IP Security (IPsec):
Internet Key Exchange IKEv2

Internet Key Exchange (IKE)

- IKEv2 [RFC 7296]: authenticated key exchange for IPsec
  - Diffie-Hellman or ECDH, SIGMA (sign and MAC) protocol
  - Minimum two request-response exchanges (4 messages, 2 RTT)
  - Works over UDP port 500
- Initial exchanges create the IKE security association (IKE SA) for (re)keying and one IPsec SA pair for session data
  - CREATE_CHILD_SA exchange for later rekeying
- Endpoints: initiator I and responder R
  - Initiator can be the client or server (why?)

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, 0, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r ,
   $Sign_{\mathbf{i}}$ (Message1, N_r, $MAC_{SK}(ID_i)$), SA_{i2}, TS_i, TS_r, $MAC_{SK}(\ldots)$)
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$,
   $Sign_{\mathbf{R}}$ (Message2, N_i, $MAC_{SK}(ID_r)$), SA_{r2}, TS_i, TS_r, $MAC_{SK}(\ldots)$)

SPI_x = two values that together identify the protocol run and the created IKE SA
SA_{x1} = offered and chosen algorithms, DH or ECDH group
SK = h(Ni, Nr, g^{xy}) — actually, many different keys are derived from this
$Sign_{\mathbf{x}}$ ($Message_x$, N_y, $MAC_{SK}(ID_x)$) – SIGMA authentication
ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs
SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)
$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, 0, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r , $Sign_{\mathbf{i}}$ (Message1, N_r, $MAC_{SK}(ID_i))$, SA_{i2}, TS_i, TS_r, $MAC_{SK}(\ldots))$
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$, $Sign_{\mathbf{R}}$ (Message2, N_i, $MAC_{SK}(ID_r))$, SA_{r2}, TS_i, TS_r, $MAC_{SK}(\ldots))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

SK = h(Ni, Nr, g^{xy}) — many different keys are derived from the key material

$Sign_{\mathbf{x}}$ ($Message_x$, N_y, $MAC_{SK}(ID_x)$) – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. $I \rightarrow R$: SPI_i, 0, SA_{i1}, g^x, N_i
2. $R \rightarrow I$: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. $I \rightarrow R$: SPI_i, SPI_r, $E_{SK}(ID_i, CERT_i, CERTREQ_i, ID_r,$
   $Sign_i (Message1, N_r, MAC_{SK}(ID_i)), SA_{i2}, TS_i, TS_r, MAC_{SK}(\ldots))$
4. $R \rightarrow I$: SPI_i, SPI_r, $E_{SK}(ID_r, CERT_r,$
   $Sign_R (Message2, N_i, MAC_{SK}(ID_r)), SA_{r2}, TS_i, TS_r, MAC_{SK}(\ldots))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

$SK = h(Ni, Nr, g^{xy})$ — actually, many different keys are derived from this

$Sign_x (Message_x, N_y, MAC_{SK}(ID_x))$ – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, 0, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r , $Sign_i$ (Message1, N_r, $MAC_{SK}(ID_i)$), SA_{i2}, TS_i, TS_r, $MAC_{SK}(...))$
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$, $Sign_R$ (Message2, N_i, $MAC_{SK}(ID_r)$), SA_{r2}, TS_i, TS_r, $MAC_{SK}(...))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

SK = $h(Ni, Nr, g^{xy})$ — actually, many different keys are derived from this

$Sign_x$ ($Message_x$, N_y, $MAC_{SK}(ID_x)$) – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(..., MAC_{SK}(...))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. $I \rightarrow R$: SPI_i, 0, SA_{i1}, g^x, N_i
2. $R \rightarrow I$: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. $I \rightarrow R$: SPI_i, SPI_r, $E_{SK}(ID_i, CERT_i, CERTREQ_i, ID_r,$
   $Sign_i(Message1, N_r, MAC_{SK}(ID_i))$, $SA_{i2}, TS_i, TS_r, MAC_{SK}(\ldots))$
4. $R \rightarrow I$: SPI_i, SPI_r, $E_{SK}(ID_r, CERT_r,$
   $Sign_R(Message2, N_i, MAC_{SK}(ID_r))$, $SA_{r2}, TS_i, TS_r, MAC_{SK}(\ldots))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

$SK = h(Ni, Nr, g^{xy})$ — actually, many different keys are derived from this

$Sign_x(Message_x, N_y, MAC_{SK}(ID_x))$ – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. $I \rightarrow R$: SPI_i, 0, SA_{i1}, g^x, N_i
2. $R \rightarrow I$: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. $I \rightarrow R$: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r ,
   $Sign_i$ $(Message1, N_r, MAC_{SK}(ID_i))$, SA_{i2}, TS_i, TS_r, $MAC_{SK}(...))$
4. $R \rightarrow I$: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$,
   $Sign_R$ $(Message2, N_i, MAC_{SK}(ID_r))$, SA_{r2}, TS_i, TS_r, $MAC_{SK}(...))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

SK = $h(Ni, Nr, g^{xy})$ — actually, many different keys are derived from this

$Sign_x$ $(Message_x, N_y, MAC_{SK}(ID_x))$ – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(..., MAC_{SK}(...))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, 0, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r ,
   $Sign_i$ (Message1, N_r, $MAC_{SK}(ID_i)$), SA_{i2}, TS_i, TS_r, $MAC_{SK}(\ldots))$
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$,
   $Sign_R$ (Message2, N_i, $MAC_{SK}(ID_r)$), SA_{r2}, TS_i, TS_r, $MAC_{SK}(\ldots))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

SK = $h(Ni, Nr, g^{xy})$ — actually, many different keys are derived from this

$Sign_x$ ($Message_x$, N_y, $MAC_{SK}(ID_x)$) – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = Authenticated encryption for identity protection

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, 0, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r , $Sign_i$ (Message1, N_r, $MAC_{SK}(ID_i)$), SA_{i2}, TS_i, TS_r, $MAC_{SK}(…))$
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$, $Sign_R$ (Message2, N_i, $MAC_{SK}(ID_r)$), SA_{r2}, TS_i, TS_r, $MAC_{SK}(…))$

SPI_x = two values that together identify the protocol run and the created IKE SA

SA_{x1} = offered and chosen algorithms, DH or ECDH group

SK = h(Ni, Nr, g^{xy}) — actually, many different keys are derived from this

$Sign_x$ ($Message_x$, N_y, $MAC_{SK}(ID_x)$) – SIGMA authentication

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted root CAs

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

$E_{SK}(…, MAC_{SK}(…))$ = Authenticated encryption for identity protection

IKEv2 notation in RFC 7296

Initial exchanges in the notation of the standard:

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ]
3. I → R: HDR(A,B), SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTH, SAr2, TSi, TSr }

(1–2) IKE_SA_INIT exchange

(3–4) IKE_AUTH exchange

SPI_x = two values that together identify the protocol run and the created IKE SA

Nx = nonces

SAx1 = offered and chosen algorithms, DH or ECDH group

KEx = Diffie-Hellman or ECDH key shares

IDx, CERT, CERTREQ = accepted root CAs, identity, certificate

AUTH = SIGMA authentication (signature and MAC)

