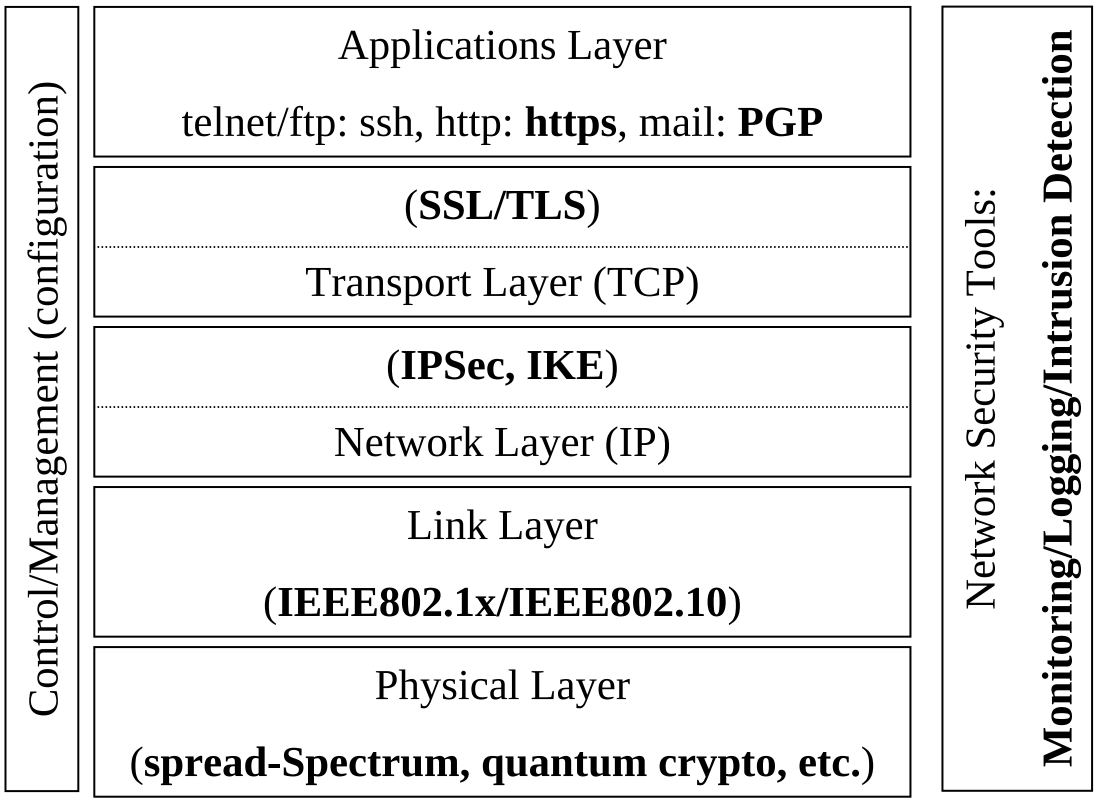
IPsec (AH, ESP), IKE

Guevara Noubir

CSG254: Network Security

noubir@ccs.neu.edu

Securing Networks



SSL vs. IPsec

- SSL:
  - Avoids modifying “TCP stack” and requires minimum changes to the application
  - Mostly used to authenticate servers
- IPsec
  - Transparent to the application and requires modification of the network stack
  - Authenticates network nodes and establishes a secure channel between nodes
  - Application still needs to authenticate the users

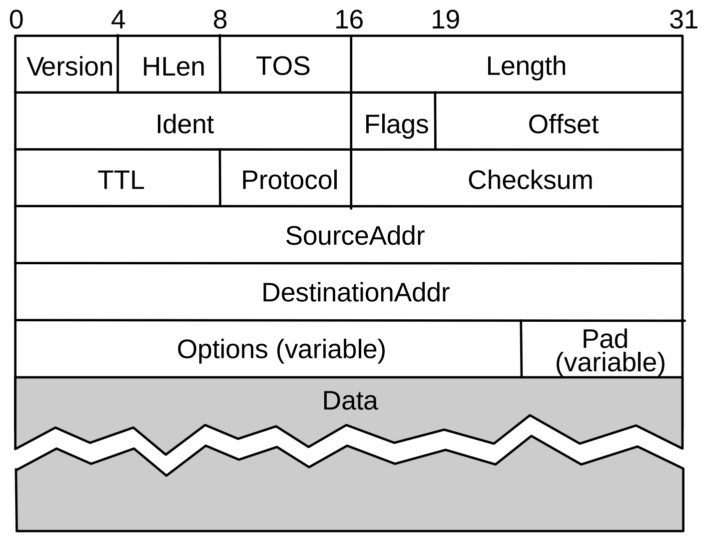
IPsec Protocol Suite (IETF Standard)

