Network Security (NetSec)

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer 2015
Chapter 05: Network Level Security
Module 03: IPsec – Implementation Issues

Prof. Dr.-Ing. Matthias Hollick

Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO
Department of Computer Science
Center for Advanced Security Research Darmstadt - CASED

Mornewegstr. 32
D-64293 Darmstadt, Germany
Tel.+49 6151 16-70922, Fax. +49 6151 16-70921
http://seemoo.de or http://www.seemoo.tu-darmstadt.de

Prof. Dr.-Ing. Matthias Hollick
matthias.hollick@seemoo.tu-darmstadt.de

CASED

Learning Objectives

How can network level security be implemented in a comprehensive fashion

- Understand IPsec in operation (how does it play together with operating systems)
- Explain the individual entities necessary to manage network level security

SECURE MOBILE NETWORKING

IPsec

processing and implemenation

IPsec Processing Model

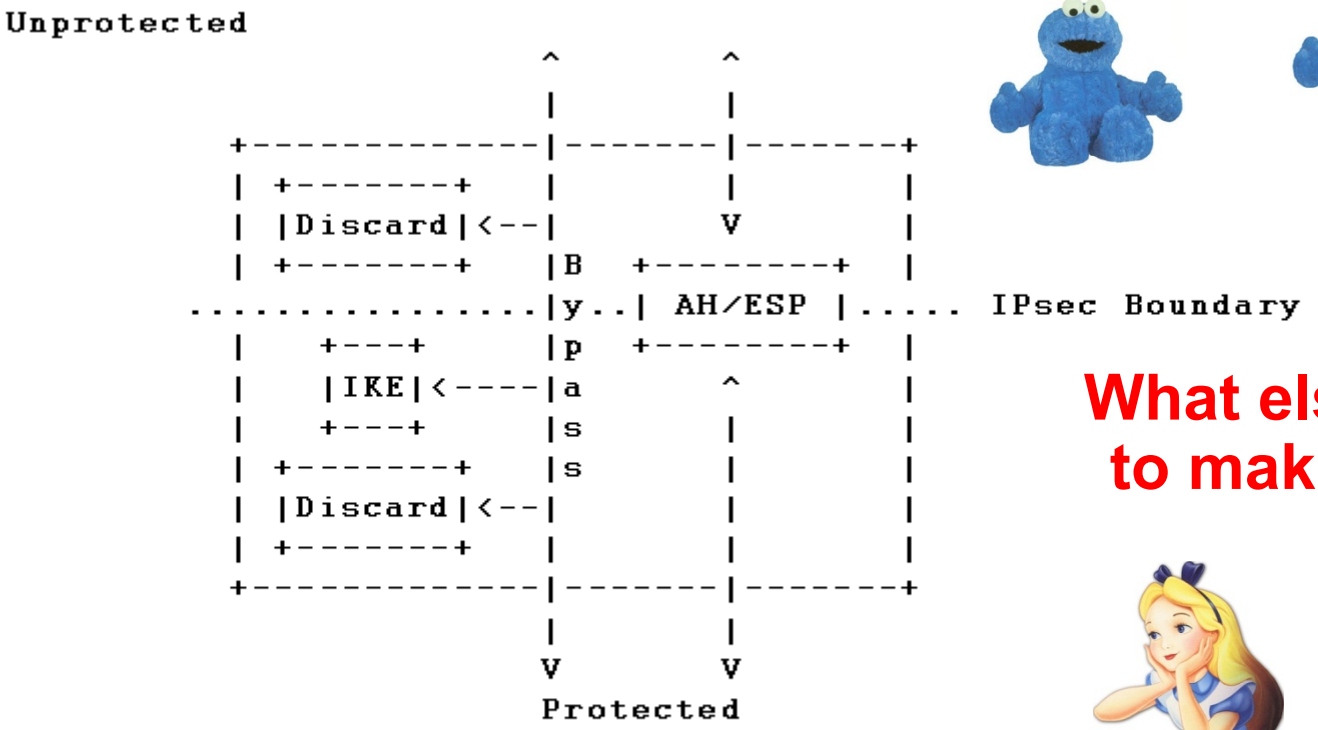
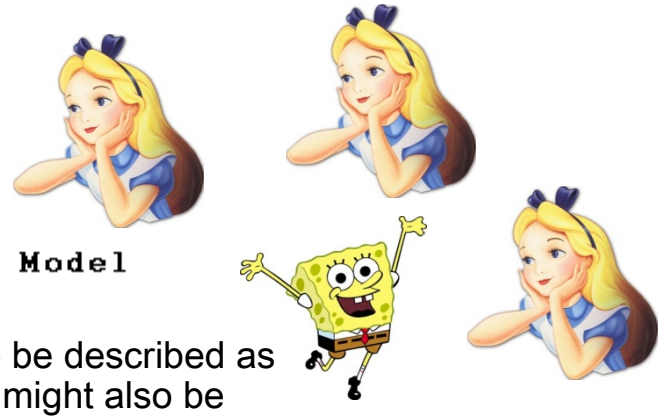


Figure 1. Top Level IPsec Processing Model

What else do we need to make IPsec work?



In this diagram, "unprotected" refers to an interface that might also be described as "black" or "ciphertext". Here, "protected" refers to an interface that might also be described as "red" or "plaintext".

Source (and more details): RFC 4301

Overview of this Module

(1) Some Implementation consideration

(2) IPsec Databases

- Security Association (SA) and processing model
- Security Association Database (SAD)
- Security Policy Database (SPD)
- Peer Authorization Database (PAD)

(3) IPsec Key Management

- SA and Key Management using
  The Internet Key Exchange (IKE)

Chapter 05, Module 03

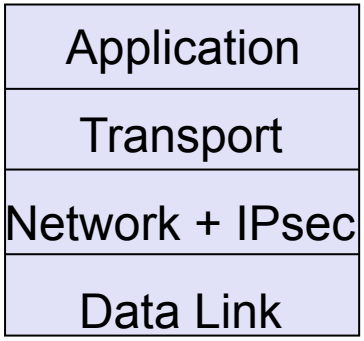
IPsec Implementation Alternatives: Host Implementation

Advantages of IPsec implementation in end systems:

- Provision of end-to-end security services
- Provision of security services on a per-flow basis
- Ability to implement all modes of IPsec

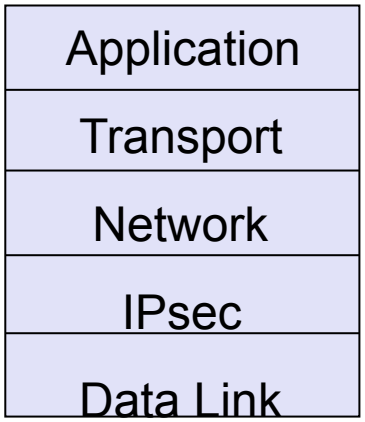
Two main integration alternatives:

OS integrated



True OS integration is the method of choice, as it avoids duplication of functionality

"Bump" in the stack (BITS)



