



IPv6 Security

Kopaonik, Serbia & Montenegro, 2006

János Mohácsi

NIIF/HUNGARNET

[mohacsi\(at\)niif.hu](mailto:mohacsi(at)niif.hu)



Copy ...Rights

- *This slide set is the ownership of the 6DISS project via its partners*
- *The Powerpoint version of this material may be reused and modified only with written authorization*
- *Using part of this material must mention 6DISS courtesy*
- *PDF files are available from www.6diss.org*



Contributions

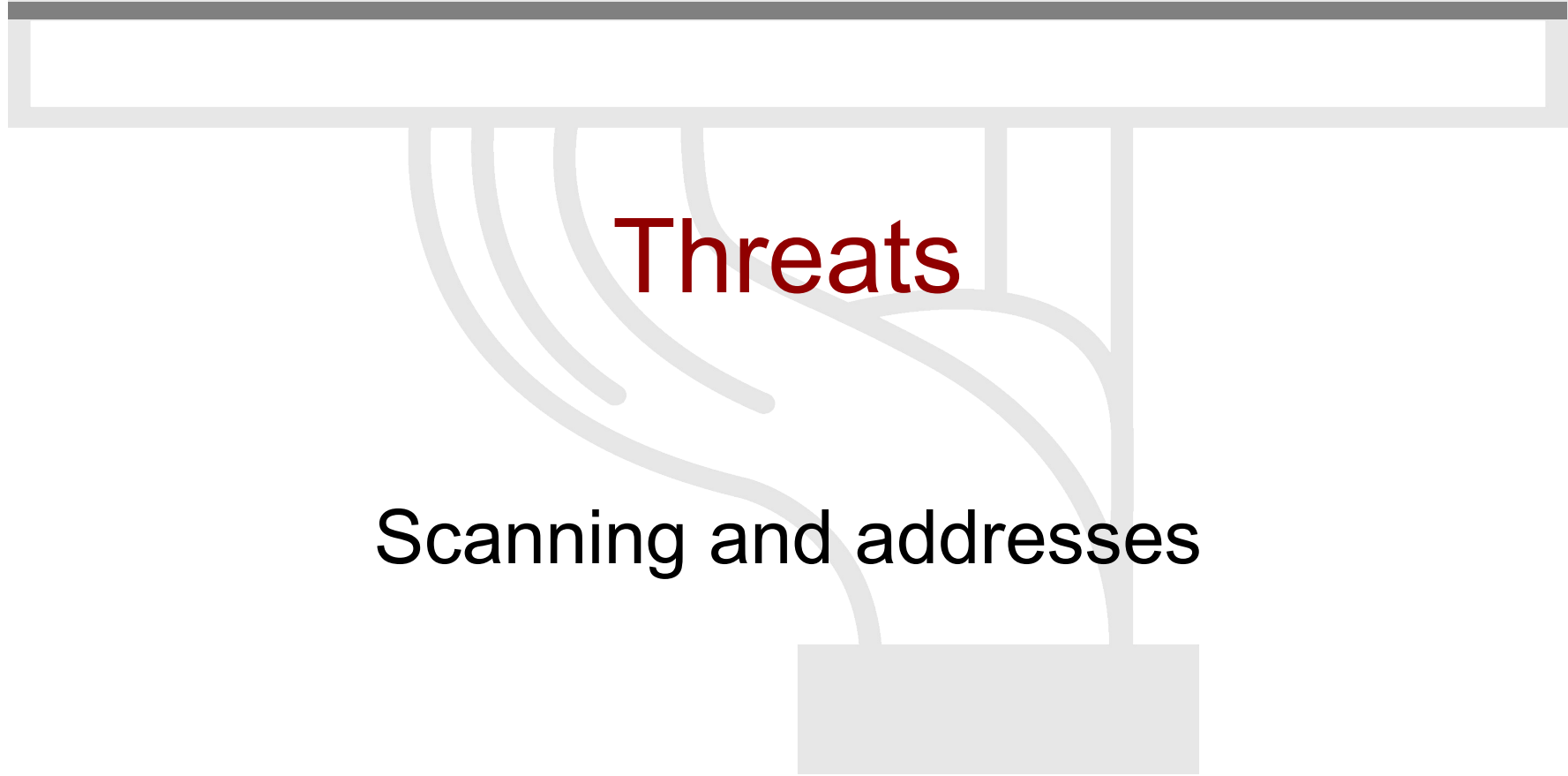
- Main authors
 - János Mohácsi, NIIF/HUNGARNET - Hungary
 - Laurent Toutain, ENST-Bretagne – IRISA, France
- Contributors
 - Bernard Tuy, Renater, France
 - Jérôme Durand, Renater, France



Outline of the presentation

- Threats against IPv6– comparing with IPv4
 - Scanning
 - Unauthorised access – IPv6 firewalls review
 - Fragmentation attacks
 - Spoofing
 - Host initialisation attacks
 - Broadcast amplification attacks
 - Other types of attacks
- Specific IPv6 related problems
- IPv6 Security infrastructure
 - IPSec





Scanning in IPv6

- Subnet Size is much larger
 - Default subnets in IPv6 have 2^{64} addresses (approx. 18×10^{18}). Exhaustive scan on every address on a subnet is no longer reasonable (if 1 000 000 address per second then $> 500\ 000$ year to scan)
 - NMAP doesn't even support for IPv6 network scanning



Scanning in IPv6 /2

- IPv6 Scanning methods are likely to change
 - Public servers will still need to be DNS reachable giving attacker some hosts to attack – this is not new!
 - Administrators may adopt easy to remember addresses (::1, ::2, ::53, or simply IPv4 last octet)
 - EUI-64 address has “fixed part”
 - Ethernet card vendors guess
 - New techniques to harvest addresses – e.g. from DNS zones, logs
 - Deny DNS zone transfer
 - By compromising routers at key transit points in a network, an attacker can learn new addresses to scan
- Other possible network hiding: DNS splitting



Scanning in IPv6 / 3

- New attack vectors “All node/router addresses”
- New Multicast Addresses - IPv6 supports new multicast addresses that can enable an attacker to identify key resources on a network and attack them
- For example, all nodes (FF02::1), all routers (FF05::2) and all DHCP servers (FF05::5)
- These addresses must be filtered at the border in order to make them unreachable from the outside – this is the default if no IPv6 multicasting enabled.



Security of IPv6 addresses

- Privacy enhanced addresses as defined RFC 3041
 - prevents device/user tracking from
 - makes accountability harder
- New privacy extended IPv6 addresses generated CGA (cryptographically generated addresses)
 - maintains privacy
 - accountability possible by link administrators
- New feature: Host ID could be a token to access to a network. – additional security possible





Unauthorized Access control in IPv6

- Policy implementation in IPv6 with Layer 3 and Layer 4 is still done in firewalls
- Some design considerations! – see next slides also
 - Filter site-scoped multicast addresses at site boundaries
 - Filter IPv4 mapped IPv6 addresses on the wire
 - Multiple address per interfaces

Action \	Src	Dst	Src port	Dst port
permit	a:b:c:d::e	x:y:z:w::v	any	ssh
deny	any	any		



Unauthorized Access control in IPv6

- non-routable + bogon address filtering slightly different
 - in IPv4 easier deny non-routable + bogon
 - in IPv6 easier to permit legitimate (almost)

Action	Src	Dst	Src port	Dst port
deny	2001:db8::/32	host/net		
permit	2001::/16	host/net	any	service
permit	2002::/16	host/net	any	service
permit	2003::/16	host/net	any	service
permit	3ffe::/16	host/net	any	service
deny	any	any		



IANA allocations in September 2005

2001:0000::/23	IANA	2001:4400::/23	APNIC
2001:0200::/23	APNIC	2001:4600::/23	RIPE NCC
2001:0400::/23	ARIN	2001:4800::/23	ARIN
2001:0600::/23	RIPE NCC	2001:4A00::/23	RIPE NCC
2001:0800::/23	RIPE NCC	2001:4C00::/23	RIPE NCC
2001:0A00::/23	RIPE NCC	2001:5000::/20	RIPE NCC
2001:0C00::/23	APNIC	2001:8000::/19	APNIC
2001:0E00::/23	APNIC	2001:A000::/20	APNIC
2001:1200::/23	LACNIC	2002:0000::/16	6to4
2001:1400::/23	RIPE NCC	2003:0000::/18	RIPE NCC
2001:1600::/23	RIPE NCC	2400:0000::/19	APNIC
2001:1800::/23	ARIN	2400:2000::/19	APNIC
2001:1A00::/23	RIPE NCC	2400:4000::/21	APNIC
2001:1C00::/22	RIPE NCC	2600:0000::/22	ARIN
2001:2000::/20	RIPE NCC	2604:0000::/22	ARIN
2001:3000::/21	RIPE NCC	2608:0000::/22	ARIN
2001:3800::/22	RIPE NCC	260C:0000::/22	ARIN
2001:3C00::/22	RESERVED	2A00:0000::/21	RIPE NCC
2001:4000::/23	RIPE NCC	2A01:0000::/23	RIPE NCC
2001:4200::/23	ARIN		