- Provides inter-operable crypto-based security services:
  - Services: confidentiality, authentication, integrity, and key management
  - Protocols:
    - Authentication Header (AH): RFC2402
    - Encapsulated Security Payload (ESP): 2406
    - Internet Key Exchange (IKE)
  - Environments: IPv4 and IPv6
  - Modes:
    - Transport (between two hosts)
    - Tunnel (between hosts/firewalls)

IPsec

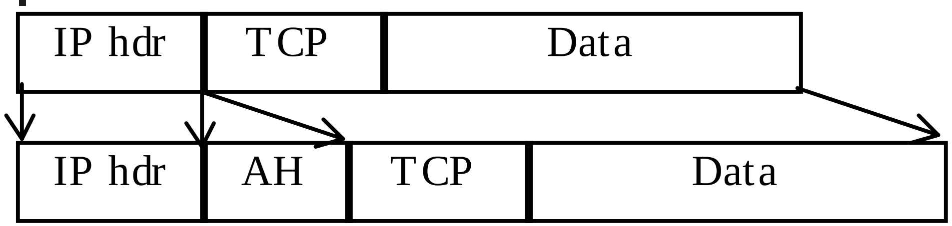
- Assumption:
  - End nodes already established a shared session key:
    - Manually or IKE
- Security Association:
  - Each secure connection is called a *security association* (SA)
  - For each SA: key, end-node, sequence number, services, algorithms
  - SA is unidirectional and identified by:
    - (destination-address, SPI = Security Parameter Index)
- Protocols:
  - Authentication Header: integrity protection
  - Encapsulated Security Payload: encryption and/or integrity

IP Packets



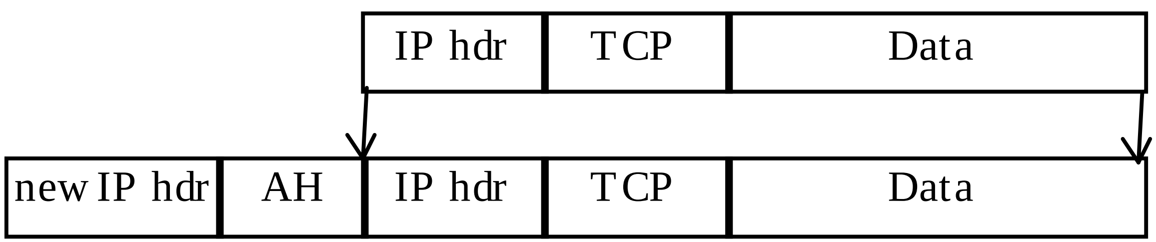
0
4
8
16
19
31
Version
HLen
TOS
Length
Ident
Flags
Offset
TTL
Protocol
Checksum
SourceAddr
DestinationAddr
Options (variable)
Pad (variable)
Data

AH Formatting



Transport mode

AH Protocol Number = 51



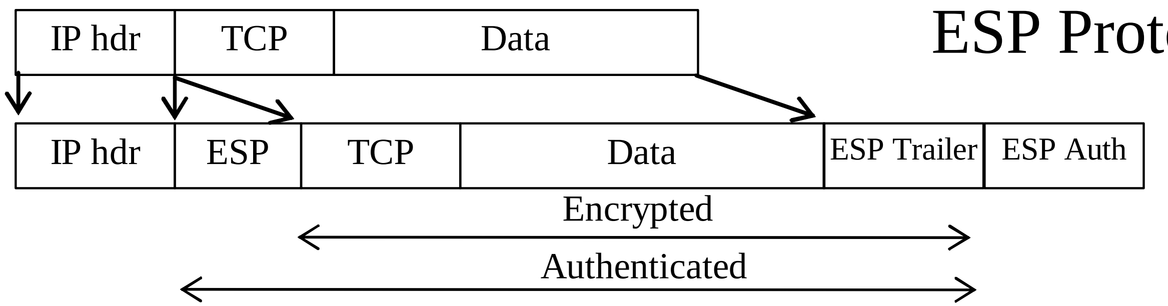
Tunnel mode

Next Header	Length (8)	Reserved (16)
Security Parameters Index (32)		
Sequence Number Field (32)		
Authentication Data (N*32)		

