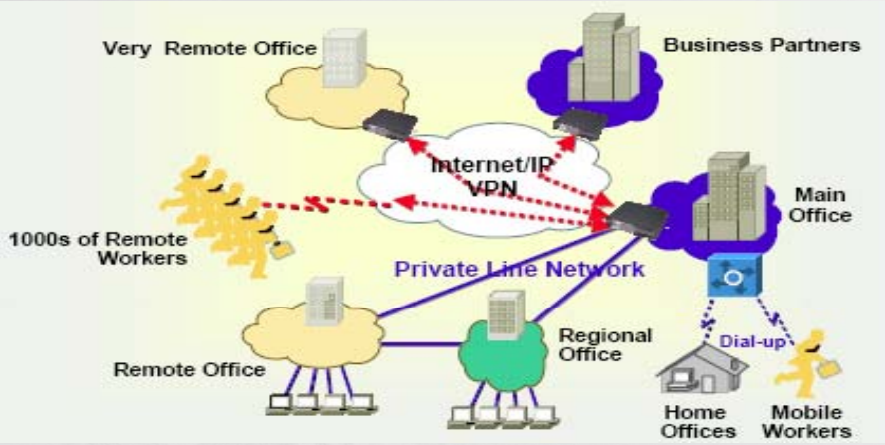
- Block / Stream Cipher (Symmetric Cryptosystem)
  - 3DES, AES, SEED, RC5 … / RC4, SEAL …
- Hash Function
  - MD5, SHA1/SHA2, HAS160, RMD160, Tiger …
- Message Authentication Code (MAC)
  - HMAC, CBC-MAC, UMAC
- Public Key (Asymmetric) Cryptosystem
  - RSA, ElGamal; Hybrid systems (Asymmetric key agreement + symmetric encryption)
- Digital Signature
  - RSA, DSA/ECDSA, KCDSA/EC-KCDSA …
- Key Exchange
  - Diffie-Hellman, ECDH, RSA key transport
- (Pseudo) Random Number Generators
  - HW pure RNG; SW pseudo RNG

What is VPN ?

Secure connectivity deployed on a shared infrastructure
with the same security policies and performance as a private network



VPN extends classic WAN



❑7

VPN Business Applications

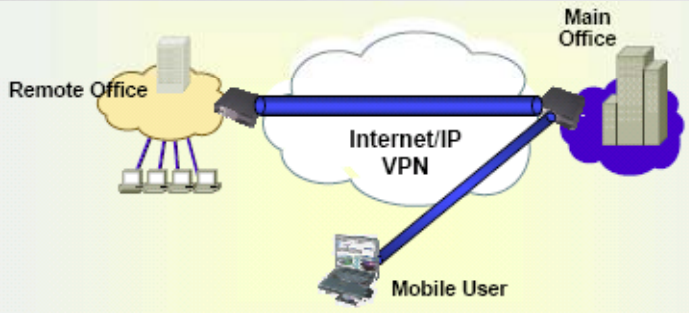
- ❑ Build secure business infrastructure
  - ➢ Integrate dispersed business environments using secure, controlled connectivity over shared networks
  - ➢ Implement once for multiple applications
  - ➢ Centrally-controlled access policy
  - ➢ Enable multi-level, layered approach to security
- ❑ Use internet for remote access
  - ➢ Mobile users use internet accounts to gain access and tunnel to offices
- ❑ Create internal security
  - ➢ Protect sensitive internal traffic/systems from others
- ❑ Can also make private networks more private
- ❑ Can be used to back-up existing private networks
- ❑ VPN issues
  - ➢ Security
  - ➢ Quality of Service
  - ➢ Scalability/Reliability
  - ➢ Manageability

❑8

VPN Key Components

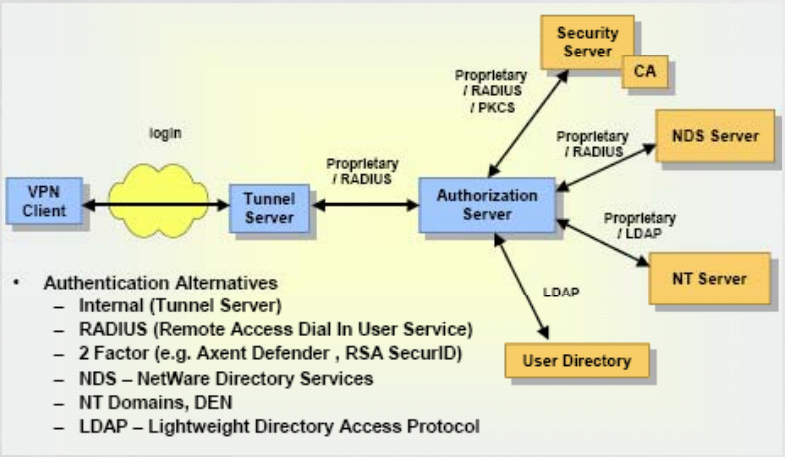
- Tunneling
  - PPTP, L2TP; MPLS; IPSEC, GRE, IP-in-IP; SSL/TLS
- Security
  - IPSEC vs Virtual path(VC, PVC, LSP, etc.)
  - Encrypted tunnel vs traffic separation
- Access control
  - Remote user authentication
  - Membership management
- Policy Management
  - Centralized policy control
  - Policy configuration, distribution & update
- Quality of Service(QoS)
  - Traffic classification, marking, policing & shaping
  - SLA: Latency, throughput, jitter, packet loss…
- High Availability
  - Transparent session fail-over
  - Load balancing, IP clustering

Secure Tunneling: Virtual Presence



- Tunneling makes geographically dispersed offices/mobiles appear to be present in the same local network.
- Security makes VPN traffic separated from other traffic.