If the OS can not be modified, IPsec is inserted above the data link driver

See Appendix for implementation issues

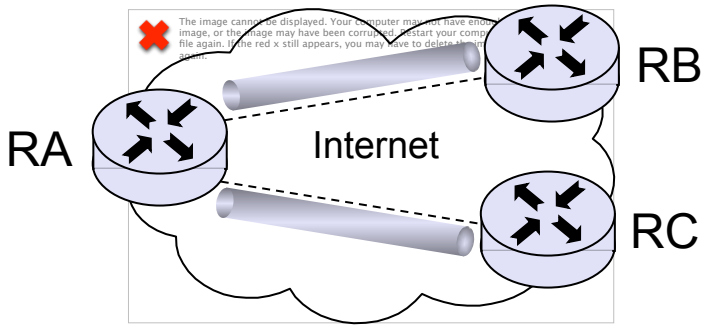
IPsec Implementation Alternatives: Router (Gateway) Implementation

Advantages of IPsec implementation in routers/gateways:

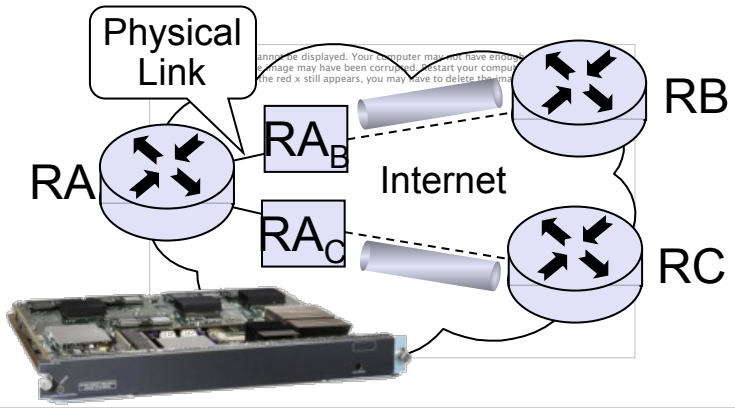
- Ability to secure IP packets flowing between two networks over a public network such as the Internet:
  - Allows to create virtual private networks (VPNs)
  - No need to integrate IPsec in every end system
- Ability to authenticate and authorize IP traffic from remote users

Two main implementation alternatives:

Router integrated



"Bump" in the wire (BITW)



Security Associations (SA)

IPsec's security associations

- Are the abstraction of an IPsec connection
- Are unidirectional
  - → Two have to be established for bidirectional communication
- Consist of (at least) a triplet
  - Security Parameter Index (SPI):
    A 32 bit value to distinguish SPI's with identical IP destination address and security protocol
  - IP destination address
  - Security Protocol: AH or ESP (but not combination in one SA)
- Define parameters and algorithms for authentication
- Define parameters and algorithms for encryption
- There exists a database of Security Associations (SAD)

Security Association (SAs)

Two kinds of SAs:

- IKE _SA: “master”
  - Long-term validity
  - Used to negotiate the CHILD_SA
  - Based on a pre-shared secret or a PKI
- CHILD_SA: “session”
  - Used for data transmission

For establishing a secure communication between two IP hosts:

- Negotiate IKE SA
- Use IKE SA to negotiate CHILD_SA
- Use CHILD_SA to encrypt the data to transmit

Security Parameters Index (SPI)

Can be up to 32 bits large

The SPI allows the destination to select the correct SA under which the received packet will be processed

- According to the agreement with the sender
- The SPI is sent with the packet by the sender

Who chooses the SPI?

Is it always unique?

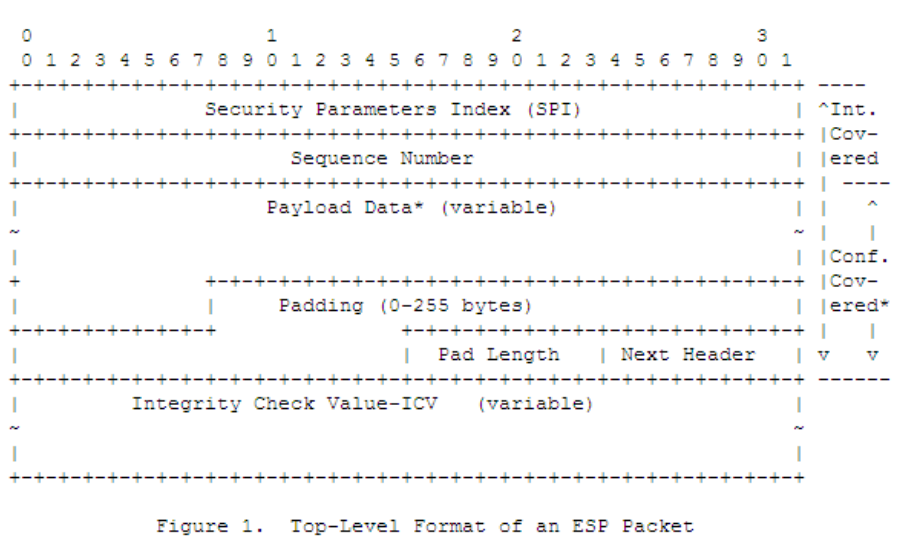


Figure 1. Top-Level Format of an ESP Packet

SPI + Dest IP address + IPsec Protocol (AH or ESP) uniquely identifies a SA

SA Database - SAD

Holds parameters for each SA

- Lifetime of this SA
- AH and ESP information
- Tunnel or transport mode

Every host or gateway participating in IPsec has their own SA database

Is searched using “longest match”

1. Search SAD for a match on the combination of SPI, destination, and source addr. If matching process according to SAD entry.
2. Search the SAD for a match on both SPI and destination address. If matching process according to SAD entry.
3. Search the SAD for a match on only SPI. Process with matching SAD entry. Otherwise, discard packet and log an auditable event.

Security Policy Database (SPD)

The SPD manages Security Associations (SA)

- What traffic to protect? Policy entries define which SA or SA bundles to use on IP traffic
- Ordered list of access control entries
- Nominally static but could be dynamic
- Per-interface, inbound & outbound databases, secure trafficdatabase
- Each SPD entry specifies:
  - DISCARD (do not let in or out)
  - BYPASS (outbound: do not apply IPsec, inbound: do not expect IPsec)
  - PROTECT (process IPsec (protocols & algorithms))
- Traffic characterized by selectors:
  - source/ destination IP addr. (also bit masks & ranges)
  - next protocol, source/ destination ports
  - user or system ID (map to other selectors in an SG)

SPD Entry Examples for a Security Gateway (SG)

SECURE MOBILE NETWORKING

SPD logically separated into three parts:

- An SPD is logically divided into three pieces. The SPD-S (secure traffic) contains entries for all traffic subject to IPsec protection. SPD-O (outbound) contains entries for all outbound traffic that is to be bypassed or discarded. SPD-I (inbound) is applied to inbound traffic that will be bypassed or discarded.
- Can be correlated or non-correlated (affects caching)

Outbound SPD entry example

- IP source= 128.89.*.*
- IP destination = 24.1.2.3
- Protocol = 6 (TCP)
- Source port = ANY, Destination port = 22 (SSH)
- Action = apply tunnel mode ESP, 3DES, HMAC- SHA- 1, instantiate per call

SPD Entry Examples for a Security Gateway (SG)

SECURE MOBILE NETWORKING

Outbound SPD entry example

- IP source= 128.89.0.0 to 128.89.0.100
- IP destination = 128.100.0.1
- Protocol = 17 (UDP)
- Source port = ANY, Destination port = 53 (DNS)
- Action = bypass