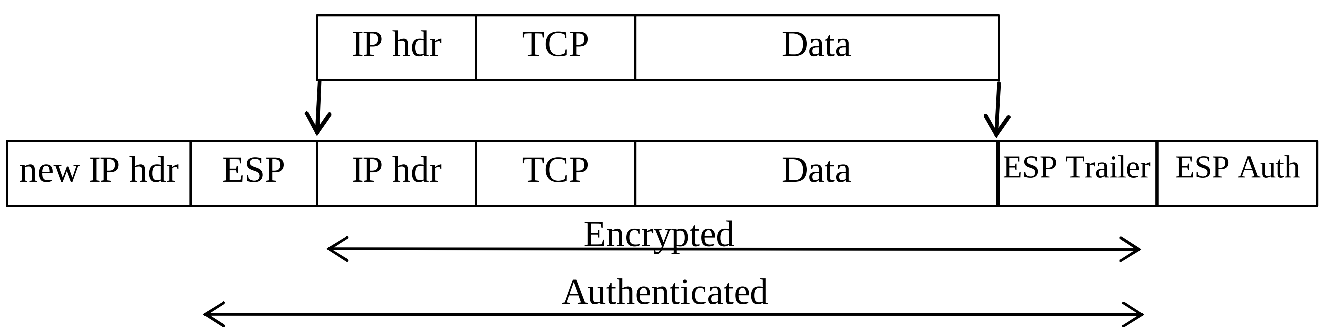
SN: for replay detection

ESP Formatting

ESP Protocol Number = 50

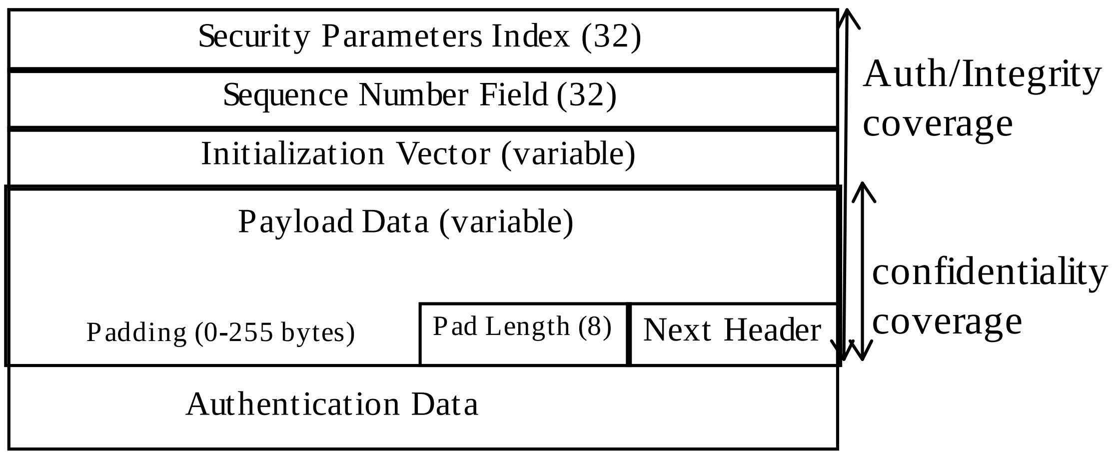


Transport mode



Tunnel mode

ESP Header


Security Parameters Index (32)
Sequence Number Field (32)
Initialization Vector (variable)
Payload Data (variable)
Padding (0-255 bytes)
Pad Length (8)
Next Header
Authentication Data
Auth/Integrity coverage
confidentiality coverage

Issues

- NAT boxes:
  - IPsec tunnel mode doesn't easily work
- Firewalls
  - IPsec encrypts information used by firewalls to filter traffic (e.g., port number)
- AH mutable/immutable/predictable fields:
  - Some fields get modified by the intermediate routers and can't be protected by the AH
  - Mutable: type of service, flags, fragment offset, TTL, header checksum
  - Why is PAYLOAD-LENGTH considered immutable (even if packets can be fragmented)? Why not fragment offset. Inconsistency!
  - Mutable but predictable fields are included in the AH computation using their expected value at the destination (e.g., destination address even when using source routing)

IPsec: Internet Key Exchange

- Goal:
  - Mutual authentication and establishment of a shared secret session key using:
    - Pre-shared secret key or public signature-only key, or public encryption key
  - Negotiation of features and cryptographic algorithms
- Specification documents:
  - ISAKMP (Internet Security Association and Key Management Protocol): RFC 2408
  - IKE: RFC 2409
  - DOI (Domain Of Interpretation): RFC 2407

Photuris

- Photuris goal: signed Diffie-Hellman exchange
  1. A -> B: C_A
  2. B -> A: C_A, C_B, crypto offered
  3. A -> B: C_A, C_B, g^a mod p, crypto selected
  4. B -> A: C_A, C_B, , g^b mod p
  5. A -> B: C_A, C_B, g^{ab} mod p{A, signature on previous messages}
  6. B -> A: C_A, C_B, , g^{ab} mod p{B, signature on previous messages}

  - Role of C_A, C_B, and messages
  - Additional features: SPI selection
  - Why not sign messages 3 & 4...?