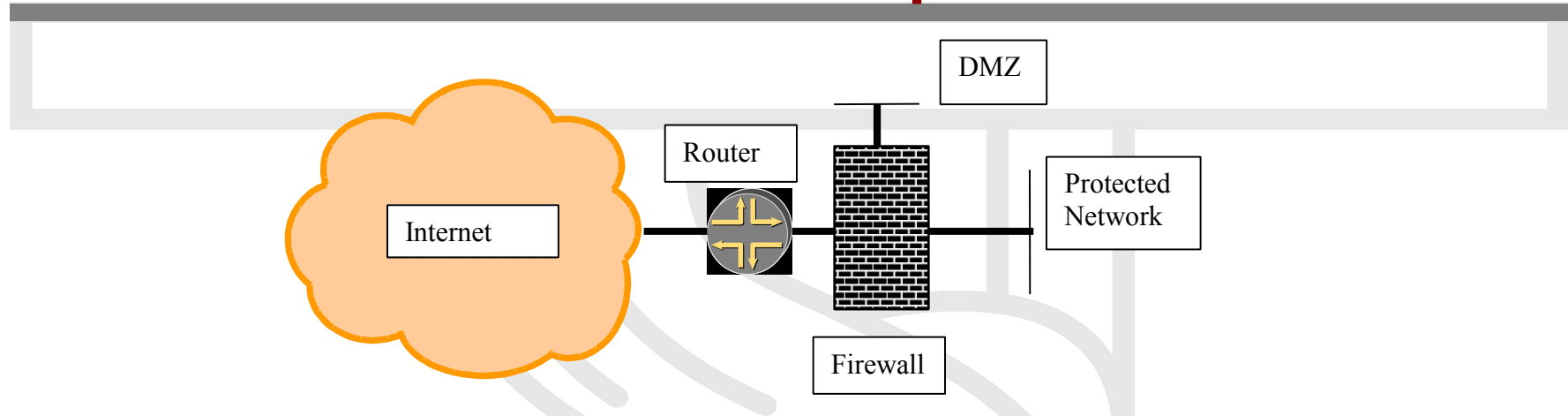


IPv6 Firewalls

- IPv6 architecture and firewall - requirements
 - No need to NAT – same level of security with IPv6 possible as with IPv4 (security and privacy)
 - even better: e2e security with IPSec
 - Weaknesses of the packet filtering cannot be made hidden by NAT
 - “IPv6 does not require end-to-end connectivity, but provides end-to-end addressability”
 - Support for IPv6 header chaining
 - Support for IPv4/IPv6 transition and coexistence
 - Not breaking IPv4 security



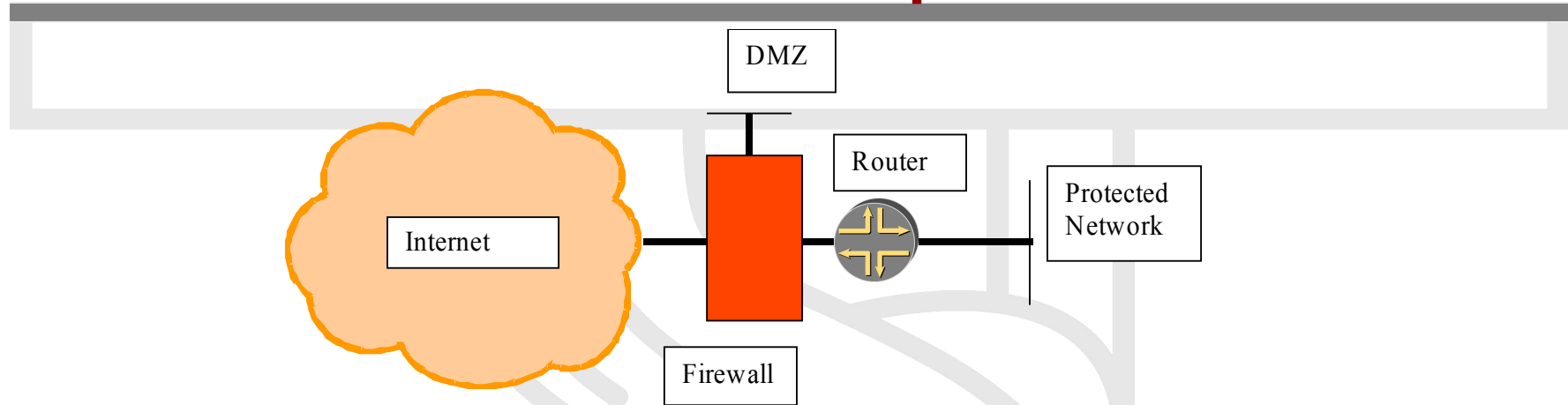
IPv6 firewall setup - method 1



- Internet ↔ router ↔ firewall ↔ net architecture
- Requirements:
 - Firewall must support/recognise ND/NA filtering
 - Firewall must support RS/RA if SLAAC is used
 - Firewall must support MLD messages if multicast is required



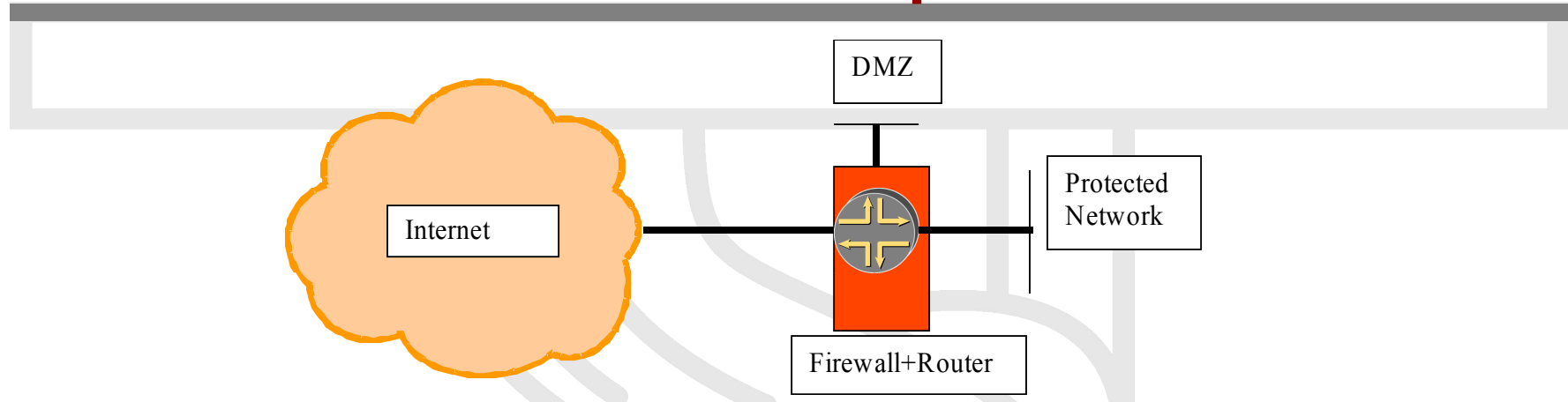
IPv6 firewall setup - method2



- Internet ↔ firewall ↔ router ↔ net architecture
- Requirements:
 - Firewall must support ND/NA
 - Firewall should support filtering dynamic routing protocol
 - Firewall should have large variety of interface types



IPv6 firewall setup - method3



- Internet ↔ firewall/router(edge device) ↔ net architecture
- Requirements
 - Can be powerful - one point for routing and security policy – very common in SOHO (DSL/cable) routers
 - Must support what usually router AND firewall do



Problems with ICMPv6

- ICMPv6 is a fundamental component of IPv6 networks
 - Some parts of ICMPv6 have an essential role in establishing communications
 - Less of an 'auxiliary' than ICMP in IPv4
- Some ICMPv6 messages can be a threat to open networks
 - IPsec not generally applicable
- Firewall filtering important for maintaining security
- Need to balance effective IPv6 communications against security needs



Major ICMPv6 Functions

- Error messages (4 types)
- Echo Request and Response
- Neighbor finding (NS, NA, RS, RA)
 - Duplicate Address Detection
 - IP and Link Layer Address exchange
 - Router Identification
 - Stateless Address Auto-configuration
- Network renumbering (NS, NA + renumber)
- Path MTU determination (Packet Too Big)
- Multicast Listener Discovery (4 messages)
- Mobile IPv6 support (4 messages)
- Node information lookup (2 messages)



Classifying ICMPv6 Functions and Messages

- Error and Informational Messages
- Addressing
 - Lots of different possibilities
- Network Topology and Address Scopes
 - Intra-link
 - End-to-end
 - ‘Any-to-end’ Role in Establishing Communications



Possible Firewall setup

- No blind ICMPv6 filtering possible:

	Echo request/reply	Debug
	No route to destination	Debug – better error indication
	TTL exceeded	Error report
	Parameter problem	Error report
IPv6 specific	NS/NA	Required for normal operation – except static ND entry
	RS/RA	For Stateless Address Autoconfiguration
	Packet too big	Path MTU discovery
	MLD	Requirements in for multicast in architecture 1

Firewall setup 2

- No blind IP options (→ extension Header) filtering possible:

Hop-by-hop header	What to do with jumbograms or router alert option? – probably log and discard – what about multicast join messages?
Routing header	Source routing – in IPv4 it is considered harmful, but required for IPv6 mobility – log and discard if you don't support MIPv6, otherwise enable only Type 2 routing header for Home Agent of MIPv6
ESP header	Process according to the security policy
AH header	Process according to the security policy
Fragment header	All but last fragments should be bigger than 1280 octets



Interoperability of filtered applications

- FTP:
 - Very complex: PORT, LPRT, EPRT, PSV, EPSV, LPSV (RFC 1639, RFC 2428)
 - virtually no support in IPv6 firewalls
 - HTTP seems to be the next generation file transfer protocol with WEBDAV and DELTA
- Other non trivially proxy-able protocol:
 - no support (e.g.: H.323)