SK = key material for deriving shared keys

SK { … } = authenticated encryption for identity protection

SAx2, TSx = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

Notation	Payload
-----------	----------------------
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
CP	Configuration
D	Delete
EAP	Extensible Authentication
HDR	IKE header (not a payload)
IDi	Identification - Initiator
IDr	Identification - Responder
KE	Key Exchange
Ni, Nr	Nonce
N	Notify
SA	Security Association
SK	Encrypted and Authenticated
TSi	Traffic Selector - Initiator
TSr	Traffic Selector - Responder
V	Vendor ID

IKEv2 with pre-shared key

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr
3. I → R: HDR(A,B), SK { IDi, [IDr,] AUTH, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, AUTH, SAr2, TSi, TSr }

- Authentication with a pre-shared key between initiator and responder: AUTH is a MAC instead of a signature
  - Receiver selects the shared key based on the sender identity IDx
  - Only strong keys, no passphrases

IKEv2 with EAP

- IKEv2 supports EAP authentication

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr
3. I → R: HDR(A,B), SK { IDi, [IDr,] [CERTREQ,] SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTH, EAP }
5. I → R: HDR(A,B), SK { EAP }
6. R → I: HDR(A,B), SK { EAP(success) } // or send more EAP requests
7. I → R: HDR(A,B), SK { AUTH, }
8. R → I: HDR(A,B), SK { AUTH, SAr2, TSi, TSr }

- EAP is a framework with many authentication methods, e.g., password and SIM
- EAP for only the initiator [RFC 7296] or mutual authentication [RFC 5998]
- AUTH in messages 7-8 contains a MAC computed with the EAP MSK

Master Session Key (MSK)

IKEv2 discussion

Internet Key Exchange (IKEv2)

1. I → R: SPI_i, SPI_r, SA_{i1}, g^x, N_i
2. R → I: SPI_i, SPI_r, SA_{r1}, g^y, N_r, $CERTREQ_r$
3. I → R: SPI_i, SPI_r, $E_{SK}(ID_i$, $CERT_i$, $CERTREQ_i$, ID_r ,
   $Sign_i$ (Message1, N_r, $MAC_{SK}(ID_i)$), SA_{i2}, TS_i, TS_r, $MAC_{SK}(\ldots)$)
4. R → I: SPI_i, SPI_r, $E_{SK}(ID_r$, $CERT_r$,
   $Sign_R$ ((Message2, N_i, $MAC_{SK}(ID_r)$), SA_{r2}, TS_i, TS_r, $MAC_{SK}(\ldots)$)

SPI_x = two values that together identify the protocol ru

SA_{x1} = offered and chosen algorithms, DH and ECDH gr

$SK = h(Ni, Nr, g^{xy})$ — actually, 7 different keys are deriv

ID_x, $CERT_x$, $CERTREQ_x$ = identity, certificate, accepted ro

SA_{x2}, TS_x = parameters for the first IPsec SA (algorithms

$E_{SK}(\ldots, MAC_{SK}(\ldots))$ = HMAC and encryption, or authenti

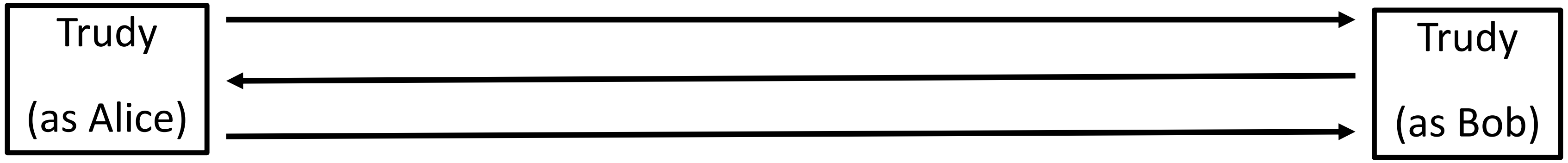
Which security properties?

- Secret, fresh session key
- Mutual or one-way authentication
- Entity authentication, key confirmation
- Perfect forward secrecy (PFS)
- Contributory key exchange
- Downgrading protection
- Identity protection
- Non-repudiation
- Plausible deniability
- DoS resistance

Privacy properties

- Identity protection
  - All identifiers and certificates are encrypted with the DH secret
  - Initiator reveals its identity first → vulnerable to active attacks
  - Responder authenticates initiator before revealing its identity → Responder identity protected also against impersonation attacks.
  - Why protect the responder better? Because the attacker can initiate IKEv2 key exchange with any target IP address. The target then becomes the responder
  - Special case: In mutual authentication with EAP, identity protection against active attackers depends on the EAP method
- Plausible deniability (سياسة الإنكار)
  - Neither endpoint signs anything that would bind it to the other endpoint's identity (Initiator and Responder can deny that any conversation took place)

Plausible deniability



- Trudy can create fake "conversation" that appears to be between Alice and Bob
  - Appears valid, even to Alice and Bob!
- It is a ***feature…***
  - **Plausible deniability:** Alice and Bob can deny that any conversation took place!
- In some cases it might create a problem
  - E.g., if Alice makes a purchase from Bob, she could later repudiate it (unless she had signed)

IKEv2 with a cookie exchange

- Responder may send a cookie (a random number) to the initiator
- Goal: verify initiator IP address; prevent DoS attacks from a spoofed IP address

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,0), N(COOKIE) // R stores no state
3. I → R: HDR(A,0), N(COOKIE), SAi1, KEi, Ni
4. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ] // R creates a state
5. I → R: HDR(A,B), SK{ IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr }
6. R → I: HDR(A,B), E_{SK} (IDr, [CERT,] AUTH, SAr2, TSi, TSr)

How to bake a good cookie? Example:

$$\text{COOKIE} = h(K_{R\text{-periodic}}, \text{ipaddr}_I, \text{ipaddr}_R)$$

where $K_{R\text{-periodic}}$ is a periodically changing secret key know only by the responder R

Negotiated parameters

Many options add complexity and reduce inter-operability

- NAT traversal:
  - NAT detection `IKE_SA_INIT` exchange
  - If NAT detected, IKEv2 and IPsec are encapsulated in UDP with port 4500

- Parameters for the key exchange:
  - Protocol version and authentication method (signatures, PSK, or EAP)
  - A, B = each endpoint chooses a locally unique SPI for the `IKE SA`
  - SAi1, SAr1 = cryptographic algorithms for the key exchange and `IKE SA` (responder chooses from initiator's offer)
  - CERTREQ = sender's supported trust anchors (CAs)
  - IDr = responder identity which the initiator wants to authenticate
- Parameters for the `IPsec SA` pair:
  - SAi2, SAr2 = cryptographic algorithms for protecting session data SA (responder chooses from initiator's offer)
  - TSi, TSr = traffic selectors i.e. which packets to protect (responder can choose a subset of the offer)

IKE versions

Internet Security Association and Key Management Protocol (ISAKMP)
Aggressive mode without identity protection

- IKE(v1) [RFC 2407, 2408, 2409]
  - Framework for authenticated key-exchange protocols, typically DH
  - Multiple authentication methods: certificates, pre-shared key, Kerberos
  - Two phases: Main Mode (MM) or Aggressive Mode creates an ISAKMP SA (i.e., IKE SA) and Quick Mode (QM) creates IPsec SAs
  - Interoperability issues, complex to implement and test, incomplete spec
  - Still used, but no reason to use for anything new
- IKEv2 [RFC 7296]
  - Redesign of IKE: fewer modes and messages, simpler to implement
  - Interoperability still requires careful configuration of the endpoints