Simple Key-Management for Internet Protocol (SKIP)

- Uses long term Diffie-Hellman keys
- Parties assumed to know each other public keys (i.e., $g^a \bmod p$) or exchange certificates
- Session key $X = g^{ab} \bmod p$ is established in 0 messages
- Each packet is encrypted using data key S and each packet contains: $X\{S\}$
  - Same S can be used for several packets
- Later on PFS was added by periodically forgetting the keys and doing a new DH

ISAKMP (RFC2408)

- Proposed by NSA as a framework and accepted by IETF
  - Runs over UDP and allows to exchange fields to create a protocol
- IKE (RFC2409) based on OAKLEY & SKEME using ISAKMP syntax

- IKE phases:
  1. Mutual authentication and session key establishment (also called ISAKMP SA or IKE SA)
  2. AH/ESP SAs establishment

– Each source/destination/port has its own SA/keys otherwise ESP traffic not using integrity could be decrypted...

Phase 1 IKE

- Two modes:
  - Aggressive mode: mutual authentication and session key establishment in three messages
    - A -> B: g^a mod p, A, crypto proposal
    - B -> A: g^b mod p, crypto choice, proof I'm B
    - A -> B: proof I'm A
  - Main: additional features such as hiding end-points identities and negotiating crypto DH algorithm
    - A -> B: crypto suite I support
    - B -> A: crypto suite I choose
    - A -> B: g^a mod p
    - B -> A: g^b mod p
    - A -> B: g^{ab} mod p {A, proof I'm A}
    - B -> A: g^{ab} mod p {B, proof I'm B}

Phase 1 IKE

- Key types:
  - Pre-shared secret key
  - Public encryption key: fields are separately encrypted using the public key
  - Optimized public encryption key: used to encrypt a random symmetric key, and then data is encrypted using the symmetric key
  - Public signature key: used only for signature purpose

⇒ 8 variants of IKE phase 1: 2 modes x 4 key types

- Proof of Identity:
  - Required in messages 2-3 aggressive mode and 5-6 main mode
  - Proves the sender knows the key associated with the identity
  - Depends on the key type
  - Hash of identity key, DH values, nonces, crypto choices, cookies
  - Alternative: MAC of previous messages

Phase 1 IKE

- Negotiating cryptographic parameters
  - *A* specifies suites of acceptable algorithms:
    - {(3DES, MD5, RSA public key encryption, DH), (AES, SHA-1, pre-shared key, elliptic curve), …}
  - The standard specifies a MUST be implemented set of algorithms:
    - Encryption=DES, hash=MD5/SHA-1, authentication=pre-shared key/DH
  - The lifetime of the SA can also be negotiated
- Session keys:
  - Key seed: SKEYID
    - Signature public keys: SKEYID = prf(nonces, g^{xy} mod p)
    - Encryption public keys: prf(hash(nonces), cookies)
    - Pre-shared secret key: prf(pre-shared secret key, nonces)
  - Secret to generate other keys: SKEYID_d = prf(SKEYID, (g^{xy}, cookies, 0))
  - Integrity key: SKEYID_a = prf(SKEYID, (SKEYID_d, (g^{xy}, cookies, 1)))
  - Encryption key: SKEYID_e = prf(SKEYID, (SKEYID_a, (g^{xy}, cookies, 2))
- Message IDs:
  - Random 32-bits serves the purpose of a SN but in an inefficient manner because they have to be remembered

IKE Phase 1: Public Signature Keys, Main Mode

- Description:
  - Both parties have public keys for signatures
  - Hidden endpoint identity (except for ...?)
- Protocol:
  - A -> B: CP
  - B -> A: CPA
  - A -> B: g^a mod p, nonce_A
  - B -> A: g^b mod p, nonce_B

  $K = f(g^{ab}$ mod p, nonce_A, $\text{nonce}_B)$
  - A -> B: K{A, proof I'm A, [certificate]}
  - B -> A: K{B, proof I'm B, [certificate]}
- Questions:
  - What is the purpose of the nonces?
  - Can we make to protocol shorter (5 messages)? At what expense?