Overview of IPv6 firewalls

	IPFilter 4.1	PF 3.6	IP6fw	Iptables	Cisco ACL	Cisco PIX 7.0	Juniper firewall	Juniper NetScreen	Windows XP SP2
Portability	Excellent	Good	Average	Weak	Weak	Weak	Weak	Weak	Weak
ICMPv6 support	Good	Good	Good	Good	Good	Good	Good	Good	Good
Neighbor Discovery	Excellent	Excellent	Good	Excellent	Excellent	Excellent	Good	Excellent	Weak
RS /RA support	Excellent	Excellent	Good	Excellent	Excellent	Excellent	Excellent	Excellent	Good
Extension header support	Good	Good	Good	Excellent	Good	Good	Good	Good	Weak
Fragmentation support	Weak	Complete block	Weak	Good	Weak	Average	Weak	Average	Weak
Stateful firewall	Yes	Yes	No	Csak USAGI	Reflexive firewall since 12.3(11)T	Yes	ASP necessary	Yes	No
FTP proxy	No	Next version	No	No		Yes	No	No	No
Other	QoS support	QoS support, checking packet validity	Predefined rules in *BSD	EUI64 check,	Time based ACL	Time based ACL	TCP flag support only in upcoming 7.2, HW based	IPSec VPN, routing support	Graphical and central configuration





Threats

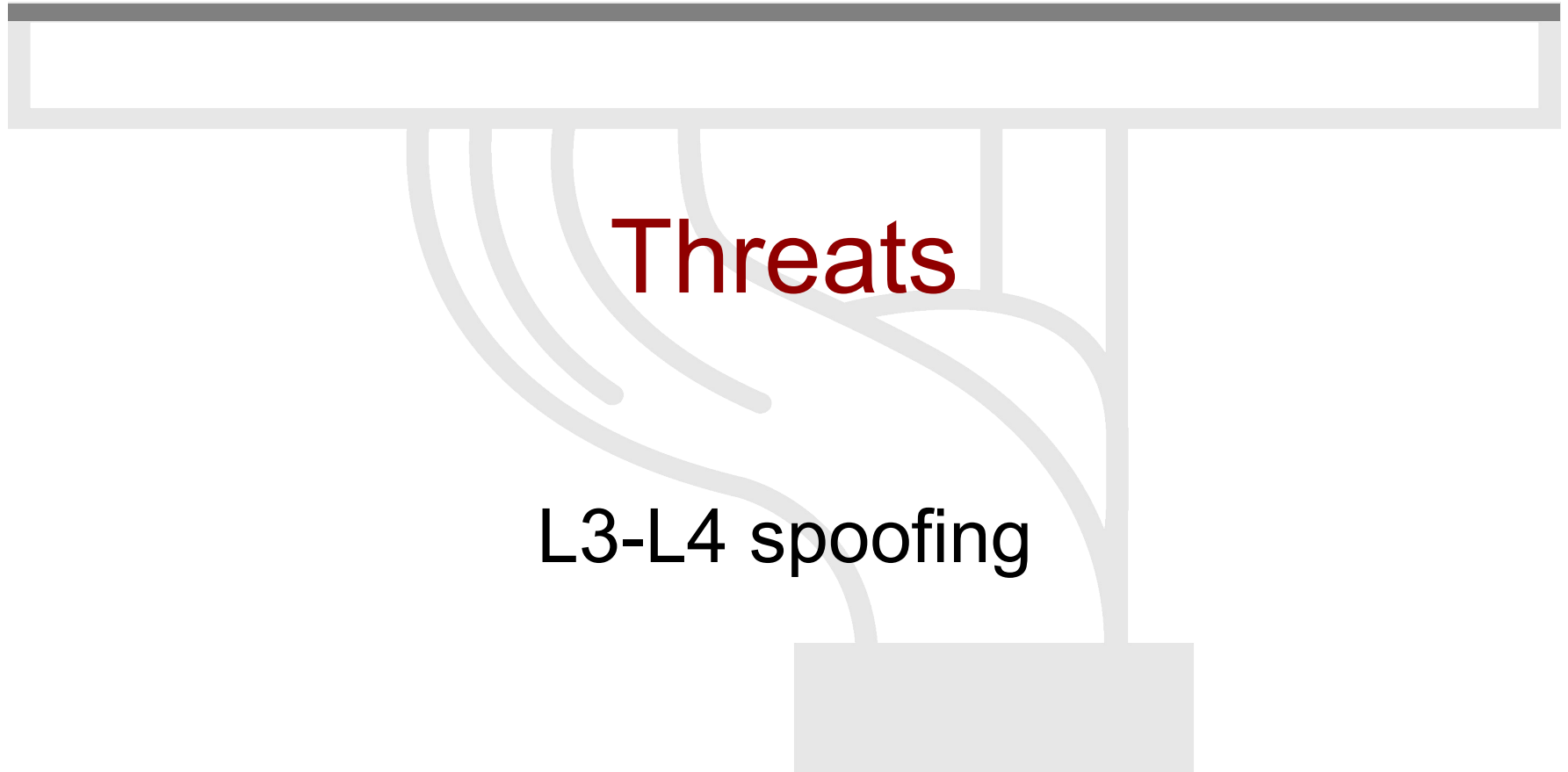
Fragmentation and header handling



Header Manipulation and Fragmentation Best Practices

- Deny IPv6 fragments destined to an internetworking device - Used as a DOS vector to attack the infrastructure
- Ensure adequate IPv6 fragmentation filtering capabilities. For example, drop all packets with the routing header if you don't have MIPv6
- Potentially drop all fragments with less than 1280 octets (except the last fragment)
- All fragment should be delivered in 60 seconds otherwise drop





L3- L4 Spoofing in IPv6

- While L4 spoofing remains the same, IPv6 address are globally aggregated making spoof mitigation at aggregation points easy to deploy
- Can be done easier since IPv6 address is hierarchical
- However host part of the address is not protected
 - You need IPv6 \leftrightarrow MAC address (user) mapping for accountability!





Threats

IPv4 ARP and DHCP attacks -
Subverting host initialization



Autoconfiguration/Neighbor Discovery

- Neighbor Discovery ~ security ~ Address Resolution Protocol
 - No attack tools – arp cache poisoning
 - No prevention tools – dhcp snooping
- Better solution with SEND
 - based on CGA: $\text{token1} = \text{hash}(\text{modifier}, \text{prefix}, \text{publickey}, \text{collision-count})$
 - RFC3972 available!
- DHCPv6 with authentication is possible
- ND with IPSec also possible





Amplification (DDoS) Attacks

- There are no broadcast addresses in IPv6
 - This would stop any type of amplification/"Smurf" attacks that send ICMP packets to the broadcast address
 - Global multicast addresses for special groups of devices, e.g. link-local addresses, site-local addresses, all site-local routers, etc.
- IPv6 specifications forbid the generation of ICMPv6 packets in response to messages to global multicast addresses (exception Packet too big message – it is questionable practice).
 - Many popular operating systems follow the specification
 - Still uncertain on the danger of ICMP packets with global multicast source addresses



Mitigation of IPv6 amplification

- Be sure that your host implementation follow the RFC 2463
- Implement RFC 2827 ingress filtering
- Implement ingress filtering of IPv6 packets with IPv6 multicast source address



Other threats

- IPv6 Routing Attack
 - Use traditional authentication mechanisms for BGP and IS-IS.
 - Use IPsec to secure protocols such as OSPFv3 and RIPng
- Viruses and Worms
- Sniffing
 - Without IPsec, IPv6 is no more or less likely to fall victim to a sniffing attack than IPv4
- Application Layer Attacks
 - Even with IPsec, the majority of vulnerabilities on the Internet today are at the application layer, something that IPsec will do nothing to prevent
- Man-in-the-Middle Attacks (MITM)
 - Without IPsec, any attacks utilizing MITM will have the same likelihood in IPv6 as in IPv4
- Flooding
 - Flooding attacks are identical between IPv4 and IPv6





Specific IPv6 related problems



Specific IPv6 related threats

Transition Mechanisms



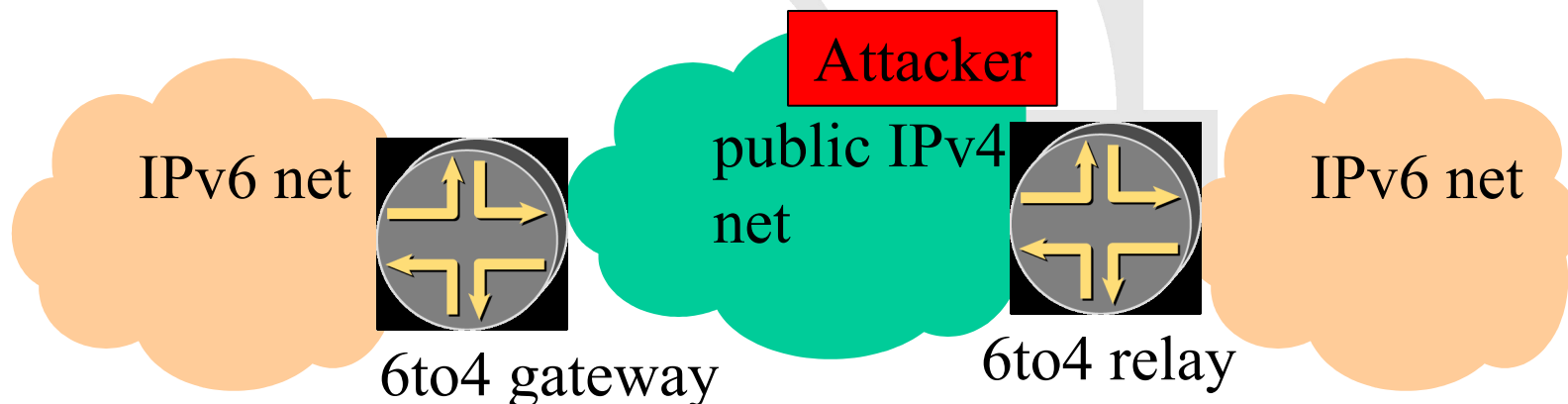
IPv6 transition mechanisms

- ~15 methods possible in combination
- Dual stack:
 - enable the same security for both protocol
- Tunnels:
 - ip tunnel – punching the firewall (protocol 41)
 - gre tunnel – probable more acceptable since used several times before IPv6