Inbound SPD entry example

- IP source= *.*.*.*
- IP destination = 128.89.1.10 (my SMTP server)
- Protocol = 6 (TCP)
- Source port = ANY
- Destination port = 25 (SMTP)
- Action = bypass

More SPD Entry Examples for a Security Gateway/Host

Outbound SPD entry example (e.g. last entry)

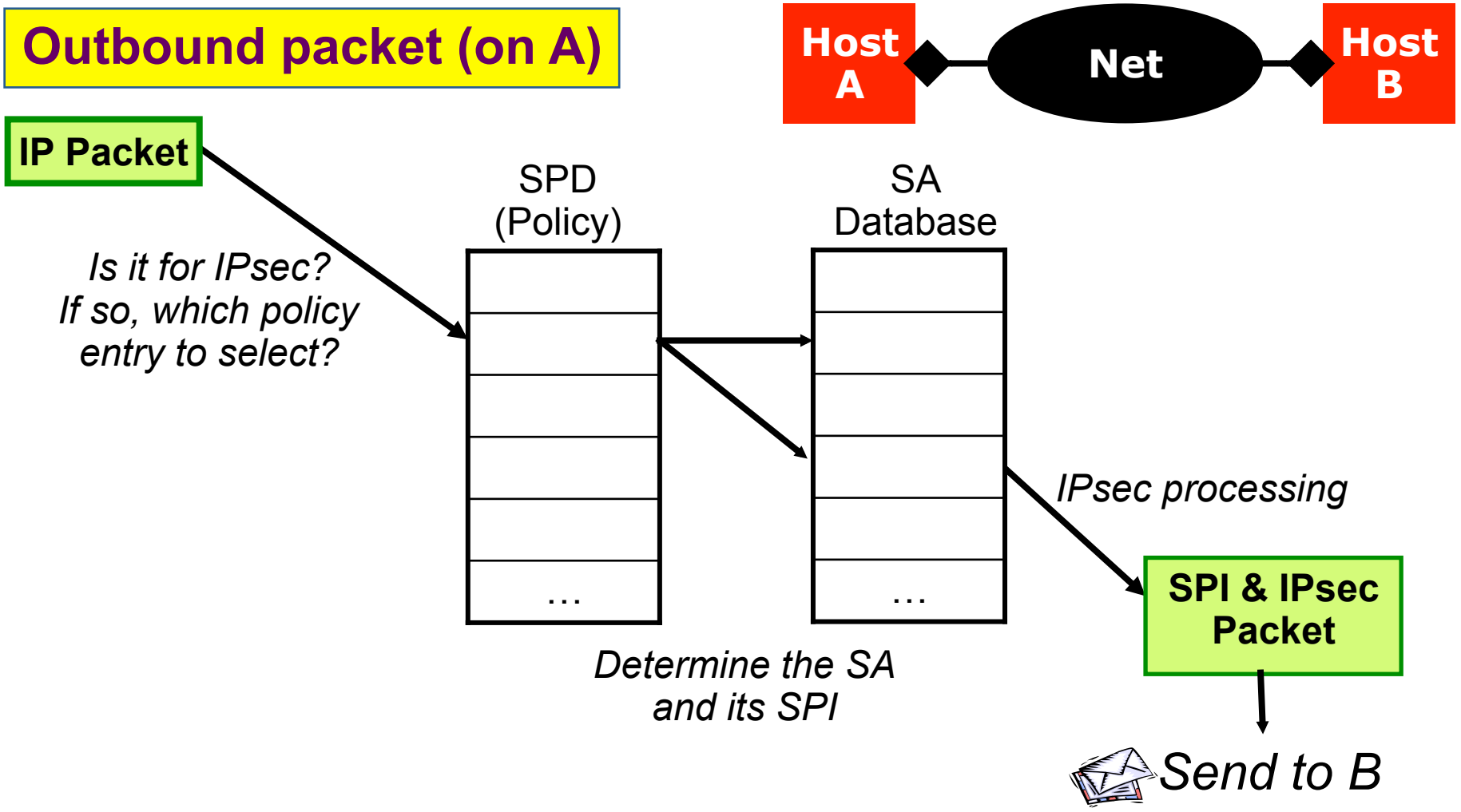
- IP source= *.*.*.* ; IP destination = *.*.*.*
- Protocol = ANY
- Source port = ANY; Destination port = ANY
- Action = DISCARD

Example for host SPD

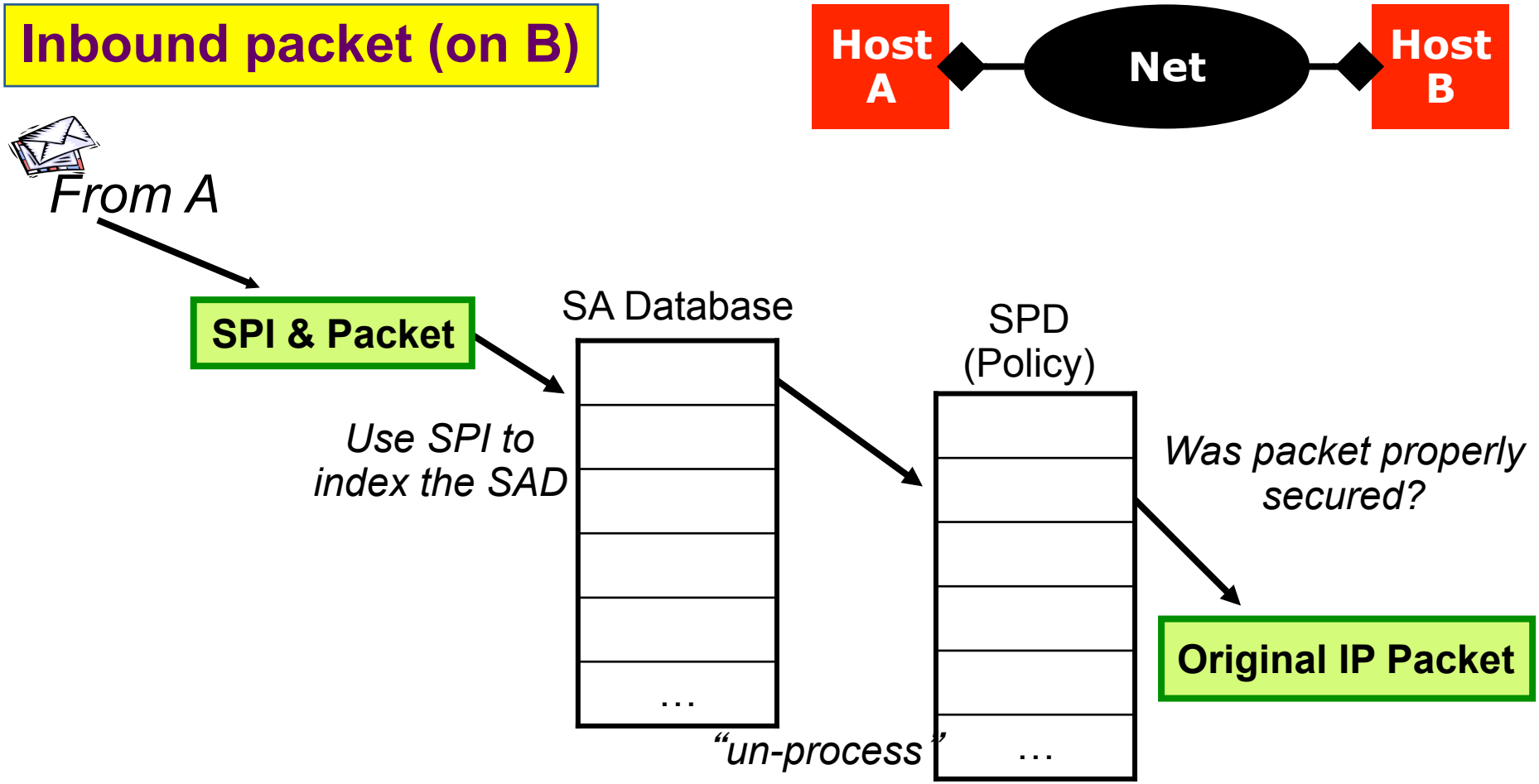
- Network 1.2.3.0/24
- DMZ 1.2.4.0/24 that is protected from both the outside world and LAN by firewalls
- Host has 1.2.3.101 and is authorized to connect to the server 1.2.4.10