IPsec session protocol

Session protocol

- Encapsulated Security Payload (ESP) [RFC 4303]
  - Encryption and MAC for each IP packet
  - Optional replay protection with sequence numbers

Features to avoid:

- ESP with encryption only is insecure but allowed by some IPsec APIs
- Authentication Header (AH) – authentication only, no encryption
  - Do not use for new applications
  - Exists because of US export controls in the 1990s

Session protocol modes

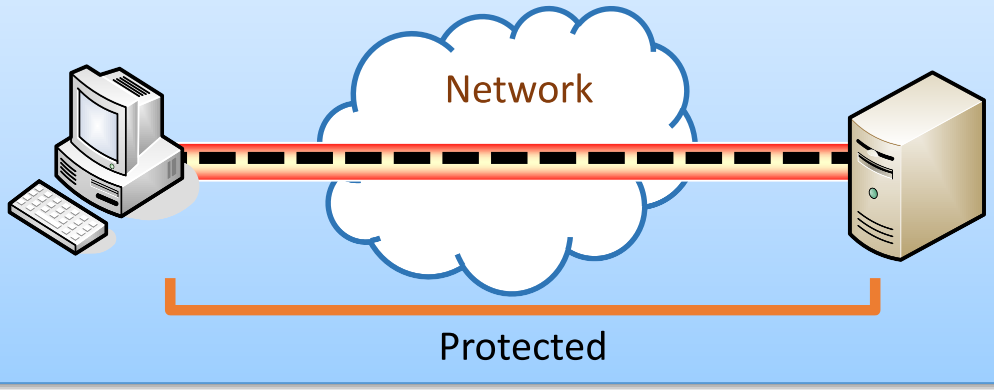
- Transport mode:
  – Host-to-host security
  – ESP header added between the original IP header and payload
- Tunnel mode:
  – Typically used for tunnels between security gateways to create a VPN
  – Entire original IP packet encapsulated in a new IP header plus ESP header
- In practice, IPsec is mainly used in tunnel mode

Transport and tunnel mode

Could be used for end-to-end protection of intranet traffic

Transport mode

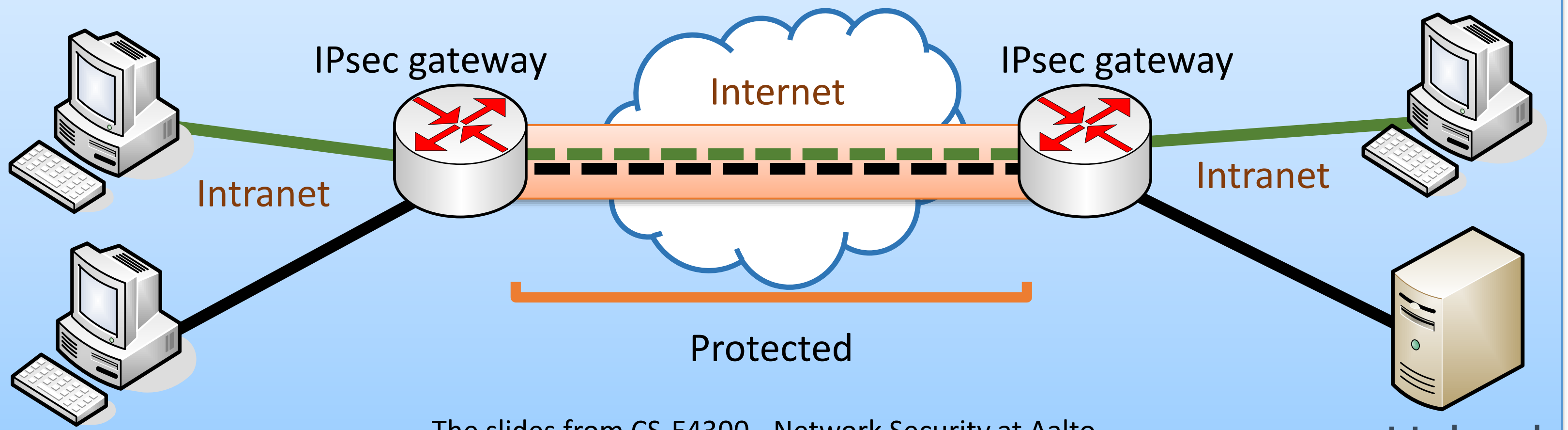
Encryption and/or authentication from end host to end host



! Typical for site-to-site VPN

Tunnel mode

Encryption and/or authentication between two gateways



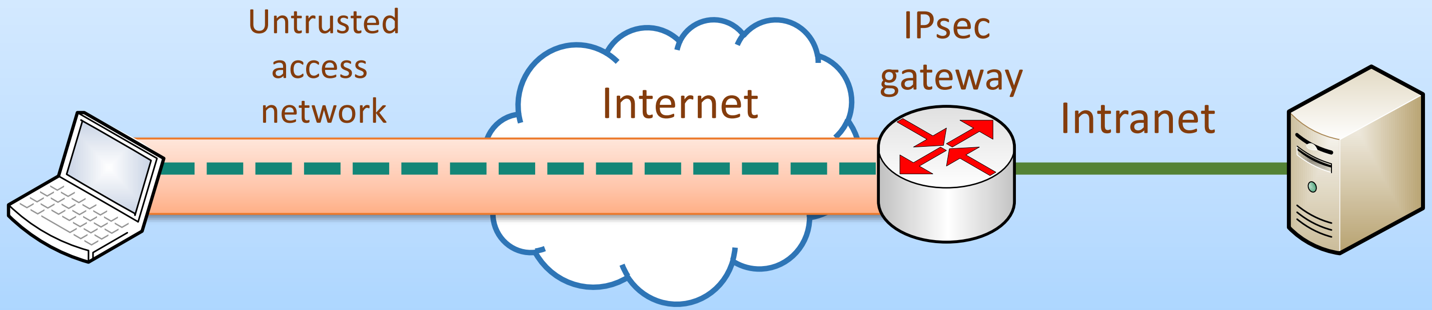
Gateway routers establish the IPsec tunnel; routing rules send traffic through the tunnel