L3 – L4 Spoofing in IPv4 with 6to4

- For example, via 6to4 tunneling spoofed traffic can be injected from IPv4 into IPv6.
 - IPv4 Src: Spoofed IPv4 Address
 - IPv4 Dst: 6to4 Relay Anycast (192.88.99.1)
 - IPv6 Src: 2002:: Spoofed Source
 - IPv6 Dst: Valid Destination

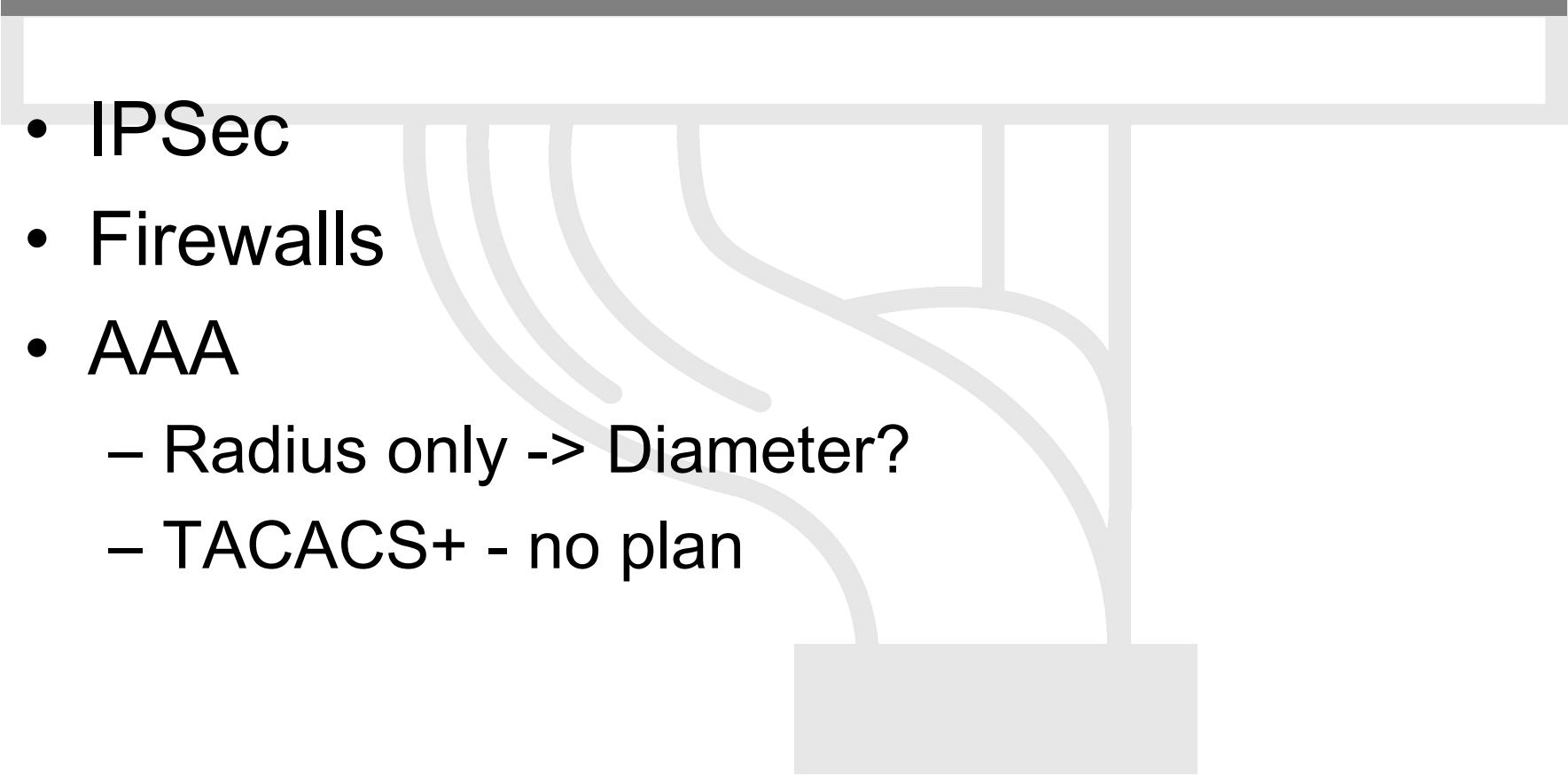


Mixed IPv4/IPv6 environments

- There are security issues with the transition mechanisms
 - Tunnels are extensively used to interconnect networks over areas supporting the “wrong” version of protocol
 - Tunnel traffic many times has not been anticipated by the security policies. It may pass through firewall systems due to their inability check two protocols in the same time
- Do not operate completely automated tunnels
 - Avoid “translation” mechanisms between IPv4 and IPv6, use dual stack instead
 - Only authorized systems should be allowed as tunnel end-points
 - Automatic tunnels can be secured by IPSec



IPv6 security infrastructure

- 
- IPSec
 - Firewalls
 - AAA
 - Radius only -> Diameter?
 - TACACS+ - no plan



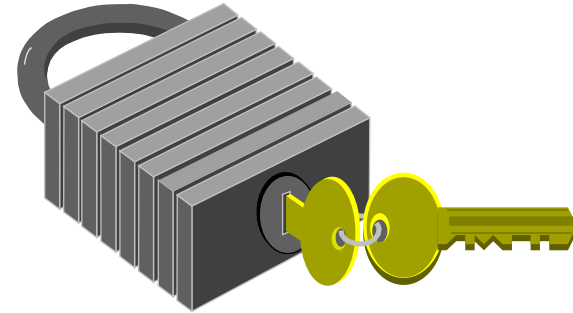


IPv6 Security infrastructure

IPSec



IPSec



- general IP Security mechanisms
- provides
 - authentication
 - confidentiality
 - key management - requires a PKI infrastructure (IKE) – new simplified and unified IKEv2 will be available soon.
- applicable to use over LANs, across public & private WANs, & for the Internet
- IPSec is not a single protocol. Instead, IPSec provides a set of security algorithms plus a general framework that allows a pair of communicating entities to use whichever algorithms provide security appropriate for the communication.
- IPSec is mandated in IPv6 – you can rely on for e2e security



Security: IPsec

- Work made by the IETF IPsec wg
- Applies to both IPv4 and IPv6 and its implementation is:
 - Mandatory for IPv6
 - Optional for IPv4
- IPsec Architecture: RFC 2401
- IPsec services
 - Authentication
 - Integrity
 - Confidentiality
 - Replay protection
- IPsec modes: Transport Mode & Tunnel Mode
- IPsec protocols: AH (RFC 2402) & ESP (RFC 2406)

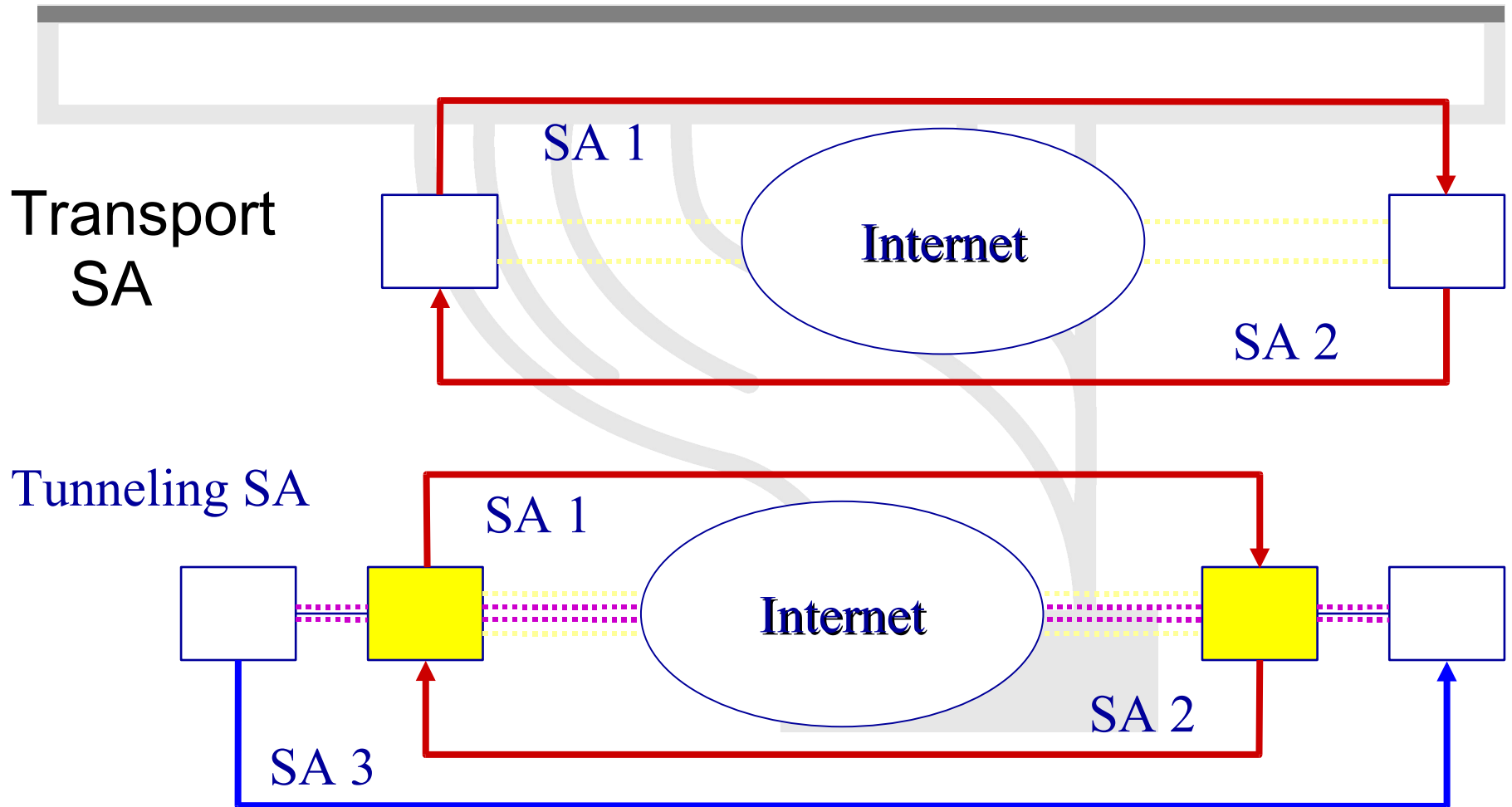


IPsec Architecture (RFC 2401)