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 intransport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP intransport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS: avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

Outbound Processing



Inbound Processing



Peer Authorization Database (PAD)

The PAD provides the link between the SPD and a security association management protocol such as IKE

Critical functions:

- identifies the peers or groups of peers that are authorized to communicate with this IPsec entity
- specifies the protocol and method used to authenticate each peer
- provides the authentication data for each peer
- constrains the types and values of IDs that can be asserted by a peer with regard to child SA creation, to ensure that the peer does not assert identities for lookup in the SPD that it is not authorized to represent, when child SAs are created
- peer gateway location info, e.g., IP address(es) or DNS names, MAY be included for peers that are known to be "behind" a security gateway

The case for

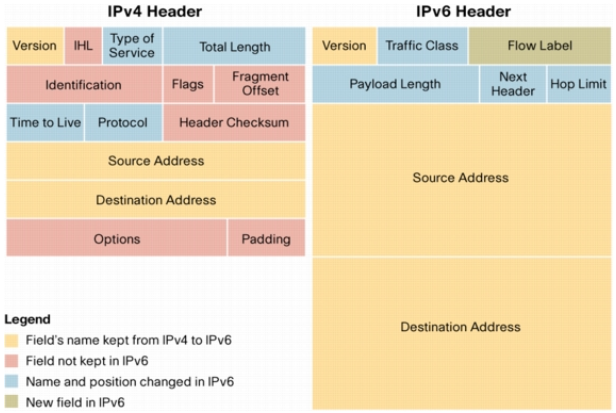
AH

courtesy Radia Perlman

AH - Peculiarities

Looks kind of like IPv6 extension header

- IPv6 has length in units of 8-octet chunks
- But AH length in units of 4-octet chunks



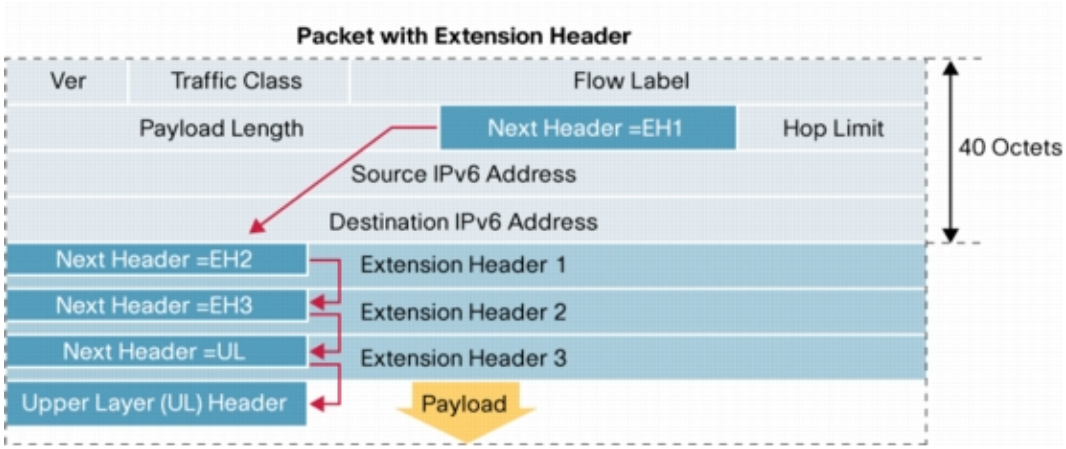
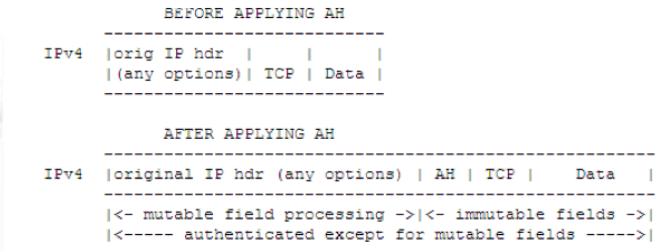


Image source: cisco whitepaper on EH processing

Because of AH, IP spec specifies, for each field, "mutable", "immutable", or "immutable but predictable", and flag options

Note: ICV before data



Why AH? History Bits

AH and ESP designed by different groups. AH designers were IPv6 supporters (and designed it alike)

AH looks more like IPv6 (but uses incompatible length field)

AH also protects "immutable" fields in IP header

Originally, ESP just encryption; encryption without integrity has flaws … Q: which ones (see Belovin paper from 1996: „Problem areas …") Then integrity protection added to ESP.

Excuses for keeping AH

- protects IP header (nobody has a credible security reason why, and ESP-tunnel can too)
- Makes NAT harder, which pleases IPv6 fans ;-)
- with AH, firewalls and routers that want to look at layer 4 info (like ports) know it's not encrypted. With ESP, can't tell from packet

Why Not AH?

IPsec way too complex already

Layer 4 info should be hidden from routers, firewalls, and can't be integrity protected en route anyway (but has end to end relevance)

If you really want to tamper with L4 header: you could peek inside ESP and almost always tell if it's encrypted or not. A flag might be nice (reserved SPIs would work)

IKE

courtesy Radia Perlman

IPsec - Key Exchange

Internet Key Exchange (IKE) protocol

- First: RFC 2409 as part of Internet Security Association and Key Management Protocol (ISAKMP) defined in RFC 2408
- Now: Combined and improved in RFC 4306 (IKEv2)
- I recommend to have a look at RFC 2009 … tell me which part you like best …

Is based on Diffie-Hellman key exchange

- Supported by “cookies” to prevent denial of service attacks

Sets up IPsec's Security Associations (SA's)

- By exchanging/establishing required parameters

Consists of two phases

- Set up secure control channel
- Set up security association

History Bits: IPSEC Key Exchange Contenders

Photuris: Signed Diffie Hellman, stateless cookies, optional hiding endpoint IDs

SKIP: Diffie-Hellman public keys, so if you know someone's public key g^B, you automatically know a shared secret g^{AB}. Each msg starts with per-msg key S encrypted with g^{AB}

And the winner was...