Host-to-gateway VPN

Mobile-user VPN back to office

without NAT

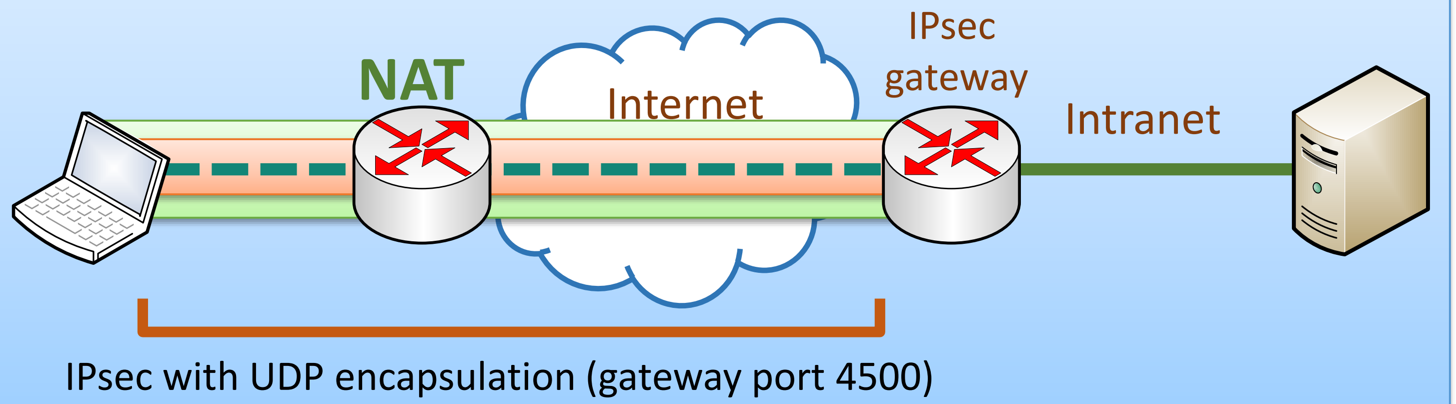
Tunnel mode between a host and a gateway