- Security Policies: Which traffic is treated?
- Security Associations: How traffic is processed?
 - SA –contract between two parties (security protocol, algorithm, keys, etc.)
 - Generally unidirectional
- Security Protocols: Which protocols (extension headers) are used?
- Key Management: Internet Key Exchange (IKE)
- Algorithms: Authentication and Encryption



Types of SAs

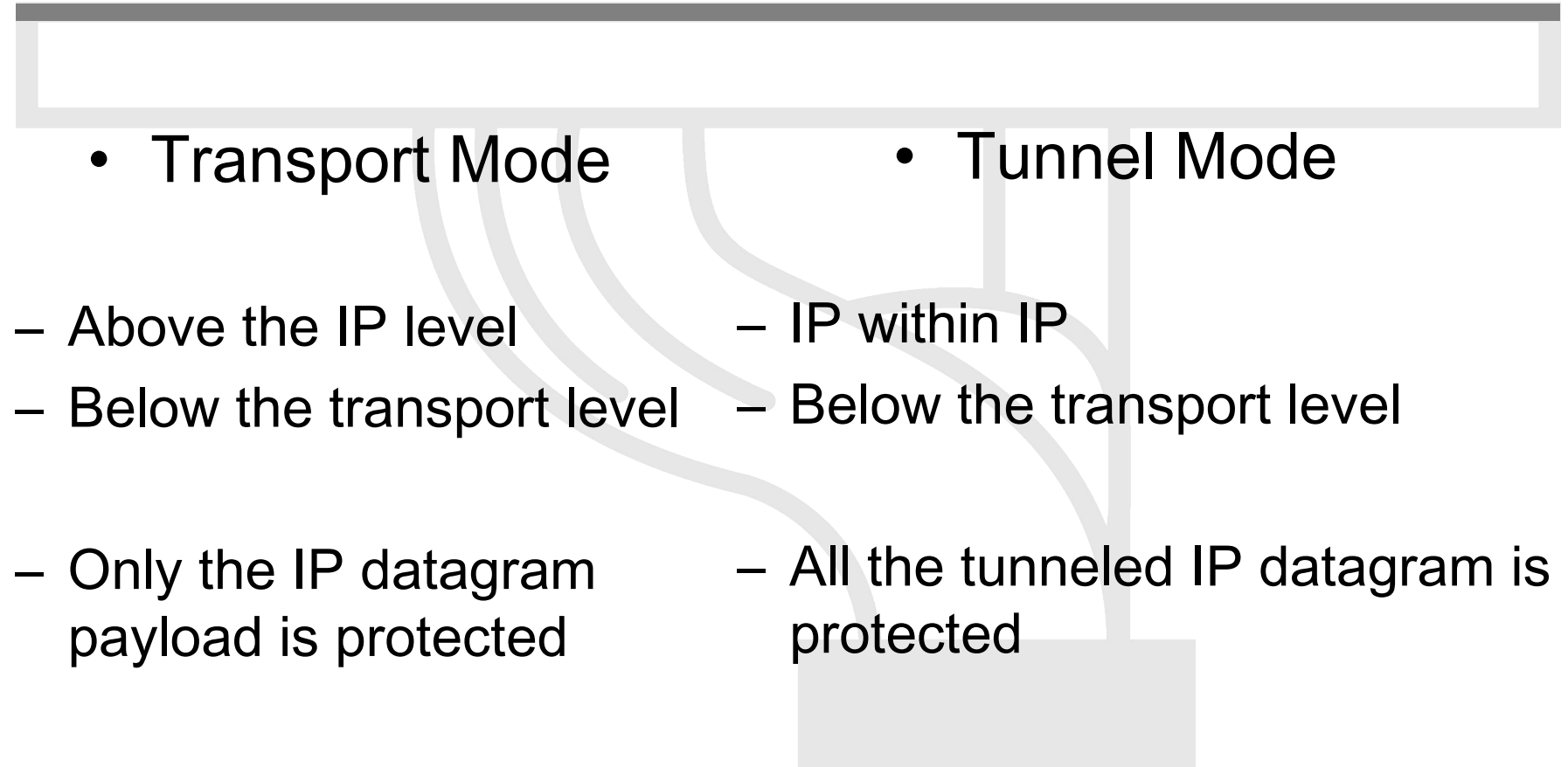


IPSec Security Databases

- Security Policy Database (SPD)
 - Rules for a certain communication relations (machines): protected, passed, rejected;
 - Pointer to SAD entry.
- Security Association Database (SAD)
 - used IPSec protocol;
 - used algorithm;
 - keys and other parameters.