Internet Security Association and Key Management Protocol (ISAKMP)

- Gift to the IETF from NSA
- A "framework", not a protocol. Complex encodings. Flexible yet constraining.
- Two "phases". Phase 1 expensive, establishes a session key with which to negotiate multiple phase 2 sessions
- Current version of IKE no longer uses ISAKMP

IKEv1 – Internet Key Exchange

IKE authors tried to fit academic papers (SKEME, OAKLEY) into ISAKMP

Mostly a rewriting of ISAKMP, but not self-contained. Uses ISAKMP

Since both so badly written, hasn't gotten thorough review

- Imagine 150 pages of this!

> *While Oakley defines "modes", ISAKMP defines "phases". The relationship between the two is very straightforward and IKE presents different exchanges as modes which operate in one of two phases.* —RFC 2409

Really 3+ specs (ISAKMP, IKE, DOI)

Plus a few more (NAT traversal, etc.)

IKEv1 Key types

Step 1: Use main mode or agressive mode to establish „secure channel“

- Bidirectional SA, for further use

For each of main and aggressive, protocols are defined for each of the following key types:

- pre-shared secret key
- public signature keys
- public encryption keys (old crufty way)
- public encryption keys (new improved method)

Step 2: Use quick mode to establish SA

- Typically two unidirectional SAs

General Idea of Aggressive-Mode (IKEv1)

Alice

Bob

I'm Alice, g^A mod p, $nonce_A$ →

← I'm Bob, g^B mod p, proof I'm Bob, $nonce_B$

proof I'm Alice →

Contrast and compare with main mode on the next slide!

General Idea of Main-Mode (IKEv1)

Alice

Bob

crypto suites I support →

← crypto suites I choose

g^A mod p, $nonce_A$ →

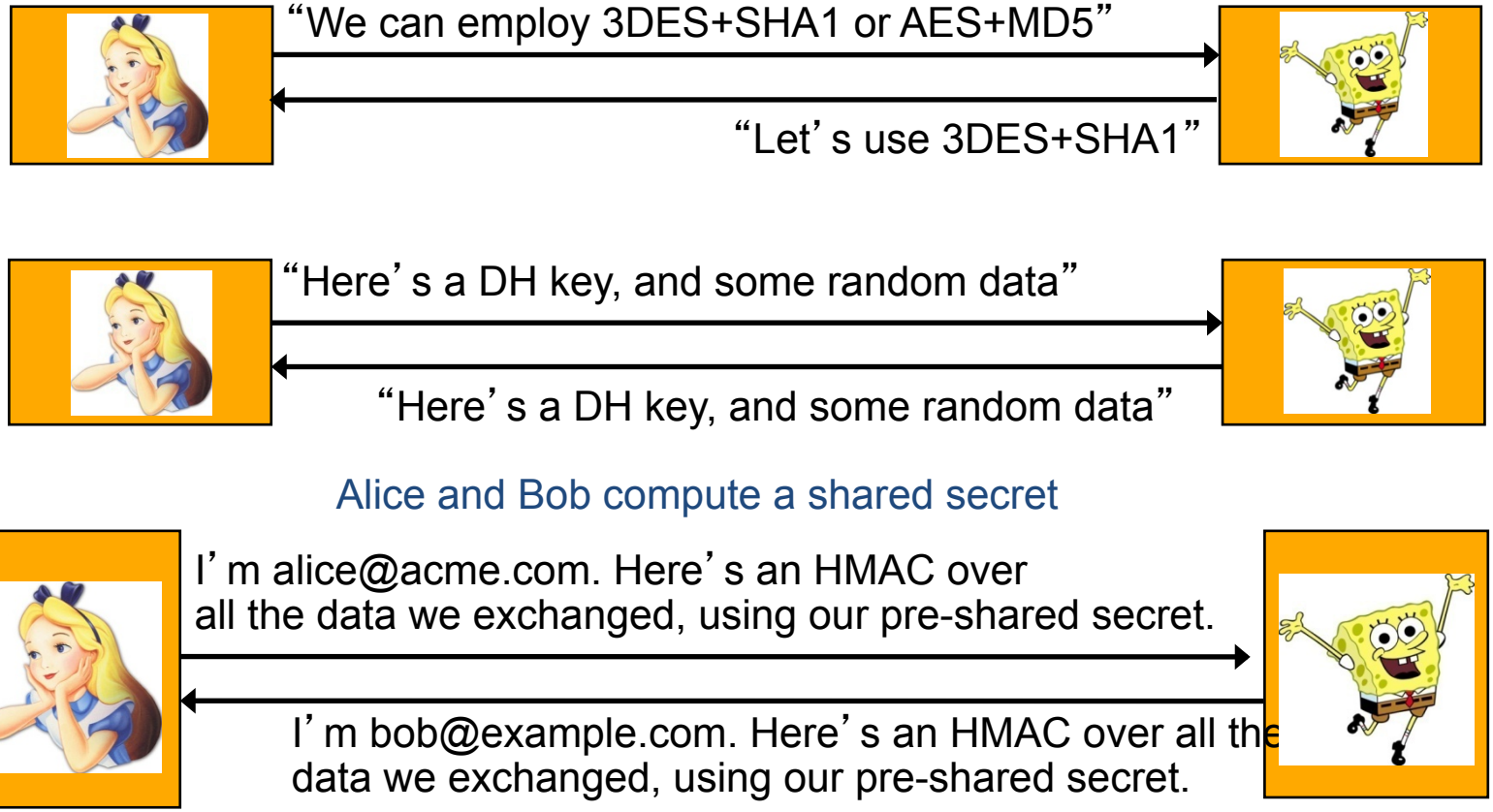
← g^B mod p, $nonce_B$

{“Alice”, proof I’m Alice} key variant-dependent →

← {“Bob”, proof I’m Bob}

Main Mode for Cartoon Lovers

Main Mode: 3 message pairs are exchanged



Main Mode vs. Aggressive Mode

The standard defined one variant as required:

main mode, pre-shared secret keys

But nobody uses it

- ID transmitted, encrypted with key which is a function of pre-shared key! Can't decrypt unless you can guess who you're talking to!
- So ID=IP Address must be static and resolvable. Useless for "road warrior"

Instead, the most commonly used scheme became aggressive mode, pre-shared secret keys …

General Idea of "Quick Mode" (Phase 2, SA for traffic)

IKE-SA, Y, traffic, SPI_A, [g^A mod p] →

← IKE-SA, Y, traffic, SPI_B, [g^B mod p]

IKE-SA, Y, ack →

Based on the IKE-SA setup in step 1, security associations to actually transfer data are established