IKE Phase 1: Public Signature Keys, Aggressive Mode

- Protocol:
  - A -> B: CP, g^a mod p, nonce$_A$, A
  - B -> A: CPA, g^b mod p, nonce$_B$, B, proof I'm B, [certificate]
  - A -> B: proof I'm A, [certificate]

IKE Phase 1: Public Encryption Keys, Main Mode, Original

- Protocol:
  - A -> B: CP
  - B -> A: CPA
  - A -> B: g^a mod p, $\{\text{nonce}_A\}_B$, $\{A\}_B$
  - B -> A: g^b mod p, $\{\text{nonce}_B\}_A$, $\{B\}_A$

  $K = f(g^{ab} \bmod p, \text{nonce}_A, \text{nonce}_B)$
  - A -> B: K\{proof I'm A\}
  - B -> A: K\{proof I'm B\}

IKE Phase 1: Public Encryption Keys, Aggressive Mode, Original

- Protocol:
  - A -> B: CP, g^a mod p, $\{\text{nonce}_A\}_B$, $\{A\}_B$
  - B -> A: CPA, g^b mod p, $\{\text{nonce}_B\}_A$, $\{B\}_A$, proof I'm B
  - A -> B: proof I'm A

IKE Phase 1: Public Encryption Keys, Main Mode, Revised

- Protocol:
  - A -> B: CP
  - B -> A: CPA

    K_A = hash(nonce$_A$, cookie$_A$)
  - A -> B: {nonce$_A$}$_B$, K_A{g^a mod p}, K_A{A}, [K_A{A's cert}]

  K_B = hash(nonce$_B$, cookie$_B$)
  - B -> A: {nonce$_B$}$_A$, K_B{g^b mod p}, K_B{B}

  K = f(g^{ab} mod p, nonce$_A$, nonce$_B$, cookie$_A$, cookie$_B$)
  - A -> B: K{proof I'm A}
  - B -> A: K{proof I'm B}

IKE Phase 1: Public Encryption Keys, Aggressive Mode, Revised

- Protocol:

  K_A = hash(nonce$_A$, cookie$_A$)

  - A -> B: CP, {nonce$_A$}$_B$, K_A{g^a mod p}, K_A{A}, [K_A{A's cert}]

  K_B = hash(nonce$_B$, cookie$_B$)

  - B -> A: CPA, {nonce$_B$}$_A$, K_B{g^b mod p}, K_B{B}, proof I'm B

  $K = f(g^{ab}$ mod p, nonce$_A$, nonce$_B$, cookie$_A$, cookie$_B$)

  - A -> B: K{proof I'm A}

IKE Phase 1: Shared Secret Keys, Main Mode

- Assumption A and B share a secret J
- Protocol:
  - A -> B: CP
  - B -> A: CPA
  - A -> B: $g^a \bmod p$, nonce_A
  - B -> A: $g^b \bmod p$, nonce_B

  $K = f(J, g^{ab} \bmod p, \text{nonce}_A, \text{nonce}_B, \text{cookie}_A, \text{cookie}_B)$
  - A -> B: K\{proof I'm A\}
  - B -> A: K\{proof I'm B\}

IKE Phase 1: Shared Secret Keys, Aggressive Mode

- Protocol:
  - A -> B: CP, g^a mod p, nonce$_A$, A
  - B -> A: CPA, g^b mod p, nonce$_B$, B, proof I'm B
  - A -> B: proof I'm A

IKE: Phase 2

- Also known as “Quick Mode”: 3- messages protocol
  - *A* -> *B*: *X*, *Y*, *CP*, *traffic*, SPI_A, $nonce_A$, $[g^a \bmod p]_{optional}$
  - *B* -> *A*: *X*, *Y*, *CPA*, *traffic*, SPI_B, $nonce_B$, $[g^b \bmod p]_{optional}$
  - *A* -> *B*: *X*, *Y*, *ack*
- All messages are encrypted using SKEYID_e, and integrity protected using SKEYID_a (except *X*, *Y*)
- Parameters:
  - *X*: pair of cookies generated during phase 1
  - *Y*: 32-bit number unique to this phase 2 session chosen by the initiator
  - CP: Crypto Proposal, CPA: Crypto Proposal Accepted
  - DH is optional and could be used to provide PFS
  - Nonces and cookies get shuffled into SKEYID to produce the SA encryption and integrity keys