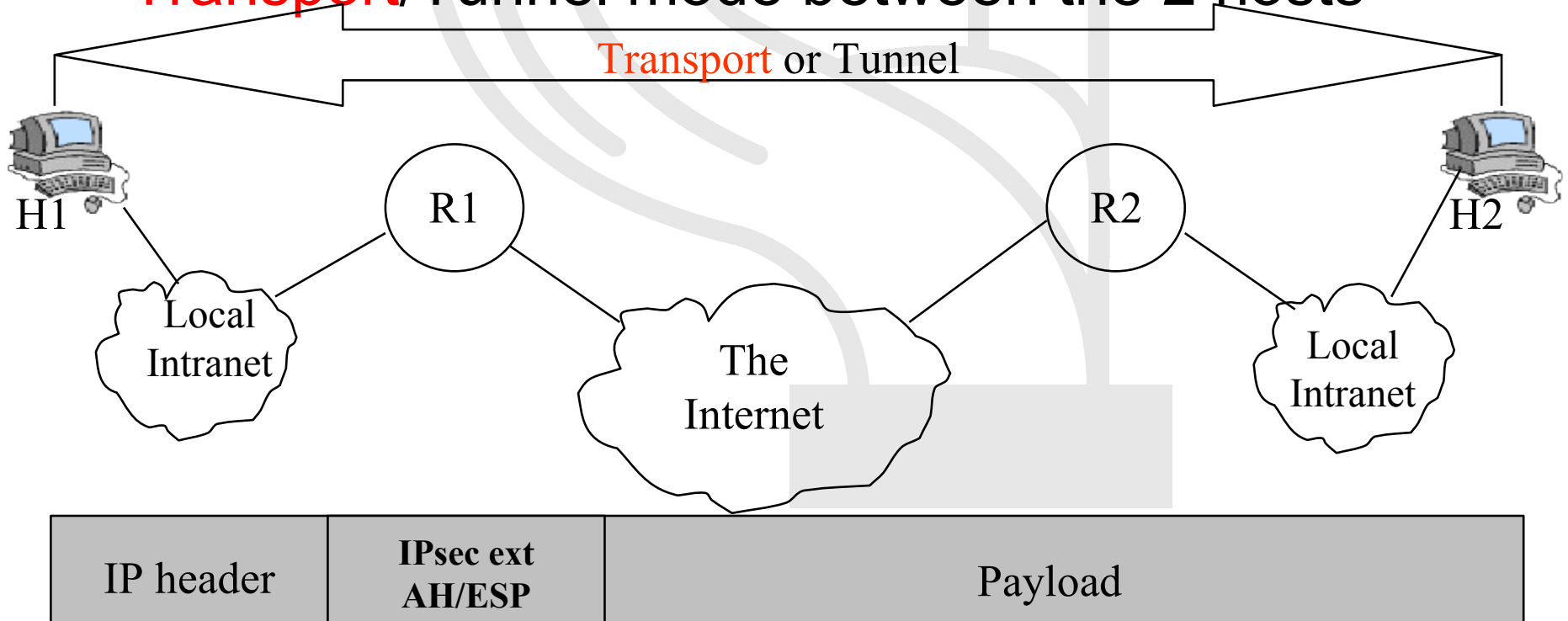
IPsec Modes



IPsec Scenarios

Scenario 1: H2H

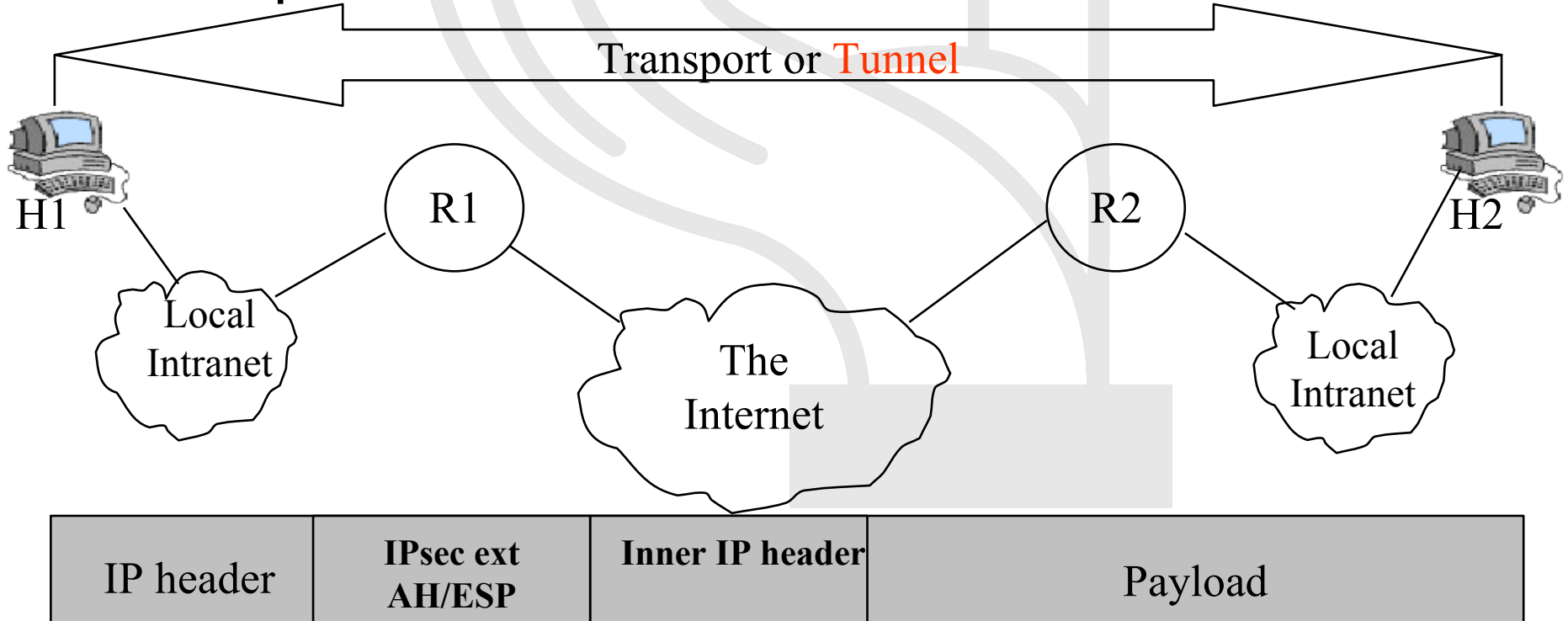
- End-to-end service
- **Transport**/Tunnel mode between the 2 hosts



IPsec Scenarios

Scenario 1: H2H

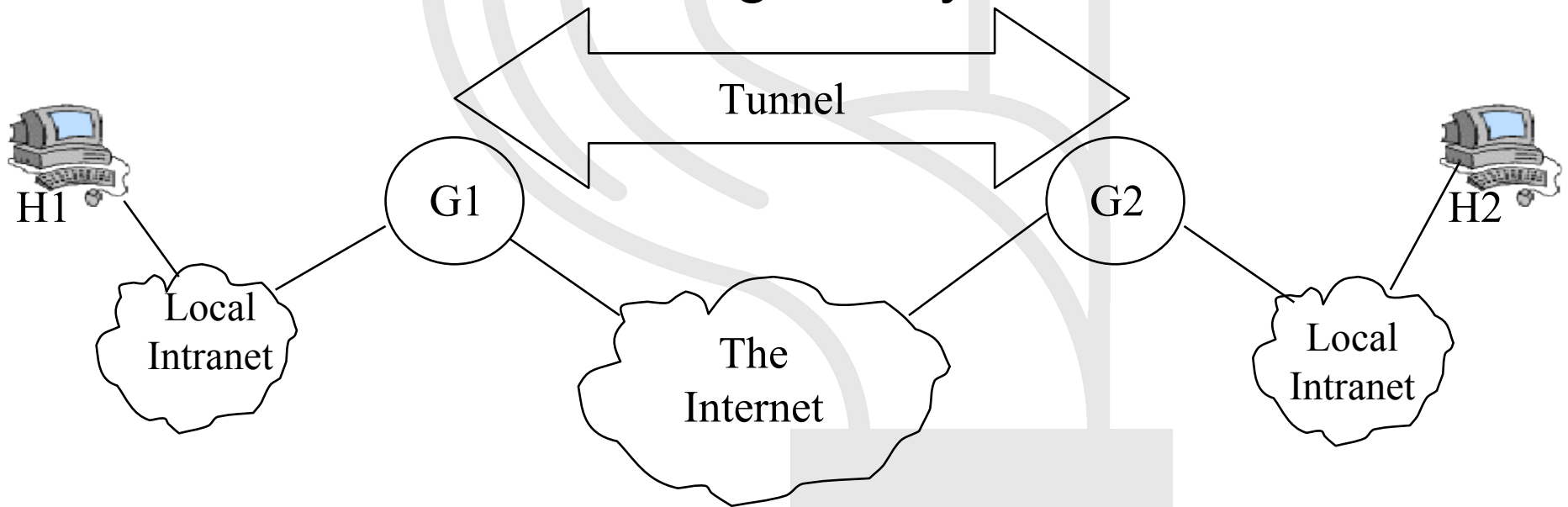
- End-to-end service
- Transport/**Tunnel** mode between the 2 hosts



IPsec Scenarios

Scenario 2: G2G

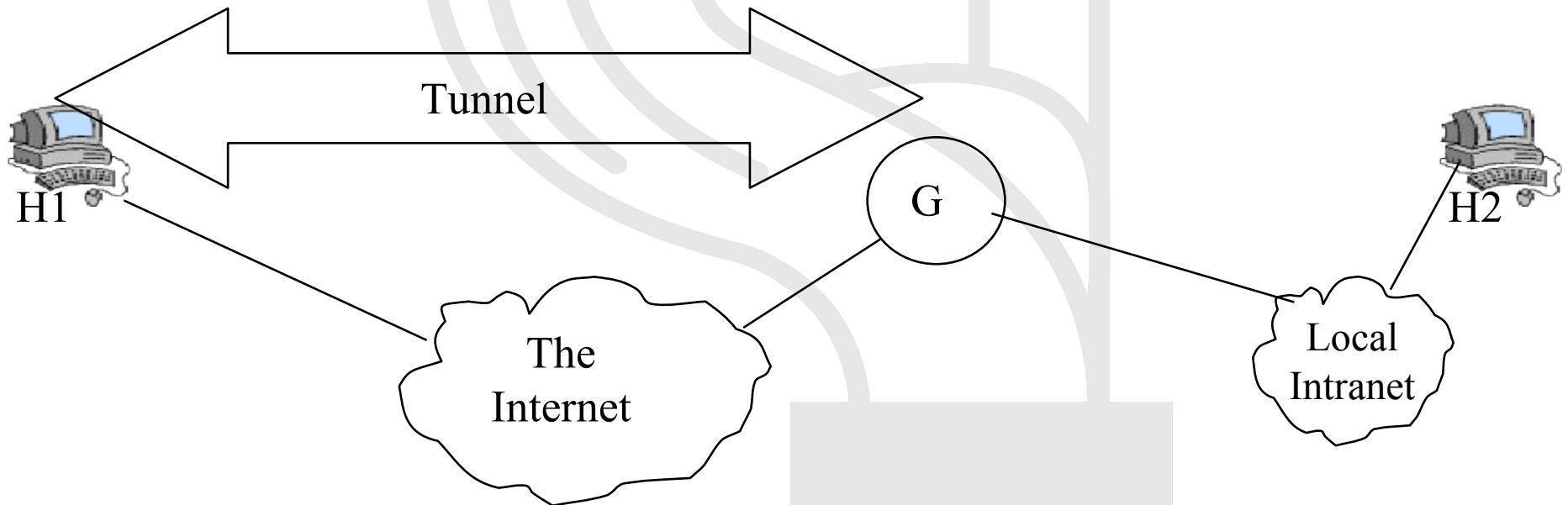
- VPN, Site-to-Site/ISP agreements, ...
- Tunnel between the 2 gateways



IPsec Scenarios

Scenario 3: H2G, G2H

- Dial-in users
- Tunnel between the “external” host and the gateway



IPsec Protocols

- Authentication Header (AH)
 - RFC 2402
 - Protocol# (Next Header) = 51
 - Provides:
 - Connectionless Integrity
 - Data origin authentication
 - Replay protection
 - Is inserted
 - In Transport mode: After the IP header and before the upper layer protocol (UDP, TCP, ...)
 - In Tunnel mode: Before the original IP header (the entire IP header is protected)
- Encapsulation Security Payload Header (ESP)
 - RFC 2406
 - Protocol# (Next Header) = 50
 - Provides:
 - Connectionless Integrity
 - Data origin authentication
 - Replay protection
 - Confidentiality
 - Is inserted
 - In Transport mode: After the IP header and before the upper layer protocol
 - In Tunnel mode: before an encapsulated IP header



IPsec: Protocols, services & modes combinations

	Transport Mode	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header	Authenticates entire inner IP datagram (header + payload), + selected portions of the outer IP header
ESP	Encrypts IP payload	Encrypts inner IP datagram
ESP with Authentication	Encrypts IP payload and authenticates IP payload but not IP header	Encrypts and authenticates inner IP datagram