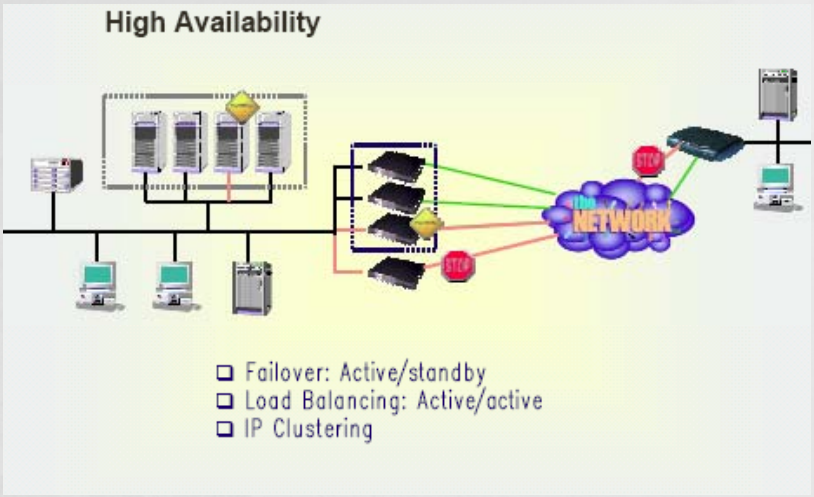
User Authentication Model



- Authentication Alternatives
  - Internal (Tunnel Server)
  - RADIUS (Remote Access Dial In User Service)
  - 2 Factor (e.g. Axent Defender , RSA SecurID)
  - NDS – NetWare Directory Services
  - NT Domains, DEN
  - LDAP – Lightweight Directory Access Protocol

❑11

High Availability



- ❑ Failover: Active/standby
- ❑ Load Balancing: Active/active
- ❑ IP Clustering

❑12

Internet VPN standards

VPN : Tunneling + Separation Technology

Protocol	PPTP	L2TP	MPLS	IPSEC	SOCKS5
Standard	MS	RFC	RFC	RFC	RFC
Layer	Link(2)	Link(2)	Layer 2.5	Network(3)	Session(5)
Security	PPP	Recommend IPSEC	None / IPSEC	IPSEC	SSL
WAN	IP only	Multi-protocol	Multi-protocol	IP only	IP only
Best for	Remote access	Remote access	Network-based VPN	Intra/extra net	Extranet

- PPTP/L2TP: Layer 2 tunneling by encapsulating PPP
- IPSEC : a set of IP layer end-to-end security protocols (AH, ESP, ISAKMP/IKE)
- MPLS : Multi-Protocol Label Switching
- SOCKS V5 : Session (circuit) level proxy with security features

❑13

Firewall

❑14

Background

- Evolution of information systems
- Now everyone want to be on the Internet and to interconnect networks
- Persistent security concerns remain

- Typically use a **Firewall** to provide **perimeter defence** as part of comprehensive security strategy

What's Firewall ?

- **Choke point** of control and monitoring
- Interconnects networks with differing trust
- Imposes restrictions on network services
  - only authorized traffic is allowed
- Auditing and controlling access
  - can implement alarms for abnormal behavior
- Provide NAT & usage monitoring
- Implement VPNs using IPSec
  - must be immune to penetration

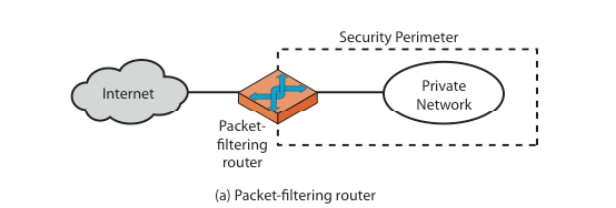
Firewall Limitations

- Can't protect from attacks bypassing it
  - *e.g.* sneaker net, utility modems, trusted organisations, trusted services (*e.g.* SSL/SSH)
- Can't protect against internal threats
  - *e.g.* disgruntled or colluding employees
- Can't protect against transfer of all virus infected programs or files
  - because of huge range of O/S & file types

Firewalls – Packet Filters(1/2)

- Simplest, fastest firewall component
- Foundation of any firewall system
- Examine each IP packet (no context) and permit or deny according to rules
- Restrict access to services (ports)
- Possible default policies
  - that not expressly permitted is prohibited
  - that not expressly prohibited is permitted

Firewalls – Packet Filters(2/2)



(a) Packet-filtering router

Table 20.1 Packet-Filtering Examples

A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

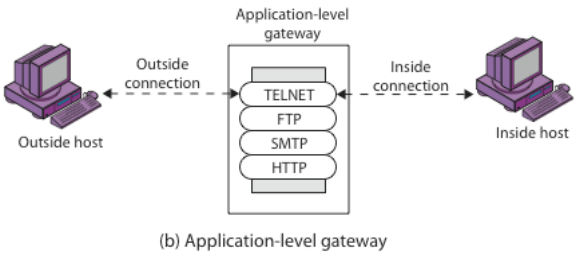
Attacks on Packet Filters

- IP address spoofing
  - fake source address to be trusted
  - add filters on router to block
- Source routing attacks
  - attacker sets a route other than default
  - block source routed packets
- Tiny fragment attacks
  - split header info over several tiny packets
  - either discard or reassemble before check

Firewalls - Application Level Gateway (or Proxy) (1/2)

- Have application specific gateway / proxy
- Has full access to protocol
  - user requests service from proxy
  - proxy validates request as legal
  - then actions request and returns result to user
  - can log / audit traffic at application level
- Need separate proxies for each service
  - some services naturally support proxying
  - others are more problematic

Firewalls - Application Level Gateway (or Proxy) (2/2)

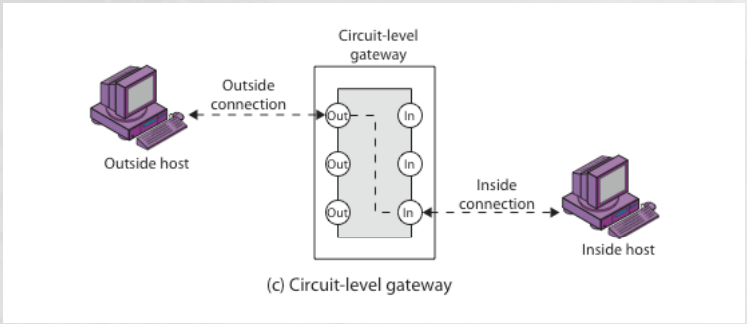


(b) Application-level gateway

Firewalls - Circuit Level Gateway(1/2)

- Relays two TCP connections
- Imposes security by limiting which such connections are allowed
- Once created usually relays traffic without examining contents
- Typically used when trust internal users by allowing general outbound connections
- SOCKS is commonly used

Firewalls - Circuit Level Gateway(2/2)