Host gets an IP address from the gateway router and becomes part of the intranet

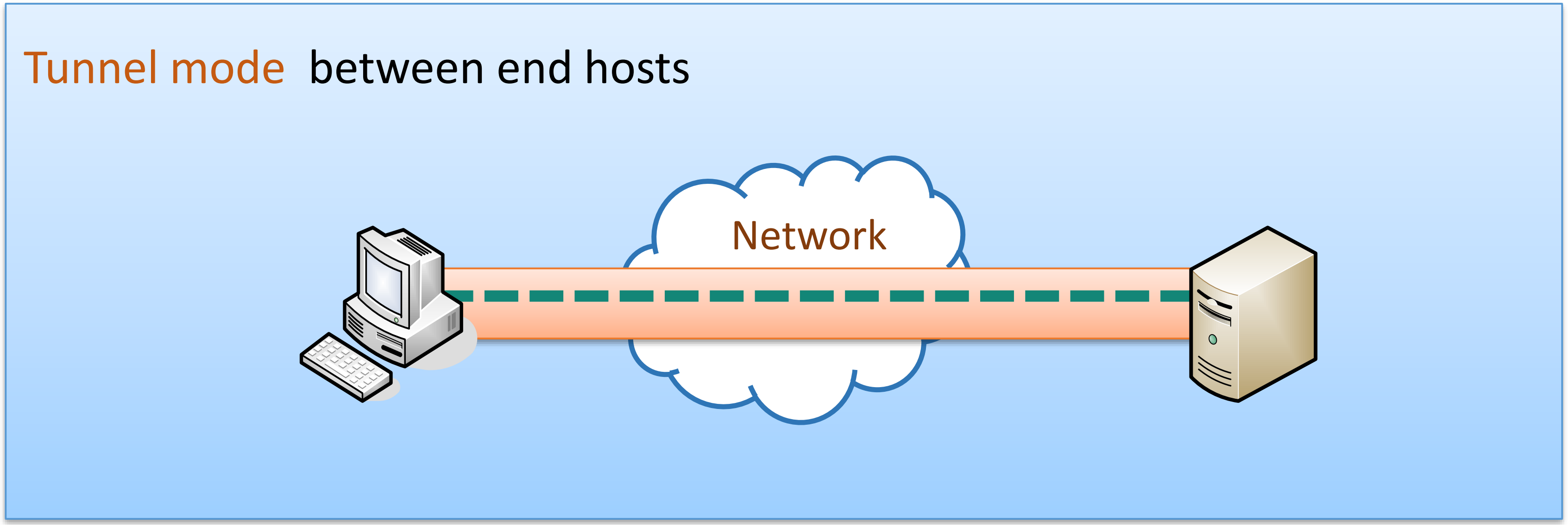
! Typical scenario with NAT

Tunnel mode between a host and a gateway with NAT traversal



Tunnel mode between hosts

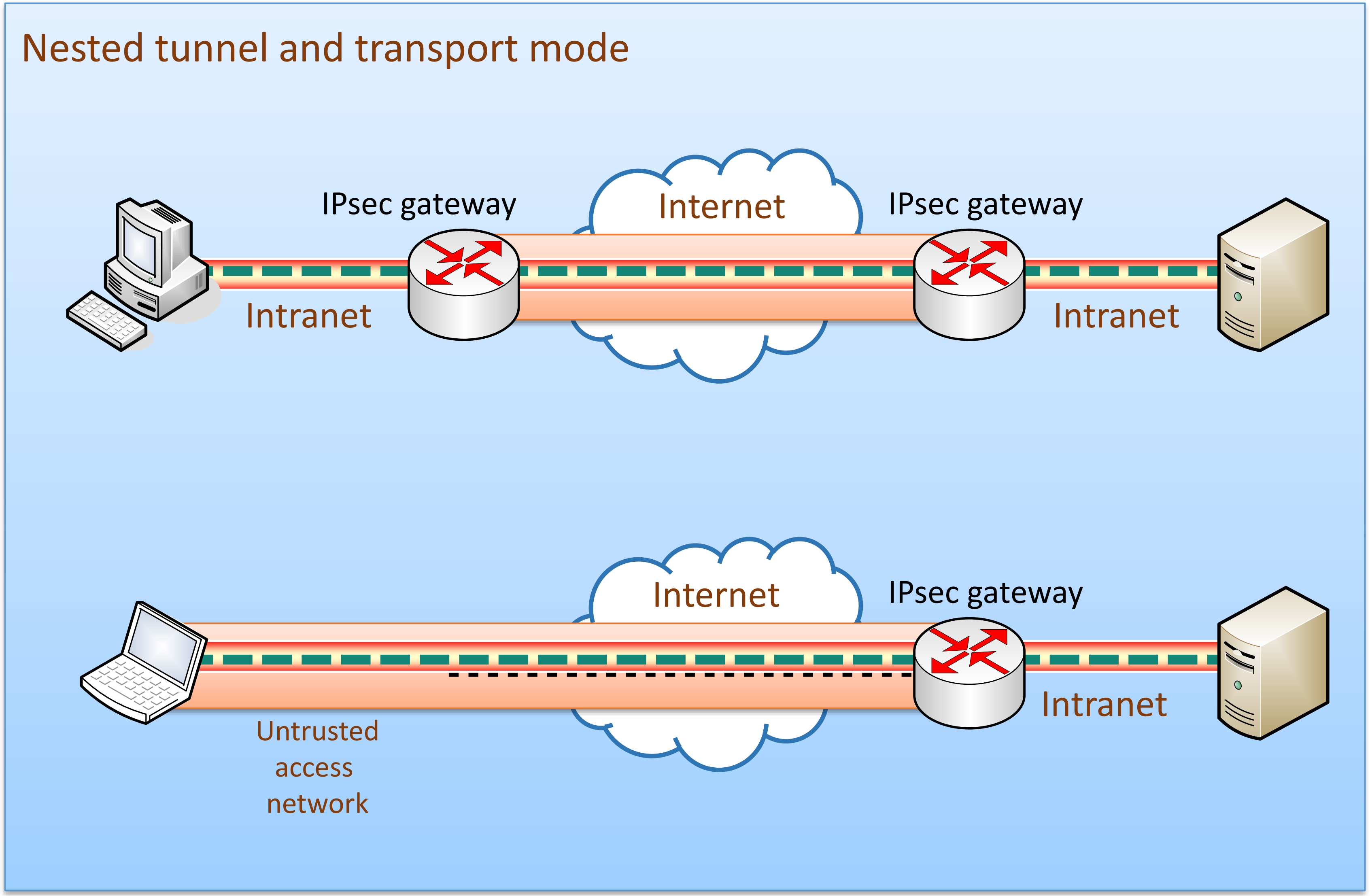
Security equivalent to transport mode



Nested protection

Combined VPN and end-to-end protection

– less common but possible

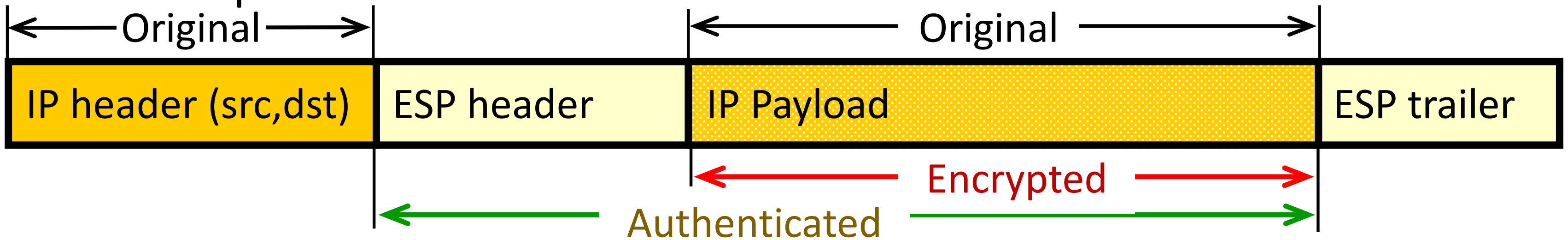


ESP packet formats

Original IP packet:

ESP header = SPI + sequence number
ESP trailer = padding + integrity check (HMAC)

ESP in transport mode:



RFC4303

ESP in tunnel mode:

ESP in tunnel mode with NAT traversal:

ESP tunnel headers

- Tunnel-mode ESP packet:

IP header(src gateway, dst gateway) |
UDP(gateway port 4500) |
ESP header(spi, sqn) |
IP header(src host, dst host) |
payload |
ESP trailer(padding, integrity check)

Outer IP header with gateway IP addresses

UDP header for NAT traversal

Security association identifier SPI, and optional sequence number for replay protection

Inner IP header with end-host IP addresses (=original IP header)

Original TCP/UDP/SCTP/ICMP

HMAC

Host-to-gateway VPN and IP addresses

- Tunnel-mode ESP packet:

IP header(src gateway, dst gateway) |
UDP(gateway port 4500) |
ESP header(spi, sqn) |
IP header(src host, dst host) |
payload |
ESP trailer(padding, integrity check)S

Outer IP header:
- Host’s current IP address and the gateway IP addresses
- With NAT, the host’s IP address changes on the way, and the UDP header is included

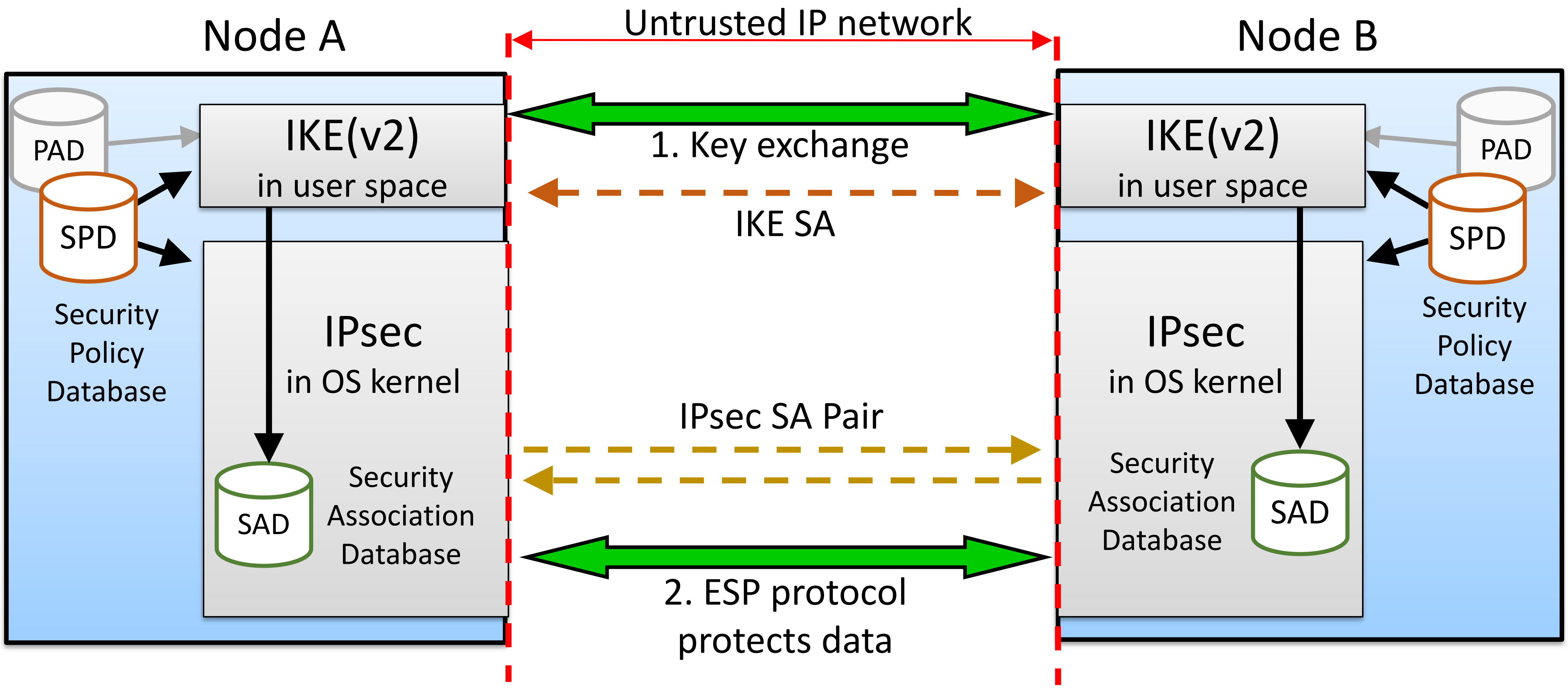
Inner IP header:
- Host’s intranet address as the source or destination
- Intranet server IP address as the other endpoint

IPsec architecture

Internet protocol security (IPsec)

- Network-layer security protocol
  - Protects IP packets between two hosts or gateways
  - Transparent to transport layer and applications: security policy is defined and enforced on network level
  - IP addresses are used as host identifiers
- Two steps:
  1. IKE authenticated key exchange creates security associations
  2. ESP session protocol protects data
- Specified by Internet Engineering Task Force (IETF)
  - Initially designed as for IPv6
    - Original goal: encryption and authentication layer that will replace all others and meet all Internet security needs
    - Security (IPsec) was a sales point for IPv6, but IPsec now works also for IPv4