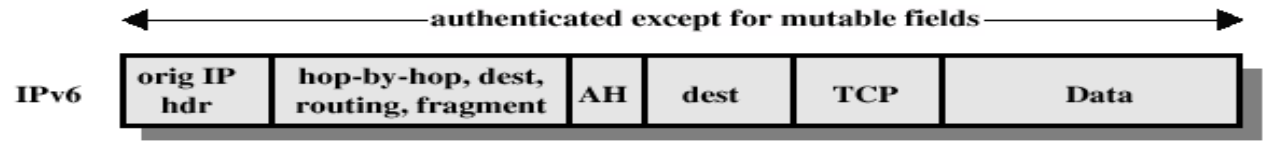
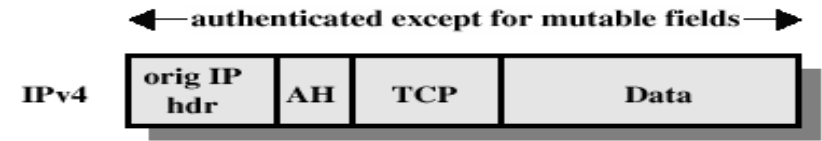
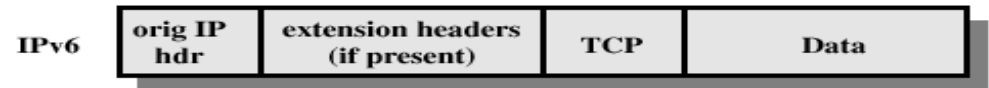


Authentication Header (AH) Protocol

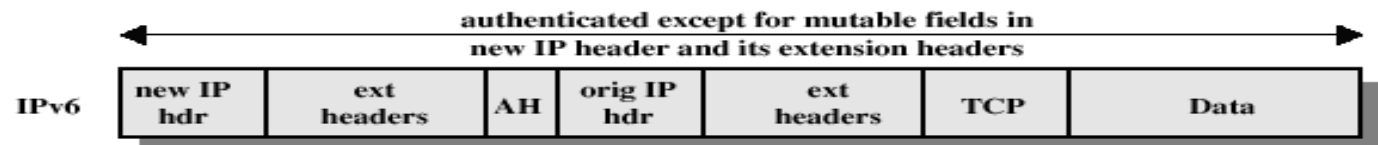
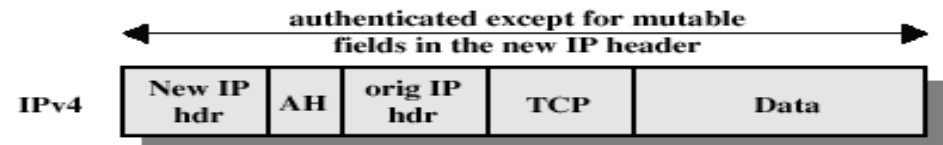
- Original IP packets



- Transport Mode AH
 - Host-to-host authentication

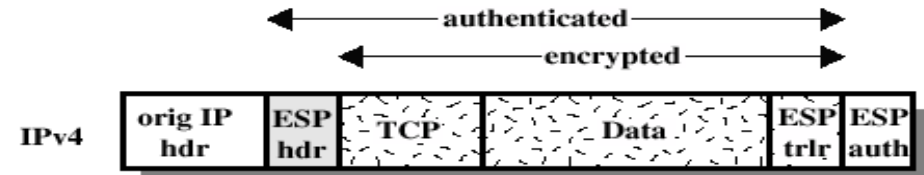


- Tunnel Mode AH
 - Host-to-host
 - Host-to-router (i.e. remote access)
 - Router-to-router



Encapsulating Security Payload (ESP) Protocol

- Transport Mode ESP

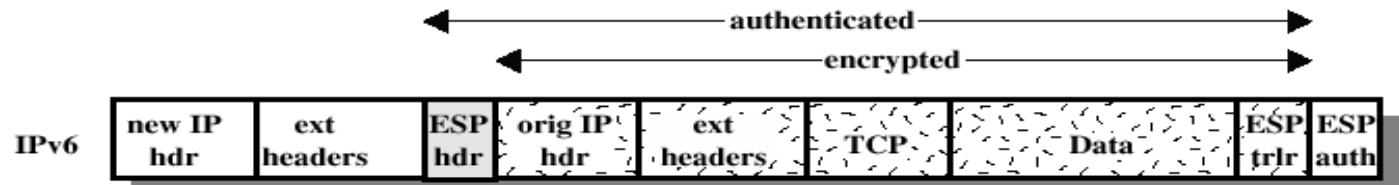
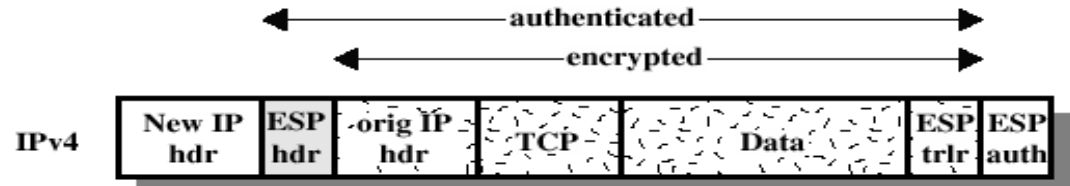


IPv6



(a) Transport Mode

- Tunnel Mode ESP



(b) Tunnel Mode



IPsec : Key Management

- Manual
 - Keys configured on each system
- Automatic: IKE (Internet Key Exchange, RFC 2409)
 - Security Association negotiation: ISAKMP (Internet Security Association and Key Management Protocol, RFC 2408)
 - Different blocs (payloads) are chained together after ISAKMP header
 - Key Exchange Protocols: Oakley, Scheme
 - IKEv2: much simpler (work in progress)
- Algorithms: Authentication and Encryption



Key Management: Requirements

- AH and ESP require encryption and authentication keys
- Process to negotiate and establish IPSec SA's between two entities



Key management: Concepts

- PFS: Perfect Forward Secrecy
 - Obtaining one key does not give access to all data, only data protected by that one key
 - Keys not derived from predecessors
- Nonces: locally generated pseudorandom numbers

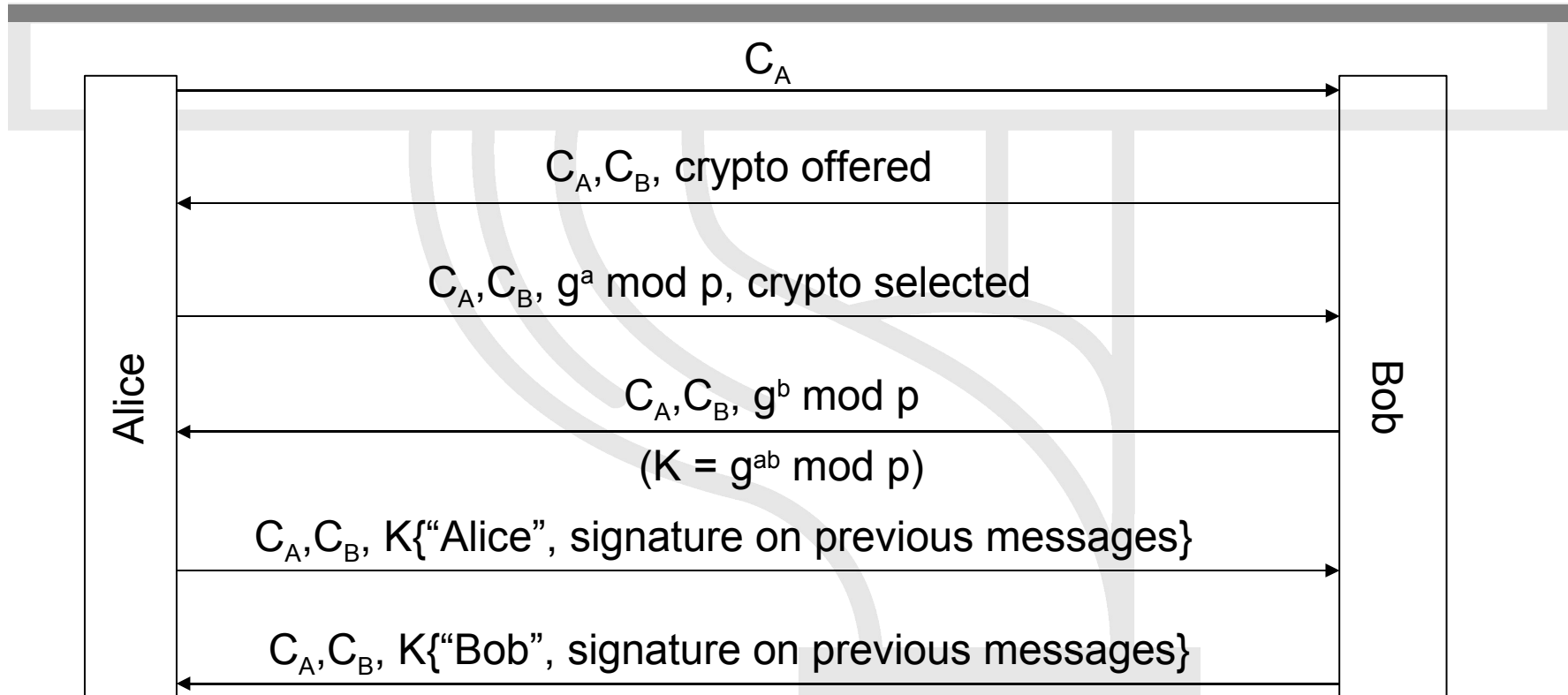


History of IKE

- Early proposals:
 - Photuris: Authenticated DH with cookies & identity hiding
 - SKIP: Auth. DH with long-term exponents
- ISAKMP RFC 2408 – general framework for SAs and key-exchange:
 - A protocol specifying only payload formats & exchanges (i.e., an empty protocol)
 - Adopted by the IPsec working group with RFC 2407 (IPSec Domain of Interpretation)
- Oakley RFC 2412 – key determination protocol: Modified Photuris; can work with ISAKMP
- SKEME – versatile secure key exchange – fast re-keying with Nonces
- IKE: A particular Oakley-ISAKMP combination +fast re-keying