IKEv1 vs IKEv2

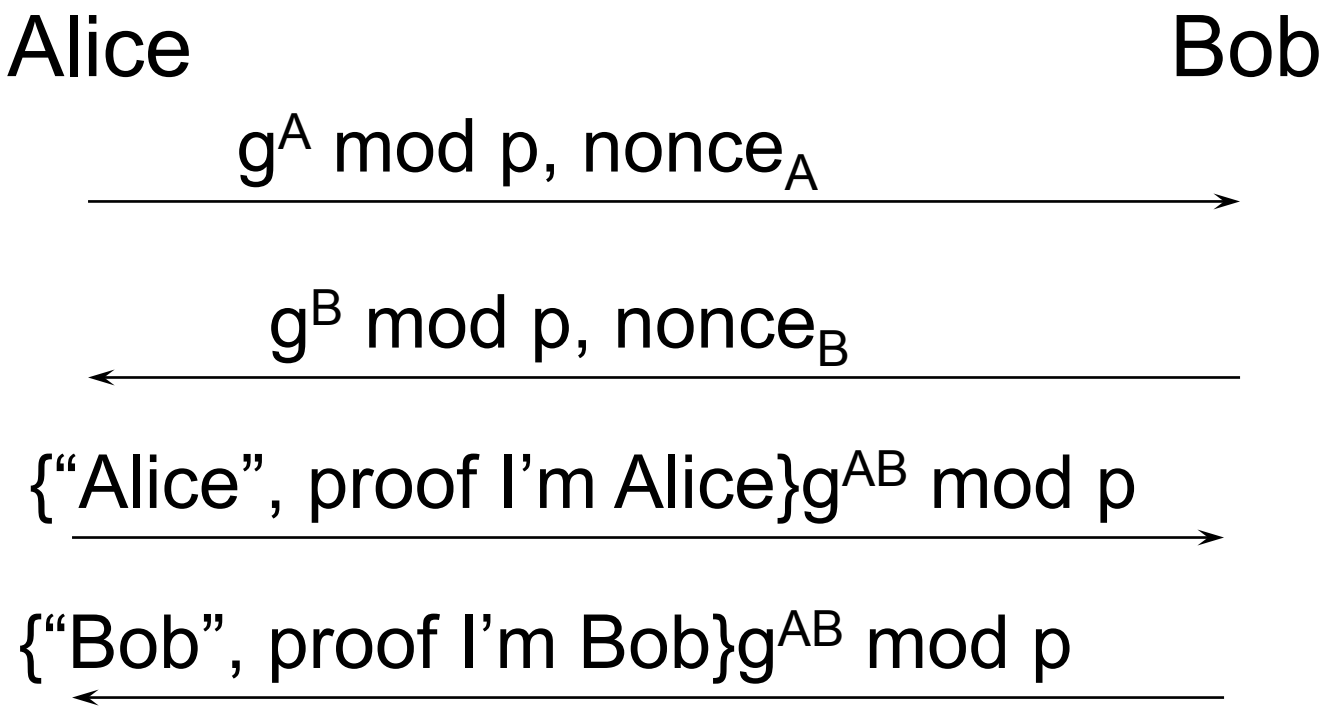
IKEv1:

- 9 msgs (ID hiding) or 6 to set up IPsec SA
- 8 different protocols
  - Which 8 did you count?
- Lots of other issues
  - public encryption keys, must know public key of other side before it sends cert. Original doesn't even allow sending cert.
  - original public encryption keys: separately encrypt fields with other side's public key (requires separate private key ops too). Undefined if field (e.g. name) bigger than an RSA block.

What was done for IKEv2:

- one protocol, 4 messages, ID hiding
- cleaned it up and simplified it a lot

General idea of IKEv2



IKEv1 vs the Version Number

Version number: doesn't say what to do if version number bigger… "SHOULD reject". So can't (in theory) count on v1 throwing away v2.

- 8 bit field, 4 bits major, 4 bits minor
- What major/minor should be: ignore minor
- But ISAKMP says "SHOULD reject if major larger than yours, or if the same, if minor larger than yours"
- So just like 8 bit field, but more complicated, and more likely to run out of numbers

From the Perlman et al. book:

- *"ISAKMP doesn't exactly say you reject it if the version is larger than yours. It says you SHOULD reject it. So implementations are free to ignore the version number, but perhaps feel a little guilty about it"*

Cookies

What are cookies good for in the context of IPSec

Image source: http://www.inkatrinaskitchen.com/2011/04/cookie-monster-cookies.html

Stateless Cookies vs. IKEv1

Photuris came up with stateless cookies so Alice proves she can receive from her claimed IP address before Bob devotes any state or significant computation

- ISAKMP has fields called "cookies", but lost the ability to be stateless
- Could have been stateless by copying info from msg 1 into msg 3
- Except they're required to be unique

The pair of cookies (initiator, responder) form the SA identifier

possible to have "cookie collision"

- Alice initiates to Bob, choosing A
- Carol initiates to Alice, choosing A
- Alice responds to Carol, choosing B
- Bob might choose B
- result: two connections Alice is involved in defined by (A,B)

Traffic Restrictions

IPsec policy: Traffic between these sets of IP addresses (or address ranges), and protocol types, and ports, must have this sort of cryptographic protection

Creating SA, specify “traffic selectors”

IKEv1:

- Initiator proposes. Responder (if has more restrictive policy) can just say “no”

IKEv2:

- allowed responder to narrow or say “single address pair”

Lost messages

IKEv1 kind of didn't say

- had "commit bit", defined incomprehensibly and almost oppositely in ISAKMP and IKE
  - ISAKMP: for Bob to tell Alice to wait for his ack
  - IKE: for Bob to tell Alice to send an ack

IKEv2: all messages request/response, and requester keeps sending until it gets a response