IPsec architecture [RFC 4301]



- Security associations (SA) in SAD created by IKE, used by IPsec ESP
- Security policy in SPD guides SA creation and use

Security Associations (SA)

- One IKE SA for each pair of nodes
  - Stores a master session key for creating IPsec SAs
- At least one IPsec SA pair for each pair of nodes
  - Stores negotiated algorithms, keys, and algorithm state
  - IPsec SAs always come in pairs, one in each direction
- IPsec SAs identified by a 32-bit security parameter index (SPI)
  - The destination node selects the SPI value
- Node stores IPsec SAs in its security association database (SAD)

IPsec databases

- Security association database (SAD)
  - Contains the IPsec SAs i.e. the dynamic protection state
- Security policy database (SPD)
  - Contains the static security policy
  - Set by system admin (e.g. Windows group policy) or VPN application
- Peer authorization database (PAD)
  - Mapping between authenticated names and IP addresses
  - Conceptual; not implemented as an actual database
- The IKE service/daemon stores IKE SAs
  - Master secret for creating IPsec SAs; hash of DH secret and nonces

Note: our description of SPD differs from RFC 4301 but is closer to most implementations.

Gateway SPD/SAD example

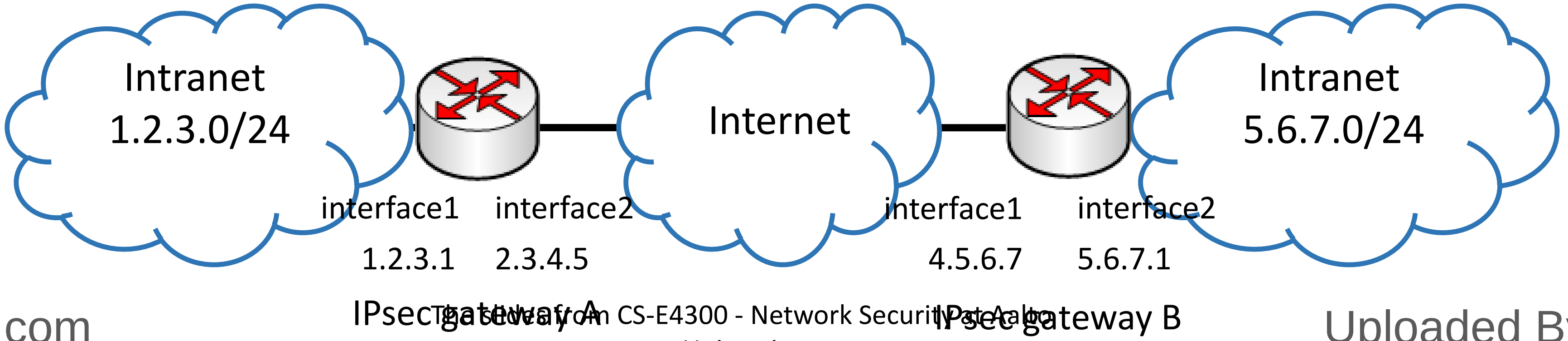
SPD of gateway A, interface 2

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
*	*	*	*	*	BYPASS	All other peers

Pointers to created associations

SAD of gateway A

SPI	SPD selector values	Protocol	Algorithms, keys, algorithm state
spi1	TCP, 1.2.3.0/24, 5.6.7.0/24	ESP tunnel from 4.5.6.7	…
spi2	—	ESP tunnel to 4.5.6.7	…



Host SPD example

Host-to-host IPsec is problematic, as explained in a later lecture

SPD for host 1.2.3.101 in intranet 1.2.3.0/24,
connecting to server 1.2.4.10 in network 1.2.4.0/24 (DMZ) and to the Internet

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT: ESP in transport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP in transport-mode	Encrypt to server in DMZ
TCP	1.2.3.101	≥1024	1.2.4.10	443	BYPASS	Allow TLS to server in DMZ
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

- Note that the other endpoint (other intranet hosts and 1.2.4.10) must have an IPsec policy that specifies the same protection for the same packets
- What is the danger in bypassing TLS traffic (line 5) and ICMP (line 2)?
- What if the attacker can poison DNS?