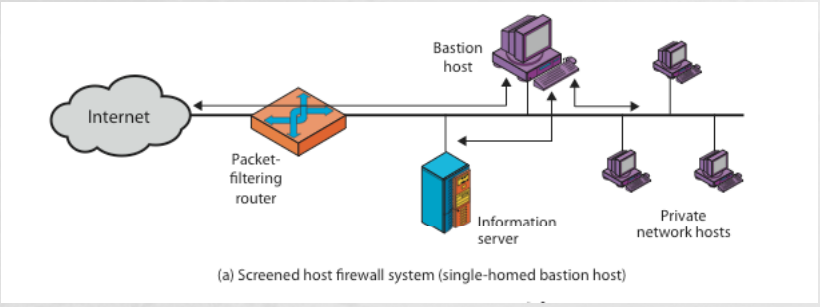


(c) Circuit-level gateway

Bastion Host

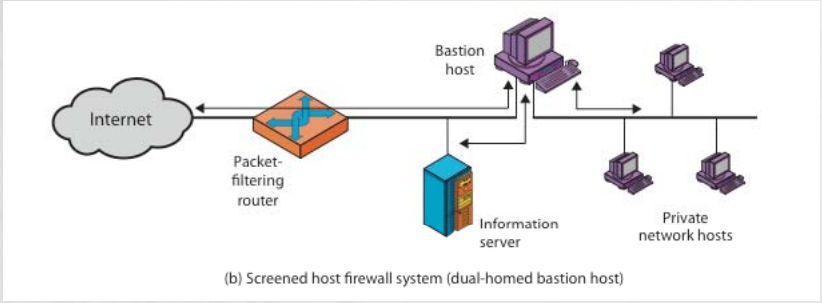
- Highly secure host system
- Runs circuit / application level gateways
  - or provides externally accessible services
- Potentially exposed to "hostile" elements
- Hence is secured to withstand this
  - hardened O/S, essential services, extra auth
  - proxies small, secure, independent, non-privileged
- May support 2 or more net connections
- May be trusted to enforce policy of trusted separation between these net connections

Firewall Configurations(1/3)



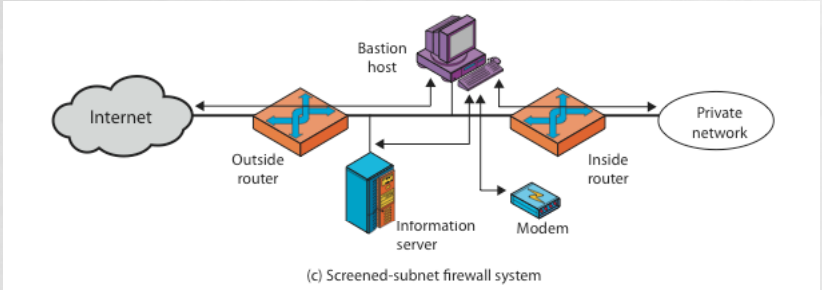
(a) Screened host firewall system (single-homed bastion host)

Firewall Configurations(2/3)



(b) Screened host firewall system (dual-homed bastion host)

Firewall Configurations(3/3)



(c) Screened-subnet firewall system

IDS

Intruders

- Significant issue for networked systems is hostile or unwanted access
- Either via network or local
- Can identify classes of intruders:
  - masquerader
  - misfeasor
  - clandestine user
- Varying levels of competence
- May use compromised system to launch other attacks
- Awareness of intruders has led to the development of CERTs

Intrusion Techniques

- Aim to gain access and/or increase privileges on a system
- Basic attack methodology
  - target acquisition and information gathering
  - initial access
  - privilege escalation
  - covering tracks
- Key goal often is to acquire passwords
  - so then exercise access rights of owner

❑31

Intrusion Detection

- Inevitably will have security failures
- So need also to detect intrusions can
  - block if detected quickly
  - act as deterrent
  - collect info to improve security
- Assume intruder will behave differently to a legitimate user
  - but will have imperfect distinction between

❑32

Approaches to Intrusion Detection

- Statistical anomaly detection
  - threshold
  - profile based
- Rule-based detection
  - anomaly
  - penetration identification

❑33

Statistical Anomaly Detection

- Threshold detection
  - count occurrences of specific event over time
  - if exceed reasonable value assume intrusion
  - alone is a crude & ineffective detector
- Profile-based
  - characterize past behavior of users
  - detect significant deviations from this
  - profile usually multi-parameter

❑34

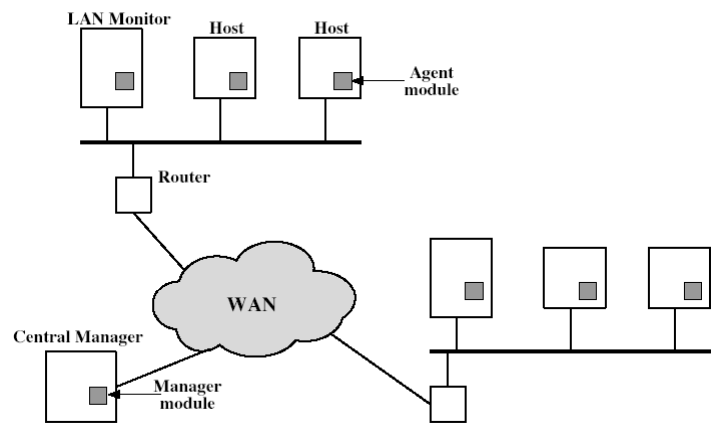
Rule-Based Intrusion Detection

- Observe events on system & apply rules to decide if activity is suspicious or not
- Rule-based anomaly detection
  - analyze historical audit records to identify usage patterns & auto-generate rules for them
  - then observe current behavior & match against rules to see if conforms
  - like statistical anomaly detection does not require prior knowledge of security flaws

Distributed Intrusion Detection

- Traditional focus is on single systems
  - but typically have networked systems
- More effective defense has these working together to detect intrusions
- Issues
  - dealing with varying audit record formats
  - integrity & confidentiality of networked data
  - centralized or decentralized architecture

Distributed Intrusion Detection - Architecture



❑37

IPS
(Intrusion Prevention System)

- ❑ Monitor network traffic
- ❑and/or system activities for malicious activities
  - ✓sending an alarm,
  - ✓dropping the malicious packets,
  - ✓resetting the connection
  - ✓and/or blocking the traffic from the offending IP address