Fragmentation Attack

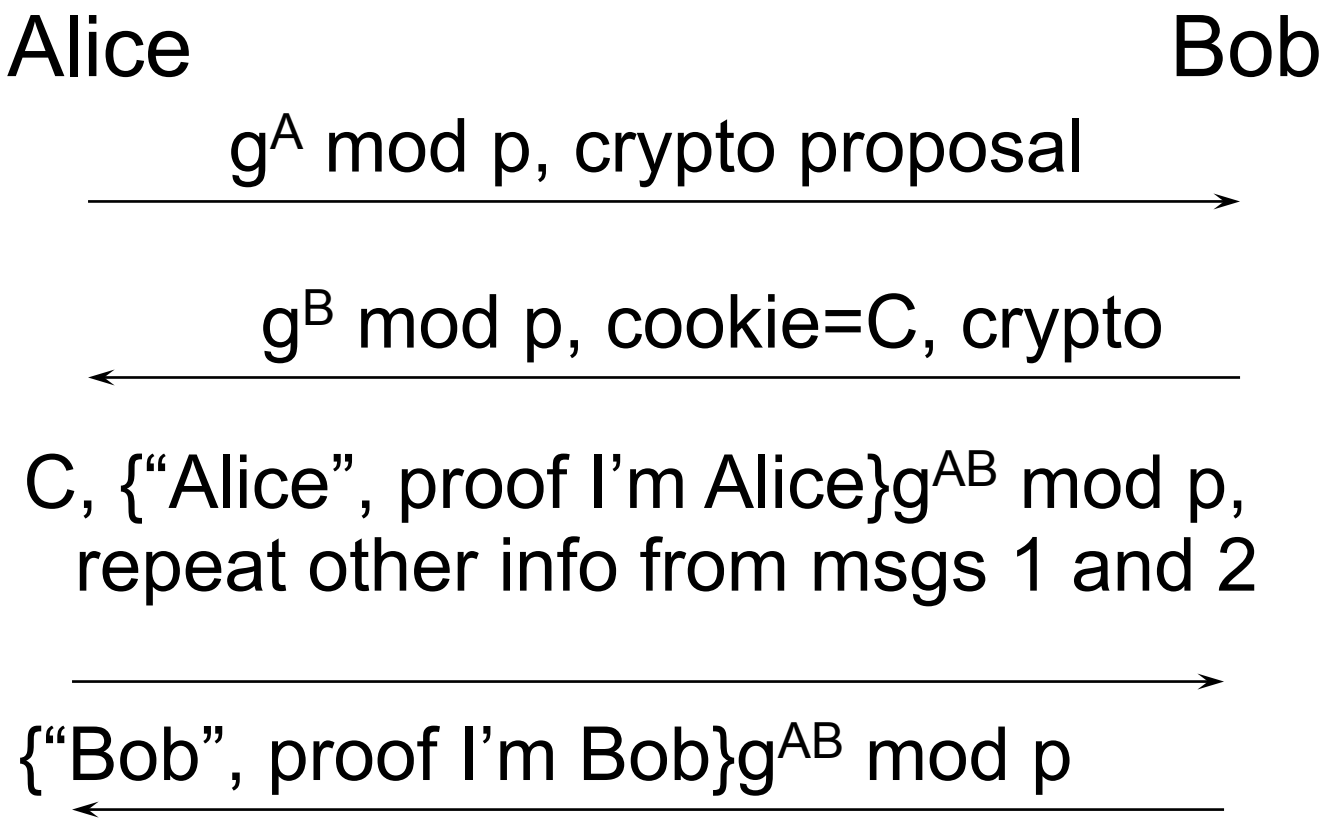
IKE runs on top of UDP

- Message 3 contains certs (depending on authentication choice) and can be really large (encryption of name, name can be very very long). If that's where cookie is returned, have fragmentation attack

Solutions:

- Shorten message 3

IKEv2 with stateless cookies 4-msgs



So, did optional pre-round trip

If Bob isn't under attack, 4-msgs

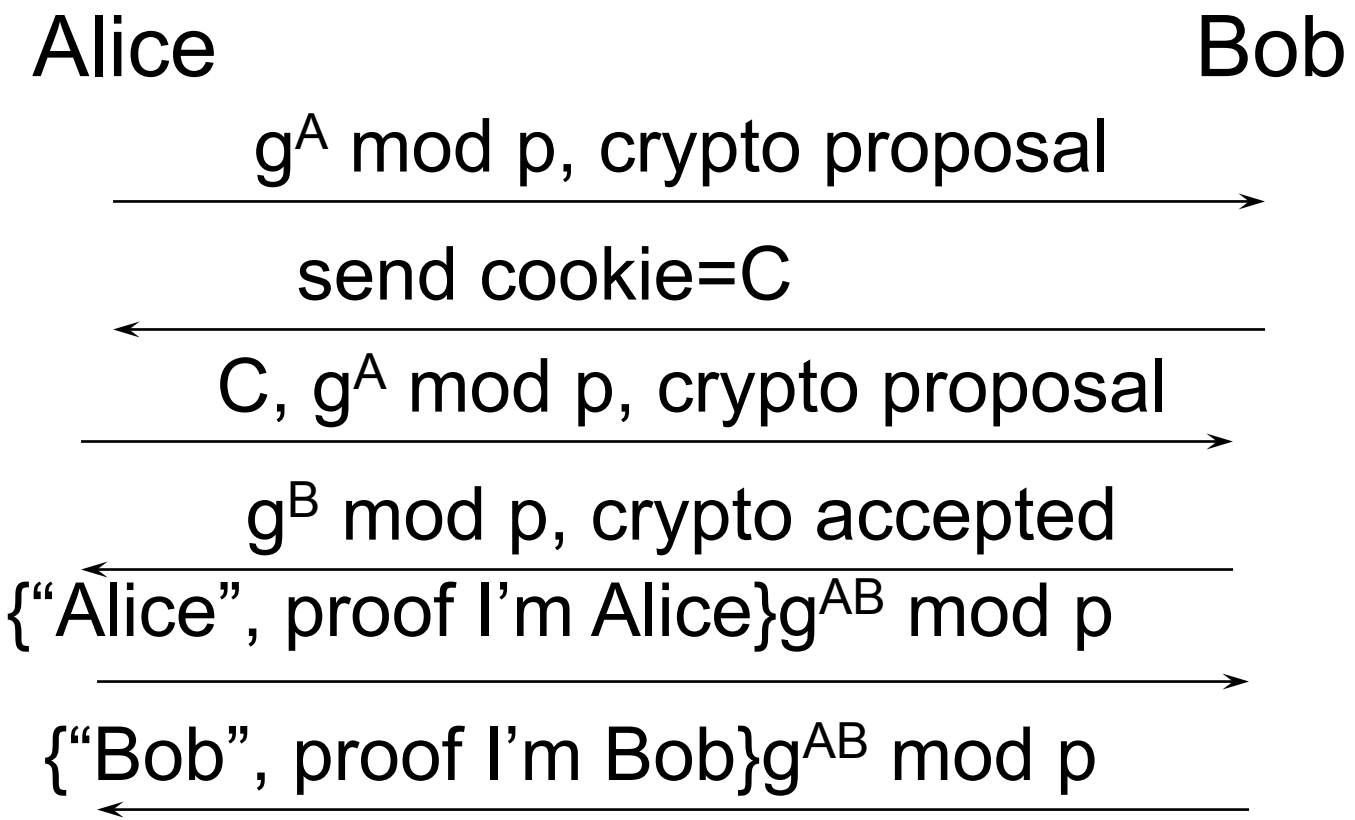
If he wants a cookie, he'll refuse message 1 that doesn't contain a cookie (and give a cookie to return)

This (among other advantages) allowed easy protection against fragmentation attack

Fragmentation defense

- With 4/6, msgs short until cookie verified
- IKE pass hint to reassembly code "this IP address preferred"
- Preferable to leave it on preferred list for minimum time