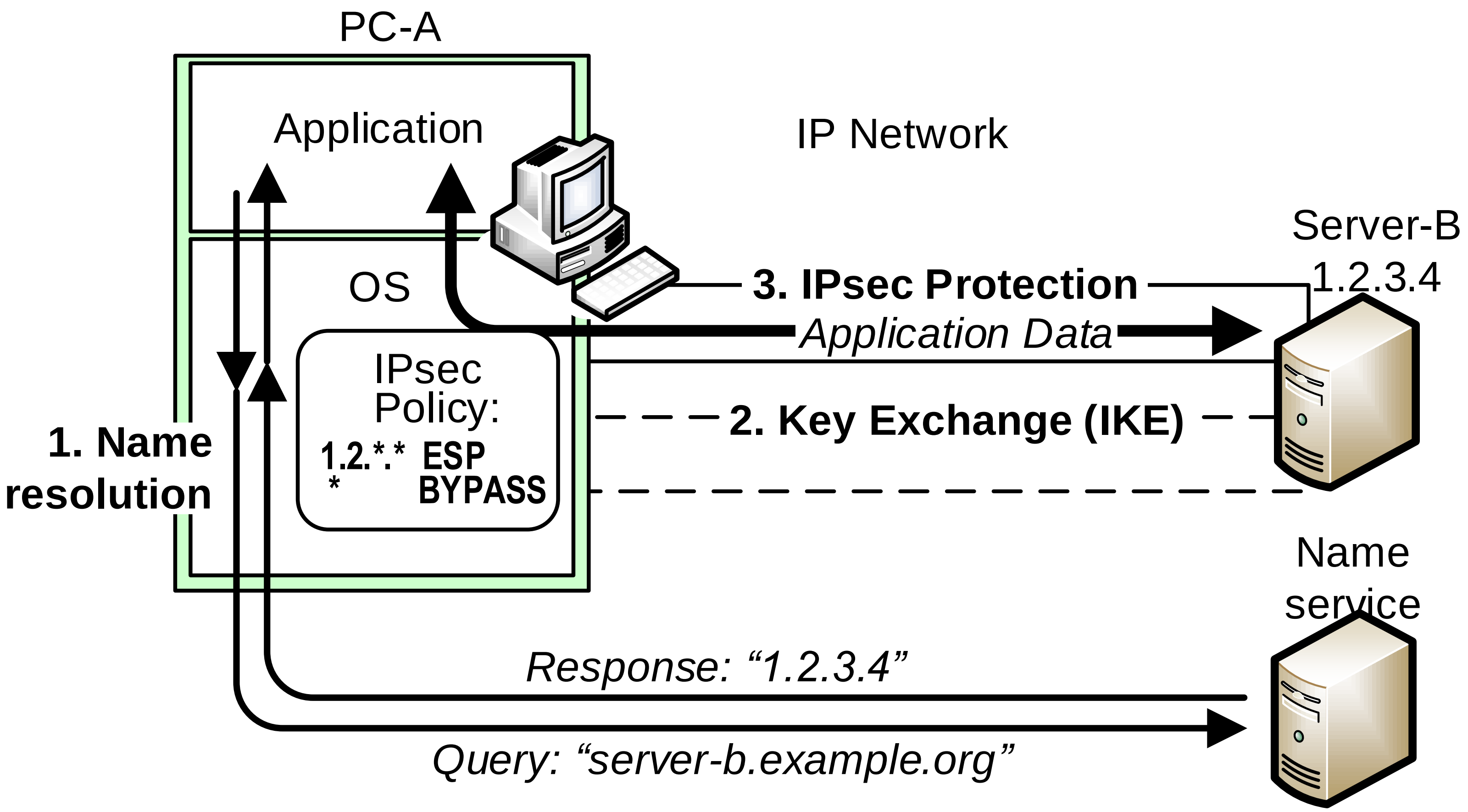
IPsec policy implementation differences

- Firewall and IPsec policies can be unified into one policy:
  – Which incoming/outgoing packets to drop or log, and which require authentication and encryption?
- IPsec policy may be specified in terms of
  – local and remote addresses
  – left and right, so that the same policy file works at both ends, or
  – source and destination addressed, with mirror flag

Mirror	Protocol	Source IP	Port	Destination IP	Port	Action	Comment
yes	UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
yes	*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
yes	*	*	*	*	*	BYPASS	All other peers

Issues with host-to-host IPsec

IPsec and name resolution



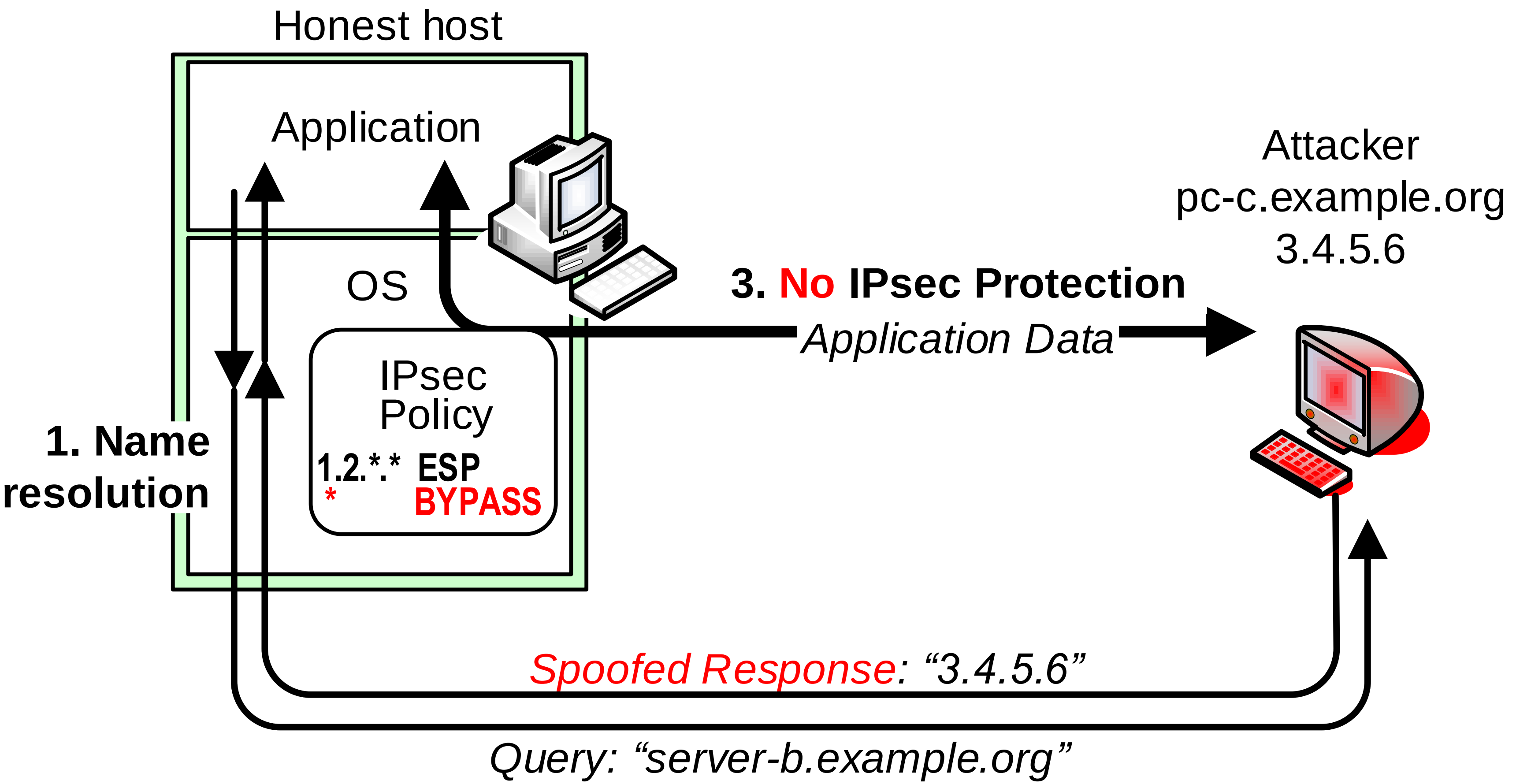
- TCP socket API: resolve name into an IP address; then connect to it
- TCP SYN to the address triggers IKEv2 (if the ESP SA does not yet exist)

IPsec and identifiers

1. Application opens a connection to an IP address. IPsec uses the IP addresses as policy selector
2. Application actually wants to connect to a specific name, and IKE usually authenticates the remote node by its DNS name

- Problem: No secure mapping between the two identifier spaces: DNS names and IP addresses