❑38

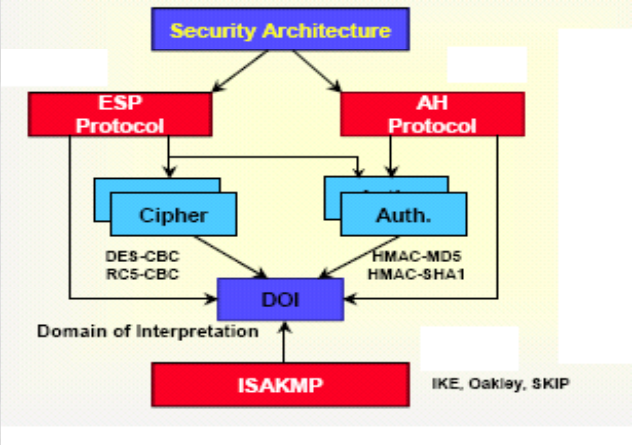
UTM
Unified Threat Management
= Firewall + AV + DRM + spam filter
etc.

IPSec

IPSec Design Objectives

- To Provide Interoperable, High quality, Crypto-based Security for IPv4 & IPv6
- Modularity
  - Designed to be algorithm-independent
  - Allows selection of different sets of algorithms without affecting the other part of the implementation
- Interoperability
  - Specify a standard set of default algorithms to facilitate interoperability in the global Internet
- Caveats
  - Security offered by IPSEC ultimately depends on the implementation quality
  - Security offered by IPSEC critically depends on many aspects of the operating environment, e.g., OS security, Random number generators, system management protocols, etc.

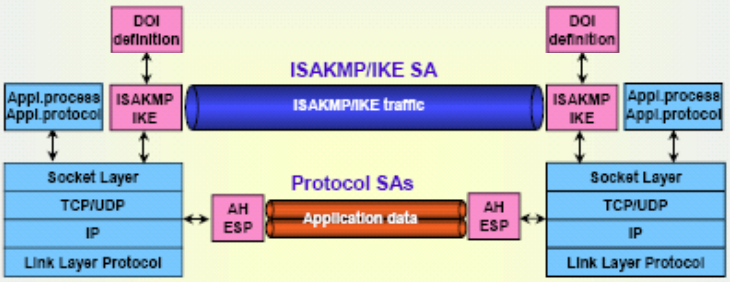
IPSec Standard Document



IPSec System Overview

- Two Security Protocols
  - AH primarily for authentication and optional anti-replay service
    - ✓ Mandatory-to-implement algorithms: HMAC-MD5, HMAC-SHA1
  - ESP primarily for confidentiality and optionally AH functionality (with limited protection range)
    - ✓ Mandatory-to-implement algorithms:
      - – DES-CBC (de facto: 3DES-CBC), NULL Encryption algorithm
      - – HMAC-MD5, HMAC-SHA1, NULL Authentication algorithm
  - AH & ESP are vehicles for access control
- Key Management
  - ISAKMP defines procedures and payload formats for SA/key management
  - Default automated SA/key management protocol for IPSEC:
    - – IKE (Internet Key Exchange) under IPSEC DOI
- Two Modes of Operations
  - Transport mode protects primarily upper layer protocols
  - Tunnel mode protects primarily tunneled IP packets

Operations of IPSec



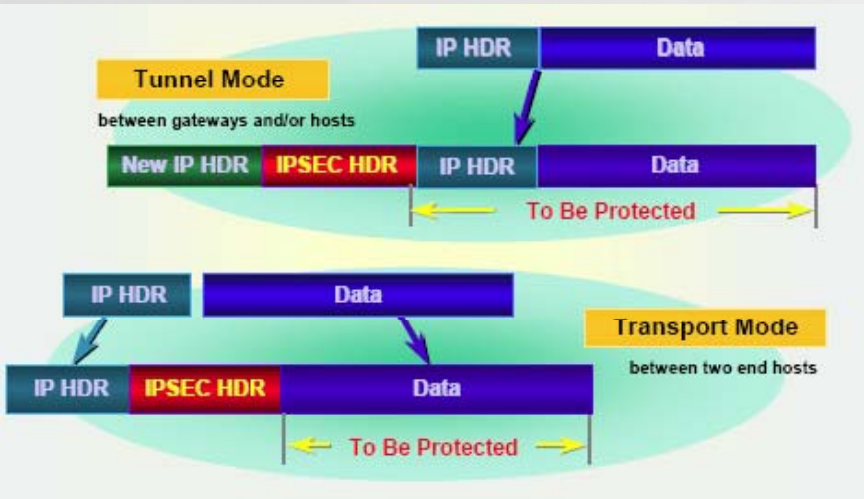
Phase I (ISAKMP SA) : SA negotiation between two ISAKMP servers

Phase II (Protocol SA) : SA negotiation for other security protocols (e.g., IPSEC AH) under the protection of ISAKMP SA

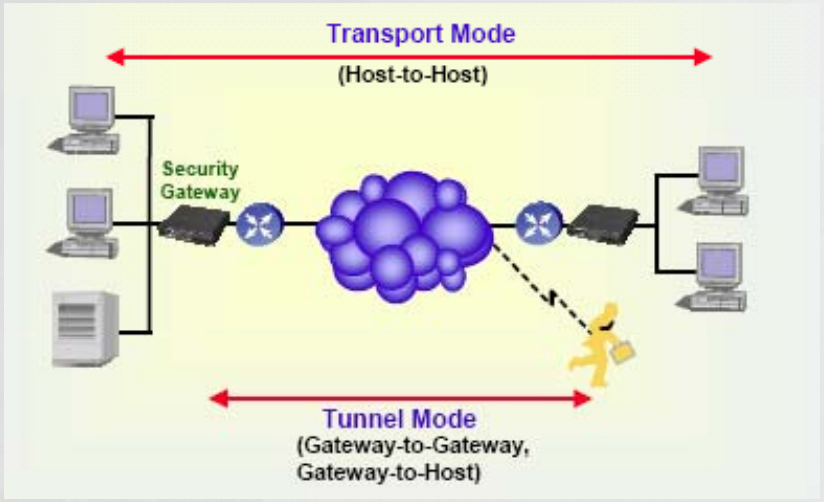
SA & SPI

- Security Association (SA)
  - A set of security parameters that completely defines the security services and mechanisms to be provided by the security protocol (IKE, AH or ESP).
  - E.g., authentication/encryption algorithm, algorithm mode and secret keys, etc.
  - uniquely identified by a triple (SPI, Destination IP addr, Security protocol).
  - receiver-oriented: the SPI is selected by the destination.
  - ISAKMP/IKE SA : bidirectional (identified by a pair of (I-Cookie, R-Cookie))
  - Protocol SA : unidirectional - one for inbound and one for outbound.
- Security Parameters Index (SPI)
  - An identifier for a SA relative to some security protocol (IPSEC: 32 bits)
  - Each security protocol has its own "SPI-space", and Initiator and Responder each select and exchange their own SPI during the security protocol negotiation.