IKEv2 with stateless cookies, 4/6-msgs

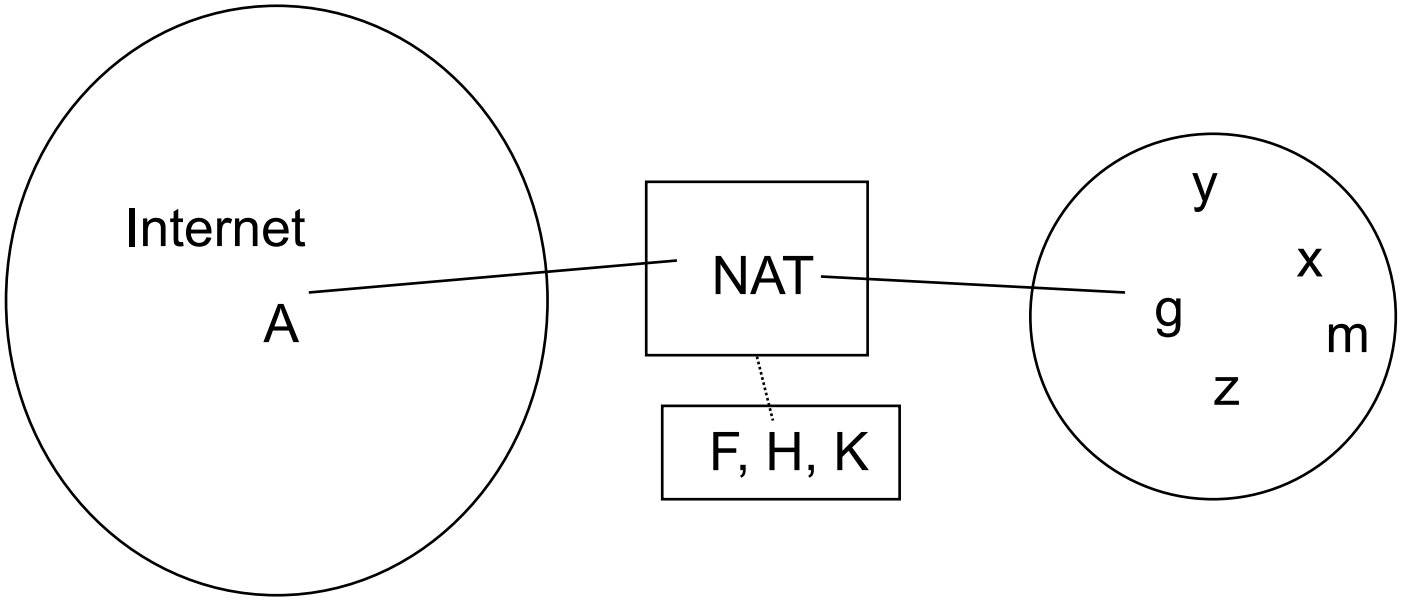


Other stuff added into IKEv2

NAT Traversal

Legacy authentication (token card, password)

Acquiring an address



Keeps mapping g/K. Overwrites source address on outgoing, overwrites destination address on incoming

NAT Issues

All sorts of IP protocols violate layering

TCP/UDP:

- "pseudoheader": checksum computed on addresses from IP header, and TCP header
- So NAT box must fudge TCP/UDP checksum

FTP

- sends addresses as text string "144.27.8.95"
- NAT must look inside FTP data to change address
- Worse yet! if changes TCP byte numbering: NAT must keep track and fix TCP ack, and msg #'s!

IPsec vs. NAT

AH:

- Safeguards against spoofing and man-in-the-middle attacks that change the source/destination IPs
- The hash includes source/destination IPs which are modified by the NAT server
- Verification of the hash fails on tunnel or transport mode
- Incompatible with NAT

ESP with Tunnel Mode:

- Full IP packet ciphered and signed but transmitted inside another packet
- Modification of the outer packet's IP address does not alter the inner packets content
- Compatible with NAT

IPsec vs. SSL

SSL is in application; IPsec in OS

- IPsec protects all apps

SSL is susceptible to a DoS attack:

- Attacker inserts bogus TCP segment into packet stream:
  - with correct TCP checksum and seq #s
- TCP acks segment and sends segment's payload up to SSL.
- SSL will discard since integrity check is bogus
- Real segment arrives:
  - TCP rejects since it has the wrong seq #
- SSL never gets real segment
  - SSL closes conn. since it can't provide lossless byte stream service

What happens if an attacker inserts a bogus IPsec datagram?

- IPsec at receiver drops datagram since integrity check is bogus; not marked as arrived in seq # window
- Real segment arrives, passes integrity check and passed up to TCP – no problem!

Additional References http://tools.ietf.org

RFC 2367: PF_KEY Interface
RFC 2401: Security Architecture for the Internet Protocol (IPsec overview) Obsolete by RFC 4301
RFC 2403: The Use of HMAC-MD5-96 within ESP and AH
RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH
RFC 2405: The ESP DES-CBC Cipher Algorithm With Explicit IV
RFC 2409: The Internet Key Exchange
RFC 2410: The NULL Encryption Algorithm and Its Use With IPsec
RFC 2411: IP Security Document Roadmap
RFC 2412: The OAKLEY Key Determination Protocol
RFC 2451: The ESP CBC-Mode Cipher Algorithms
RFC 2857: The Use of HMAC-RIPEMD-160-96 within ESP and AH
RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
RFC 3706: A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers
RFC 3715: IPsec-Network Address Translation (NAT) Compatibility Requirements
RFC 3947: Negotiation of NAT-Traversal in the IKE
RFC 3948: UDP Encapsulation of IPsec ESP Packets
RFC 4106: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
RFC 4301: Security Architecture for the Internet Protocol
RFC 4302: IP Authentication Header
RFC 4303: IP Encapsulating Security Payload
RFC 4304: Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)
RFC 4306: Internet Key Exchange (IKEv2) Protocol
RFC 4307: Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)
RFC 4308: Cryptographic Suites for IPsec
RFC 4309: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)
RFC 4478: Repeated Authentication in Internet Key Exchange (IKEv2) Protocol
RFC 4543: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH
RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)
RFC 4621: Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol
RFC 4718: IKEv2 Clarifications and Implementation Guidelines
RFC 4806: Online Certificate Status Protocol (OCSP) Extensions to IKEv2
RFC 4809: Requirements for an IPsec Certificate Management Profile
RFC 4835: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)
RFC 4945: The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX

RFC 4302, 4302, 4303

Acks & Recommended Reading

Selected slides of this chapter courtesy of

- Keith Ross, Steven Kent, G. Schäfer (TU Ilmenau) with changes of J. Schmitt (TU Kaiserslautern), R. Perlman, K. Ross, Y. Chen, W. Stallings (L. Brown); changes of myself incorporated

Highly recommended reading:

- http://tools.ietf.org/html/draft-ietf-ipsec-ikev2-tutorial-01

Recommended reading

- [KaPeSp2002] Charlie Kaufman, Radia Perlman, Mike Speciner: Network Security – Private Communication in a Public World, 2nd Edition, Prentice Hall, 2002, ISBN: 978-0-13-046019-6
- [Stallings2011] William Stallings, Network Security Essentials, 4th Edition, Prentice Hall, 2011, ISBN: 978-0-136-10805-4
- [Schäfer2003] G. Schäfer. Netzsicherheit - Algorithmische Grundlagen und Protokolle. dpunkt.verlag, 2003.

Copyright Notice

Contact

Prof. Dr.-Ing. Matthias Hollick
Department of Computer Science

SEEMOO
Mornewegstr. 32
64293 Darmstadt/Germany
matthias.hollick@seemoo.tu-darmstadt.de

Phone +49 6151 16-70920
Fax +49 6151 16-70921
www.seemoo.tu-darmstadt.de

Appendix: Native IPsec Host Implementations

In a native implementation, SAs are created in response to calls to the TLI (TCP or UDP traffic)

SPD is consulted when SA is created, using the parameters from the TLI call

- Note vulnerability in usual host context where application provides target host DNS name, but TLI uses mapped address!

Transmitted packets checked against parameters bound to the socket

Received packets are checked against (cached) SPD data for the SA

Appendix: IPsec in Gateways, BITS, BITW

SECURE MOBILE NETWORKING

No TLI, so each outbound packet must be examined, compared to the SPD (or SPD cache)

Caching requires a de-correlated SPD, to ensure cache entries don't violate ordering requirement

Inbound traffic check is easy, since SPI maps to SA, which has SPD selector values

For ID selectors, must create transient SPD entry (or cache entry) mapping ID to S/D address