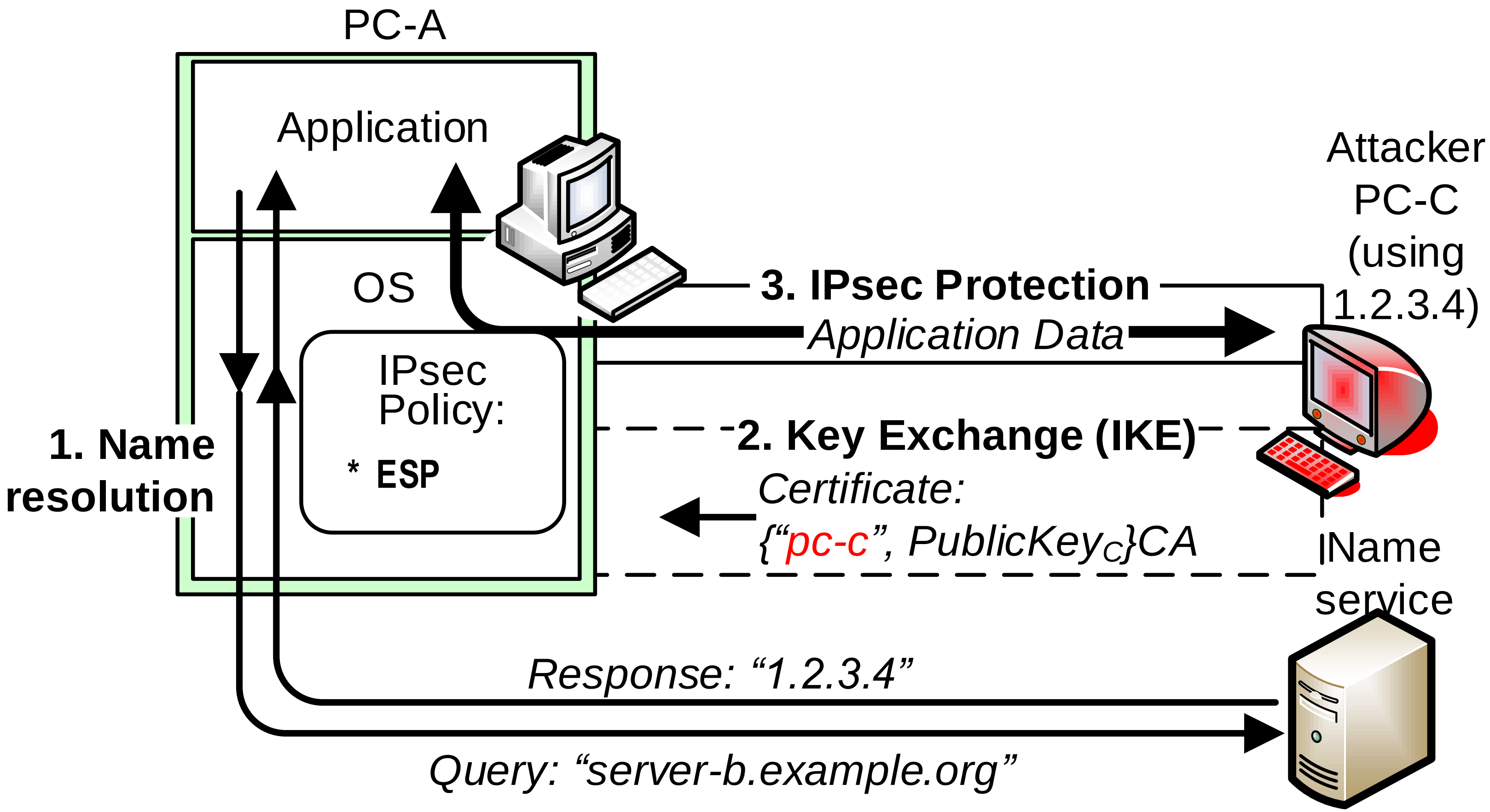
Classic IPsec/DNS Vulnerability



- Attacker spoofs DNS response to circumvent the IPsec policy

Let's assume secure DNS. Does it solve the problems?

Further problem: IPsec and Certificates



- IKE knows the peer's IP address, not its name. The certificate only contains the name. How does IKE know if the certificate is ok? No obvious solution.

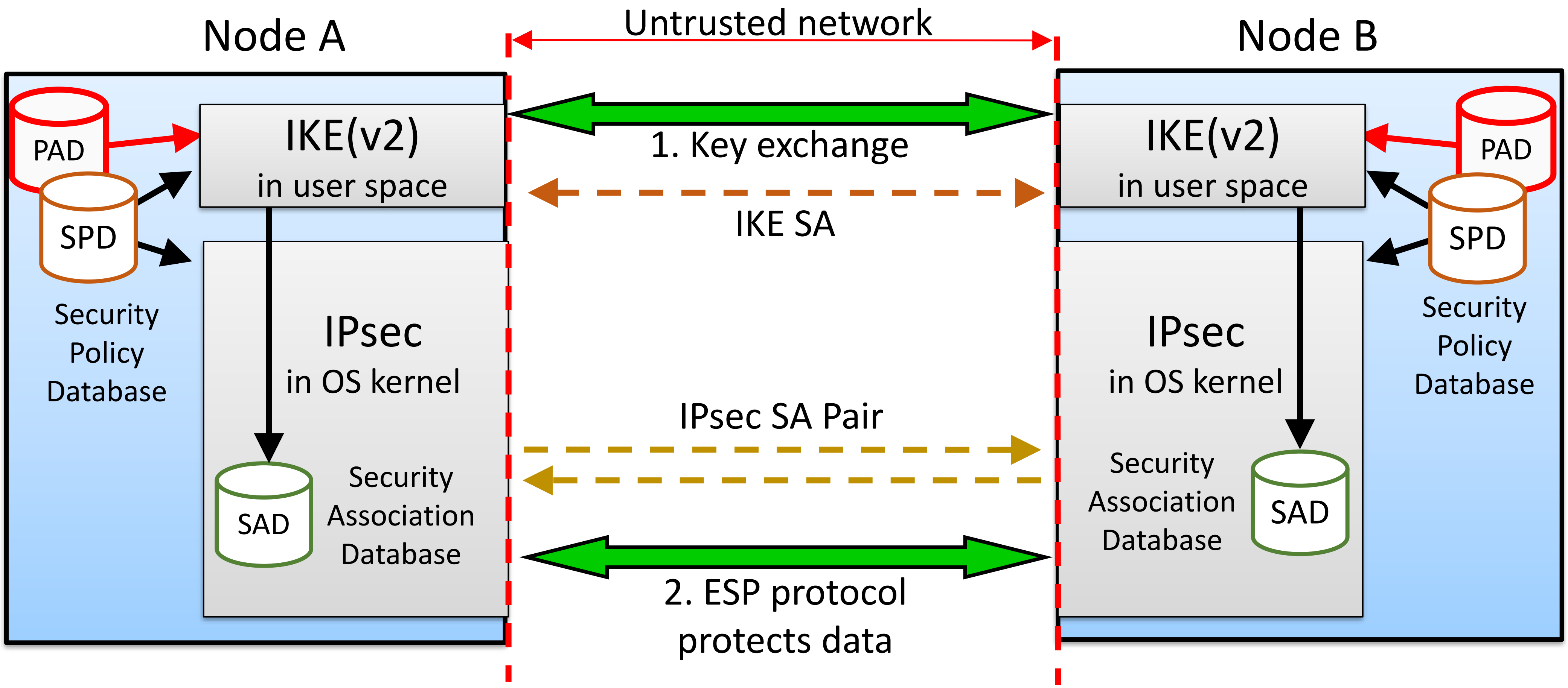
IPsec and Certificates – solutions?

- Secure DNS (forward lookup) does not help — why?
- Secure reverse DNS would be a solution — but it does not exist

Other solutions:

- Connect by name — change the socket API so that the connect() call specifies the host name, not the IP address
- Give up IPsec transparency: applications query the socket API for the authenticated name
  - VPN applications do this to check the VPN gateway name from the certificate
- Ignore the hostname: use IPsec only to isolate certified intranet hosts from outsiders/intruders
  - Example: NAP in a Windows domain uses IPsec for network access control and not for end-to-end authentication of the individual host identities

IPsec architecture [RFC 4301]



Peer authorization database (PAD)

- IPsec specification [RFC4301] defines a database that maps authenticated names to allowed IP addresses
- How is PAD implemented?
  - VPN applications check that the name on the certificate matches a known VPN gateway
  - For host-to-host IPsec in a closed domain, such as intranet, PAD could theoretically be implemented – but it has not been
  - No solution for general host-to-host IPsec in the open Internet

This is why IPsec is really only used for VPN and *not* host-to-host