IPSec Mode of Operation

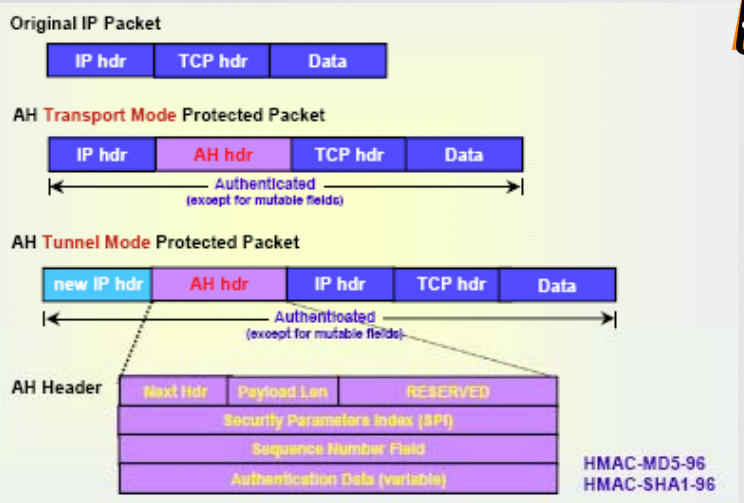


Tunnel Mode *vs* Transport Mode



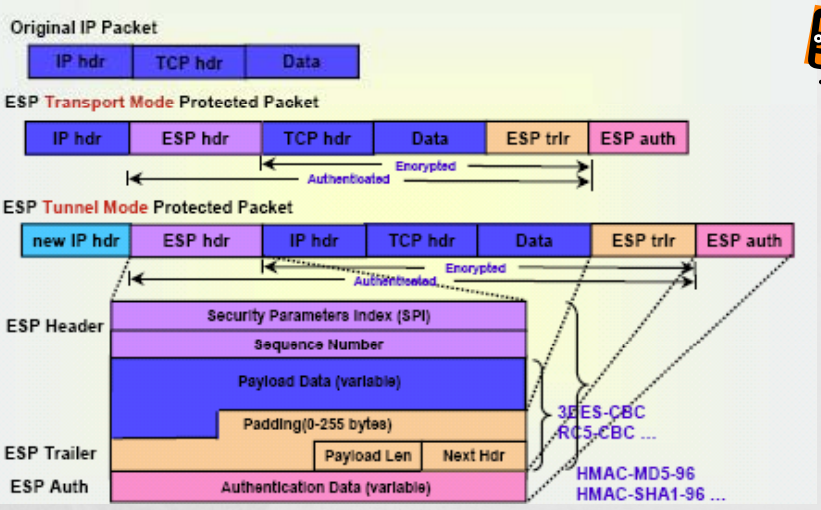
❑47

AH – Authentication Header



❑48

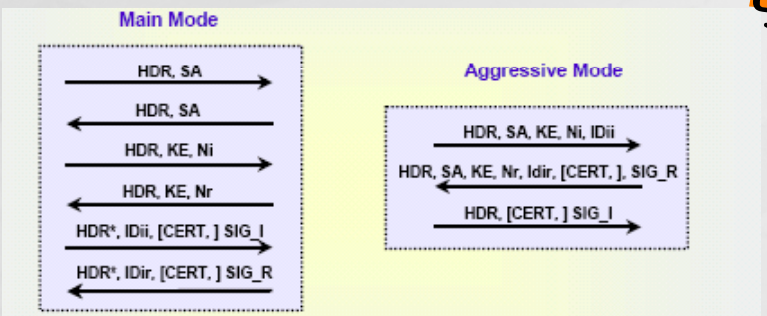
ESP- Encapsulating Security Payload



ISKAMP & IKE

- ISAKMP:
  - A general framework for establishing and managing SAs and keys
  - Header and payload definitions
  - Exchanges types for payload exchanges
  - General processing guidelines
- IKE:
  - A hybrid protocol to negotiate keys and SAs in an authenticated and protected manner
  - An authenticated key exchange algorithm based on Diffie-Hellman, with added authentication and security features from Oakley and SKEME techniques
  - Authentication methods supported
    - ✓ Pre-shared secret
    - ✓ Digital signature
    - ✓ Public key encryption
  - Exchange types defined
    - ✓ Aggressive mode
    - ✓ Main mode

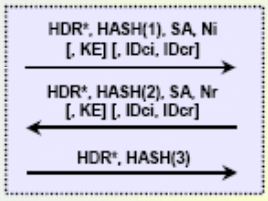
IKE Phase 1 Authentication with Signatures



- Derived SKEYIDs

SKEYID = prf(Ni_b | Nr_b, g^xy)
SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0) for key derivation
SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1) for authentication
SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2) for encryptiosn

IKE Phase 2 Quick Mode



HASH(1) = prf(SKEYID_a, M-ID | SA | Ni [| KE] [| IDci | IDcr])
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | SA | Nr [| KE] [| IDci | IDcr])
HASH(3) = prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)

- KEYMAT = prf(SKEYID_d, [g(qm)^xy |] protocol | SPI | Ni_b | Nr_b)

- The identities of SAs are implicitly assumed to be the IP address of the peers.
- If ISAKMP is acting a client negotiator on behalf of another party, the identities of the parties MUST be passed as IDci and IDcr.

- A single SA negotiation results in two SAs - one inbound and one outbound: The SPI chosen by destination of SA is used to derive KEYMAT for that SA

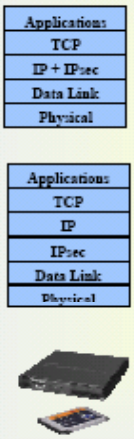
Security Policy Database(SPD)

- Specifies what services are to be offered to datagrams and in what fashion
  - Three processing choices discard, bypass IPsec, apply Ipsec
  - Used to map traffic to specific SAs or SA bundles
  - Must be consulted during of all traffic (inbound & outbound), including non-IPSEC traffic
  - SPD entries MUST be ordered and the SPD MUST always be searched in the same order
- Selectors
  - A set of IP and upper layer protocol field values that is used by SPD to map traffic to SA
  - Selector parameters
    - ✓ Source/Destination IP Address
    - ✓ Name
    - ✓ Data sensitivity level
    - ✓ Transport Layer Protocol
    - ✓ Source and Destination Ports

Security Association Database(SAD)

- SAD entry defines the parameters associated with one SA
  - For outbound processing, entries in SPD points to SAD entries
  - For inbound processing, triplet (SPI, IPSEC protocol, outer header's destination IP address) uniquely determines the SA
- If SPD entry does not currently point to an SA that is appropriate for the packet, the implementation creates an appropriates SA, and links SPD entry to SAD entry
- SAD fields
  - Sequence number counter / overflow, Anti-replay window
  - AH authentication algorithm, keys
  - ESP encryption algorithm, keys & IV
  - ESP authentication algorithm, keys
  - SA lifetime
  - IPSEC mode
  - Path MTU

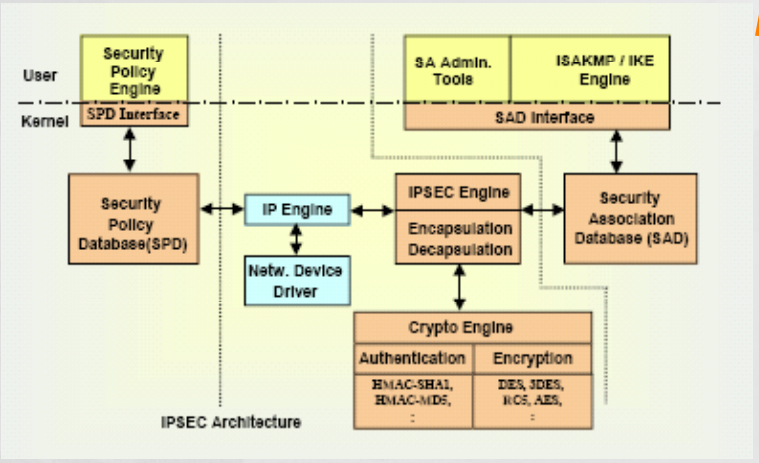
IPSec Implementation



- Integration into native IP implementation
  - ✓ Requires access to IP source code
  - ✓ implemented in host or gateway
- Bump-in-the-stack (BITS)
  - ✓ Between IP and local network driver
  - ✓ source code access for the IP stack not required
  - ✓ usually implemented in host
- Bump-in-the-wire (BITW)
  - ✓ outboard crypto processor, serve host or gateway
  - ✓ usually IP addressable
  - ✓ analogous to BITS in supporting host
  - ✓ like a security gateway in supporting router or firewall

❑55

IPSec Implementation Model: An Example

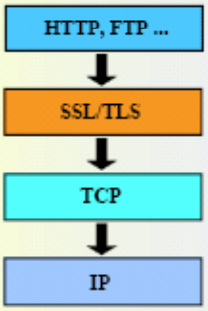


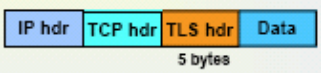
❑56

TLS
(Transport Layer Security) Protocol

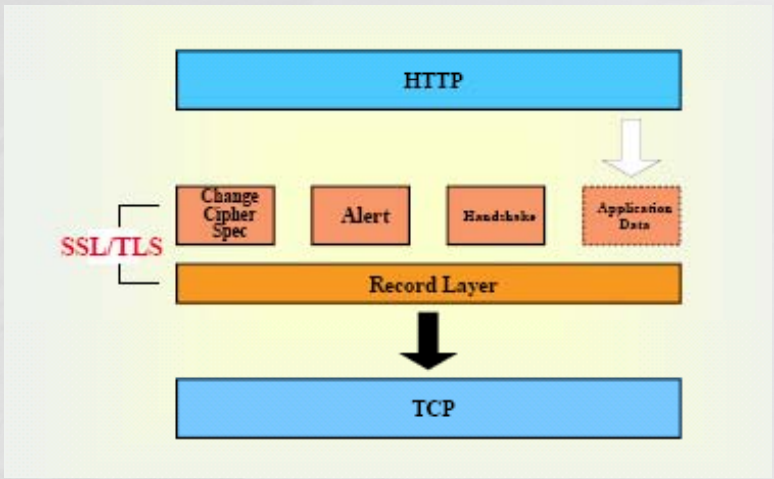
TLS protocol

- **SSL/TLS**
  - Layered on top of reliable transport protocols, e.g., TCP
  - Application protocol independent
  - Record Protocol & Handshake Protocol
- **Record Protocol**
  - Encapsulation of higher level protocols
  - Data encryption using CBC block ciphers or stream ciphers
  - Data integrity using HMAC
- **Handshake Protocol**
  - Security parameter negotiation: keys & algorithms
  - Entity authentication using public key cryptography (RSA, DSS; static DH)
  - Key exchange & verification (RSA key transport, DH key exchange)

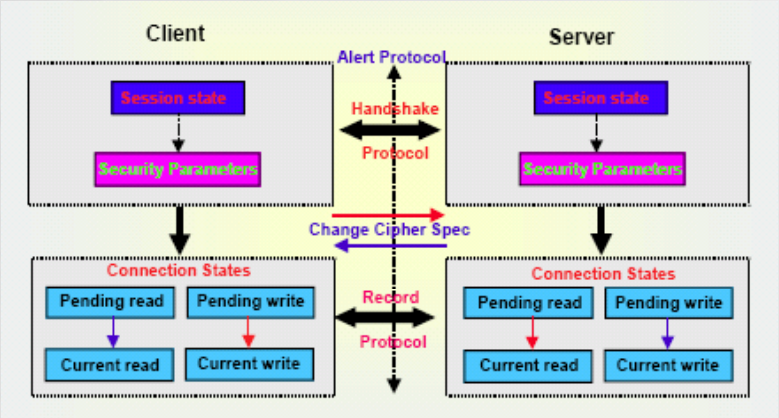




SSL/ TLS layering



SSL/ TLS operations overview



TLS Session State

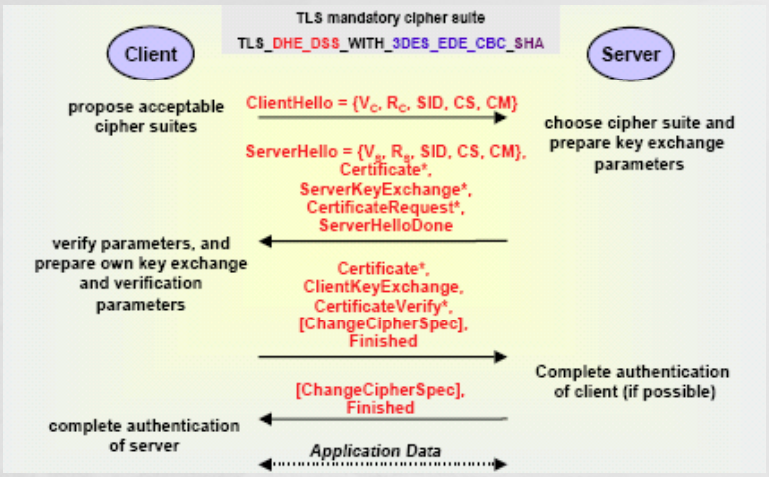
- used to create security parameters for use by the Record Layer (TLS)
- Session state consists of
  - session identifier : An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
  - peer certificate: X509v3 [X509] certificate of the peer. may be null.
  - compression method : The algorithm used to compress data prior to encryption.
  - cipher spec : Specifies the bulk data encryption algorithm (such as null, DES, etc.) and a MAC algorithm (such as MD5 or SHA). It also defines cryptographic attributes such as the hash_size. (See Appendix A.6 for formal definition)
  - master secret : 48-byte secret shared between the client and server.
  - is resumable : A flag indicating whether the session can be used to initiate new connections.

TLS security parameters

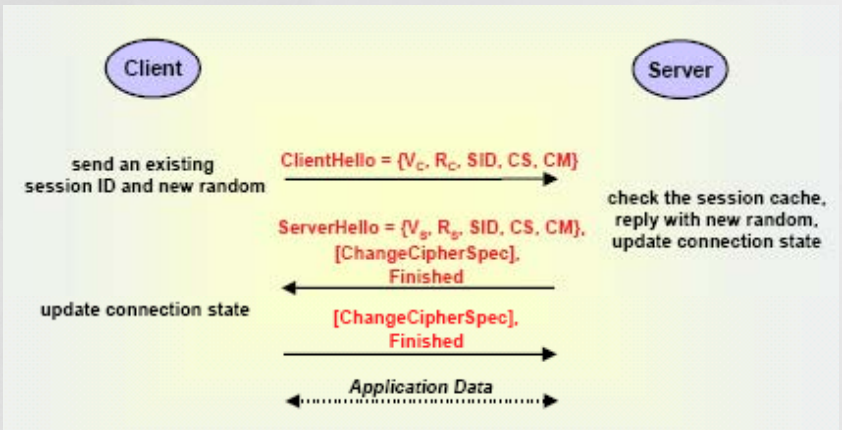
```
struct {
        ConnectionEnd           entity;                 /* server, client */
        BulkCipherAlgorithm     bulk_cipher_algorithm;  /* NULL, rc4, rc2, des, 3des, des40 */
        CipherType              cipher_type;            /* stream, block */
        uint8                   IV_size;                /* IV size for block ciphers */
        uint8                   key_material_length;    /* write key size */
        IsExportable            is_exportable;          /* true, false */
        MACAlgorithm            mac_algorithm;          /* NULL, md5, sha */
        uint8                   hash_size;              /* MAC secret size */
        CompressionMethod       compression_algorithm;  /* NULL */
        opaque                  master_secret[48];      /* 48 byte master secret */
        opaque                  client_random[32];      /* Random from ClientHello */
        opaque                  server_random[32];      /* Random from ServerHello */
} SecurityParameters;
```

(CipherSpec: BulkCipherAlgorithm, CipherType, uint8, uint8, IsExportable, MACAlgorithm, uint8)

TLS full handshake



TLS abbreviated handshake



Client Hello/ Server Hello

```
struct {
      uint8     major, minor;
    } ProtocolVersion;          /* SSL: {3,0}, TLS: {3, 1} */

struct {
      uint32   gmt_unix_time;
      opaque   random_bytes[28];
    } Random;
opaque         SessionID<0..32>;

uint8          CipherSuite[2];      /* Cryptographic suite selector */

enum { null(0), (255) } CompressionMethod;

struct {
      ProtocolVersion       client_version;
      Random                random;
      SessionID             session_id;
      CipherSuite           cipher_suites<2..2^16-1>;
      CompressionMethod     compression_methods<1..2^8-1>;
    } ClientHello;
```

TLS mandatory cipher suite
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

TLS cipher suites – RSA key exchange

CipherSuite	Key Exchange	Cipher	Hash
TLS_NULL_WITH_NULL_NULL	* NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	* RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	* RSA	NULL	SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5	* RSA_EXPORT	RC4_40	MD5
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	* RSA_EXPORT	RC2_CBC_40	MD5
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA_CBC	SHA
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	* RSA_EXPORT	DES40_CBC	SHA
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA

TLS cipher suites – DH key exchange

CipherSuite	Key Exchange	Cipher	Hash
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	* DH_DSS_EXPORT	DES40_CBC	SHA
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES_CBC	SHA
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE_CBC	SHA
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	* DH_RSA_EXPORT	DES40_CBC	SHA
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES_CBC	SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE_CBC	SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	* DHE_DSS_EXPORT	DES40_CBC	SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES_CBC	SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE_CBC	SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	* DHE_RSA_EXPORT	DES40_CBC	SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES_CBC	SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE_CBC	SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	* DH_anon_EXPORT	RC4_40	MD5
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH_anon	DES40_CBC	SHA
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES_CBC	SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE_CBC	SHA

Certificate

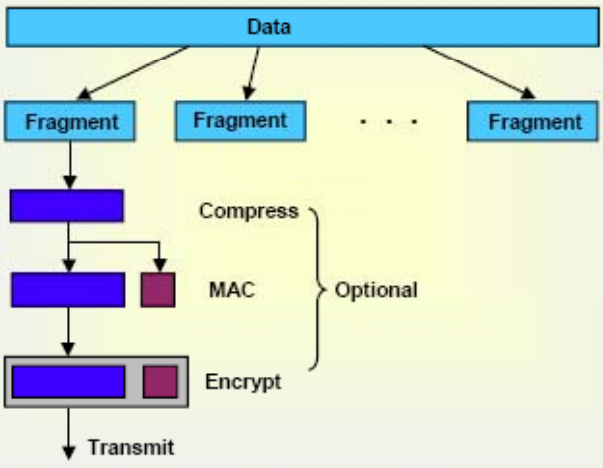
```
opaque   ASN.1Cert<1..2^24-1>;

struct {
      ASN.1Cert    certificate_list<0..2^24-1>;
   } Certificate;
```

Key Exchange Algorithm	Certificate Key Type
RSA	RSA public key for encryption
RSA_EXPORT	RSA public key (≥ 512bits for signing, ≤ 512 bits for either encryption or signing)
DHE_DSS	DSS public key.
DHE_DSS_EXPORT	DSS public key.
DHE_RSA	RSA public key for signing
DHE_RSA_EXPORT	RSA public key for signing
DH_DSS	Diffie-Hellman key signed by CA using DSS.
DH_RSA	Diffie-Hellman key signed by CA using RSA

TLS/SSL key exchange algorithms

Key Exchange Alg.	DH_DSS, DH_DSS_EXPORT / DH_RSA, DH_RSA_EXPORT		
Certificate(S)	DH (signed by CA's DSS/RSA key)		
ServerKeyExchange	None		
CertificateRequest	If dss_sign/rsa_sign,	If rsa_fixed_dh/dss_fixed_dh	If None
Certificate(C)	DSS/RSA	DH	None
ClientKeyExchange	Ephemeral DH key	empty	Ephemeral DH key
CertificateVerify	DSS/RSA signature	None	None
Key Exchange Alg.	**DHE_DSS, DHE_DSS_EXPORT / DHE_RSA, DHE_RSA_EXPORT**		
Certificate(S)	DSS / RSA (signed by CA's DSS/RSA key)		
ServerKeyExchange	DSS/RSA-signed ephemeral DH key		
CertificateRequest	If dss_sign/rsa_sign,	If None	
Certificate(C)	DSS/RSA	None	
ClientKeyExchange	Ephemeral DH key	Ephemeral DH key	
CertificateVerify	DSS/RSA signature	None	
Key Exchange Alg.	**RSA, RSA_EXPORT (for encryption)**		
Certificate(S)	RSA(e) (RSA(e) for RSA_EXPORT)		
ServerKeyExchange	None (RSA-signed temporary short RSA key)		
CertificateRequest	If dss_sign/rsa_sign,	If None	
Certificate(C)	DSS/RSA(s)	None	
ClientKeyExchange	RSA-encrypted pre-master secret	RSA-encrypted pre-master secret	
CertificateVerify	DSS/RSA signature	None	

TLS record protocol



TLS key derivation

- PRF (Pseudo-Random Function):

  P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
  HMAC_hash(secret, A(2) || seed) ||
  HMAC_hash(secret, A(3) || seed) || ...
  (A(0) = seed, A(i) = HMAC_hash(secret, A(i-1))

  PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕ P_SHA-1(S2, label || seed)
  (S1, S2 : 1st and 2nd half of secret)

- Master Secret (48 bytes) -- in Handshake protocol

  master_secret = PRF(pre_master_secret, "master secret",
  ClientHello.random || ServerHello.random) [0..47];

- Key Block -- in Record protocol

  key_block = PRF(SecurityParameters.master_secret, "key expansion",
  SecurityParameters.server_random || SecurityParameters.client_random)
  = client_write_MAC_secret[SecurityParameters.hash_size] ||
  server_write_MAC_secret[SecurityParameters.hash_size] ||
  client_write_key[SecurityParameters.key_material_length] ||
  server_write_key[SecurityParameters.key_material_length] ||
  client_write_IV[SecurityParameters.IV_size] ||
  server_write_IV[SecurityParameters.IV_size]

IPSec vs. TLS/SSL

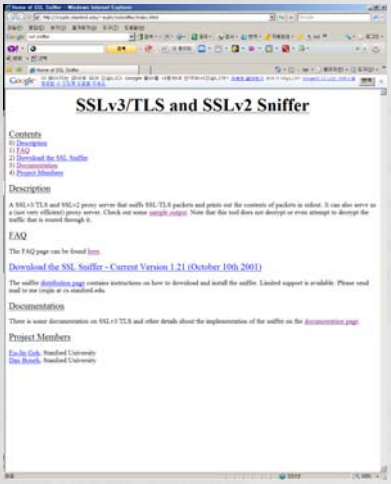
IPSEC

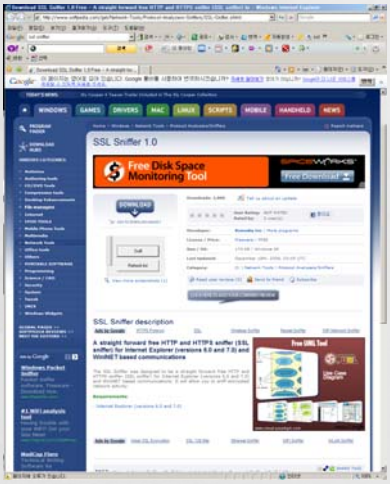
- Network layer security protocol
- Confidentiality, Integrity, Authentication, Access control, Auditing
- Transport protocol independent
- No change to applications (application/user transparency)
- Peer-to-Peer model: Host-to-Server, Host-to-Subnet, Subnet-to-Subnet
- More secure; too complex, special client SW
- IPv4 (optional), IPv6 (mandatory)

SSL/TLS

- Transport layer security protocol
- Confidentiality, Integrity, Authentication (usually client-to-server only)
- Works only with TCP (not UDP): HTTP, SMTP, POP3, NNTP, FTP, LDAP...
- Minimal changes to applications
- Client-Server model: Host-to-Server (secure Web transactions)
- Free : built in to nearly all browsers and Web servers

Sniffing SSL





❑73