Photuris



C_A : Alice's cookie; for connection ID

C_B : Bob's cookie; against DoS



Photuris – Features

- DoS protection by cookies
(note: C_B can be stateless)
- Authentication & integrity protection of the messages by a combined signature at the last rounds
- Identity hiding from passive attackers



IKE Phases

- Phase 1: Two peers authenticate each other and set up a secure channel for subsequent communications: Authenticated DH, establishes session key & “ISAKMP SA”
 - Main Mode
 - Aggressive Mode
 - The differences between them are the number of message flows needed and the services they provide.
- Phase 2: The two peers negotiate various parameters for IPSec. They include the base protocol, encapsulation mode, keying materials, etc. The end result is going to be one or more SAs. Messages encrypted & authenticated with Phase 1 keys
 - Quick Mode



Concepts - Cookies

- Requirements
 - Depend on specific parties
 - Only the issuing entity can generate acceptable cookies – implies issuer using local secret
 - Cookie generation and verification must be fast
- Hash over IP Src/Dest; UDP Src/Dest; local secret



IKE Phase 1: Main Mode

- Purposes
 - Authenticated key exchange for establishing the IKE SA.
 - Protect the identities of the two parties.
- Four keys (secret information) are to be created after phase 1:
 1. SKEYID : This value will be used to create the other three secret values. (prf=pseudo random function):
 - For signatures: $SKEYID = \text{prf}(N_i | N_r, g^{xy})$
 - For public key encryption: $SKEYID = \text{prf}(\text{hash}(N_i | N_r), CK_i | CK_R)$
 - For pre-shared keys: $SKEYID = \text{prf}(\text{pre-shared-key}, N_i | N_r)$
 2. SKEYID_d: Used to derive keying material for IPSec protocols.
 $SKEYID_d = \text{prf}(SKEYID, K | CK_i | CK_R | 0)$ - K is the secret generated by DH
 3. SKEYID_a: Used to derive keys for authentication and data integrity.
 $SKEYID_a = \text{prf}(SKEYID, SKEYID_d | K | CK_i | CK_R | 1)$
 4. SKEYID_e: Used to derive keys for confidentiality.
 $SKEYID_e = \text{prf}(SKEYID, SKEYID_a | K | CK_i | CK_R | 2)$

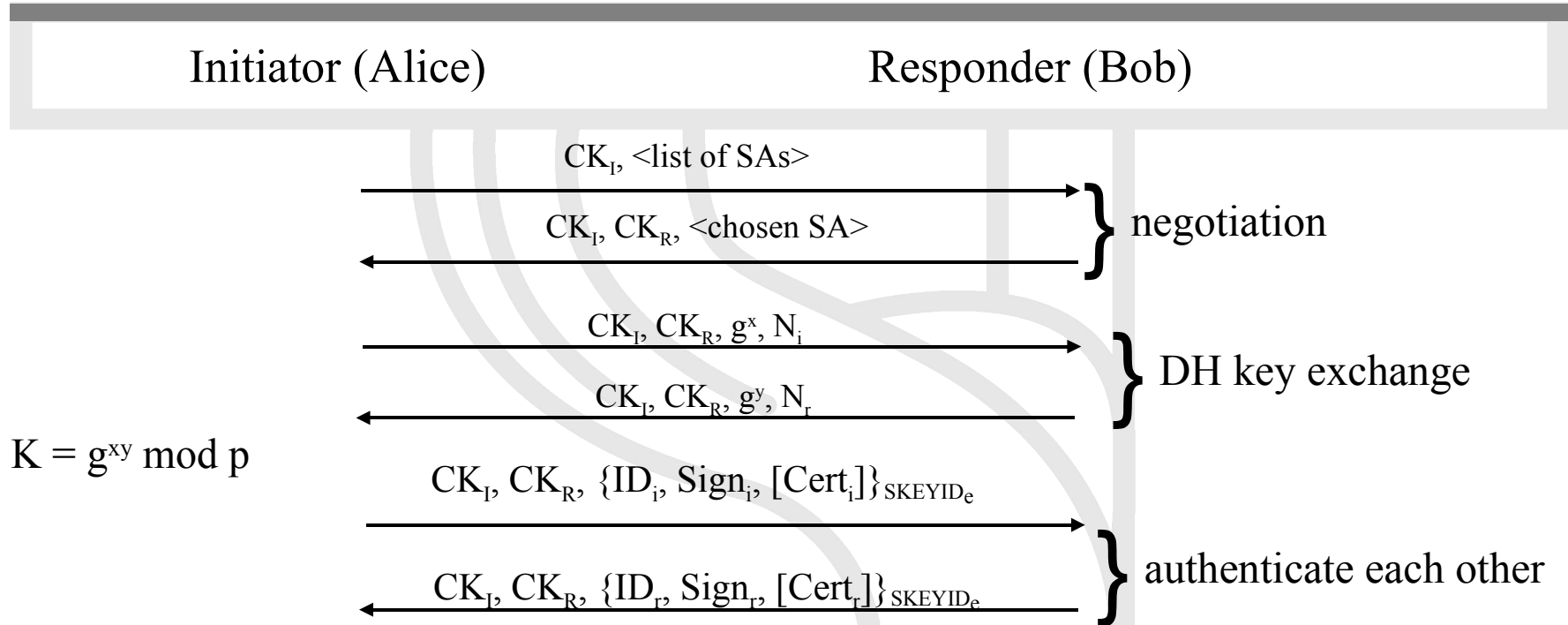


IKE Phase 1: Main Mode

- The Main Mode consists of six message flows (i.e. three rounds).
 - First round: security parameters negotiation
 - Second round: key exchange
 - Third round: mutual authentication
- The following values are used for authentication:
 - $\text{HASH}_i = \text{prf}(\text{SKEYID}, g^x | g^y | \text{CK}_i | \text{CK}_R | \langle \text{list of SAs} \rangle | \text{ID}_i)$
 - This is to be the response from the initiator.
 - This value or its signature will be transmitted.
 - $\text{HASH}_R = \text{prf}(\text{SKEYID}, g^x | g^y | \text{CK}_i | \text{CK}_R | \langle \text{list of SAs} \rangle | \text{ID}_R)$
 - This is to be the response from the responder.



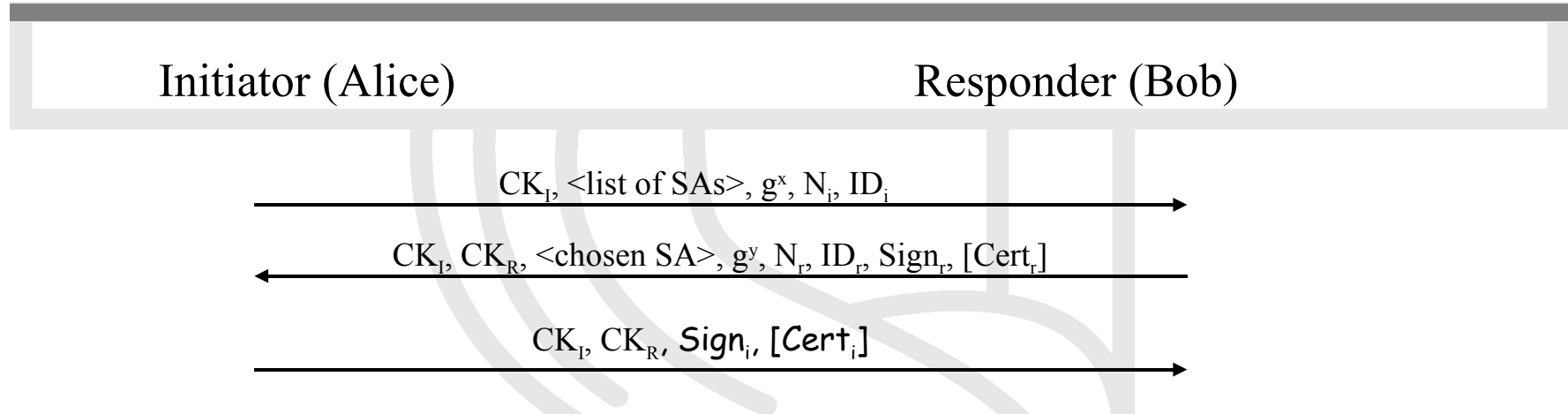
IKE Phase 1: Authentication Using Signatures (Main Mode)



- Depending on the signature scheme selected, $Sign_i$ or $Sign_r$ is the corresponding signature of $HASH_i$ or $HASH_r$ respectively.
- Identities are protected using symmetric key encryption.
- Certificates are optional.



IKE Phase 1: Authentication Using Signatures (Aggressive Mode)



- Only three message flows
- No identity protection
- Open to clogging DoS, doesn't check cookie before DH work

Other authentication methods defined for IKE Phase 1:

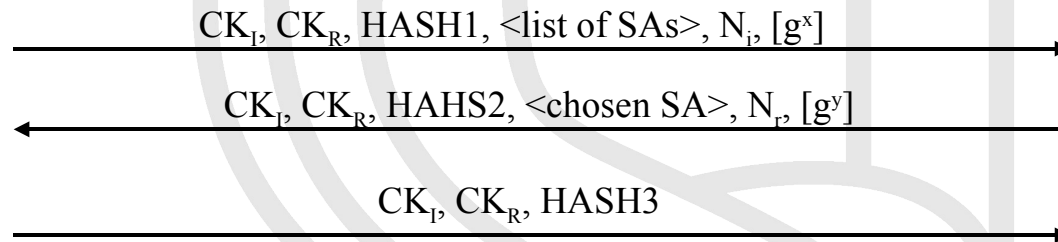
- Authentication using public key encryption
- Authentication using pre-shared keys



IKE Phase 2 (Quick Mode)

Initiator (Alice)

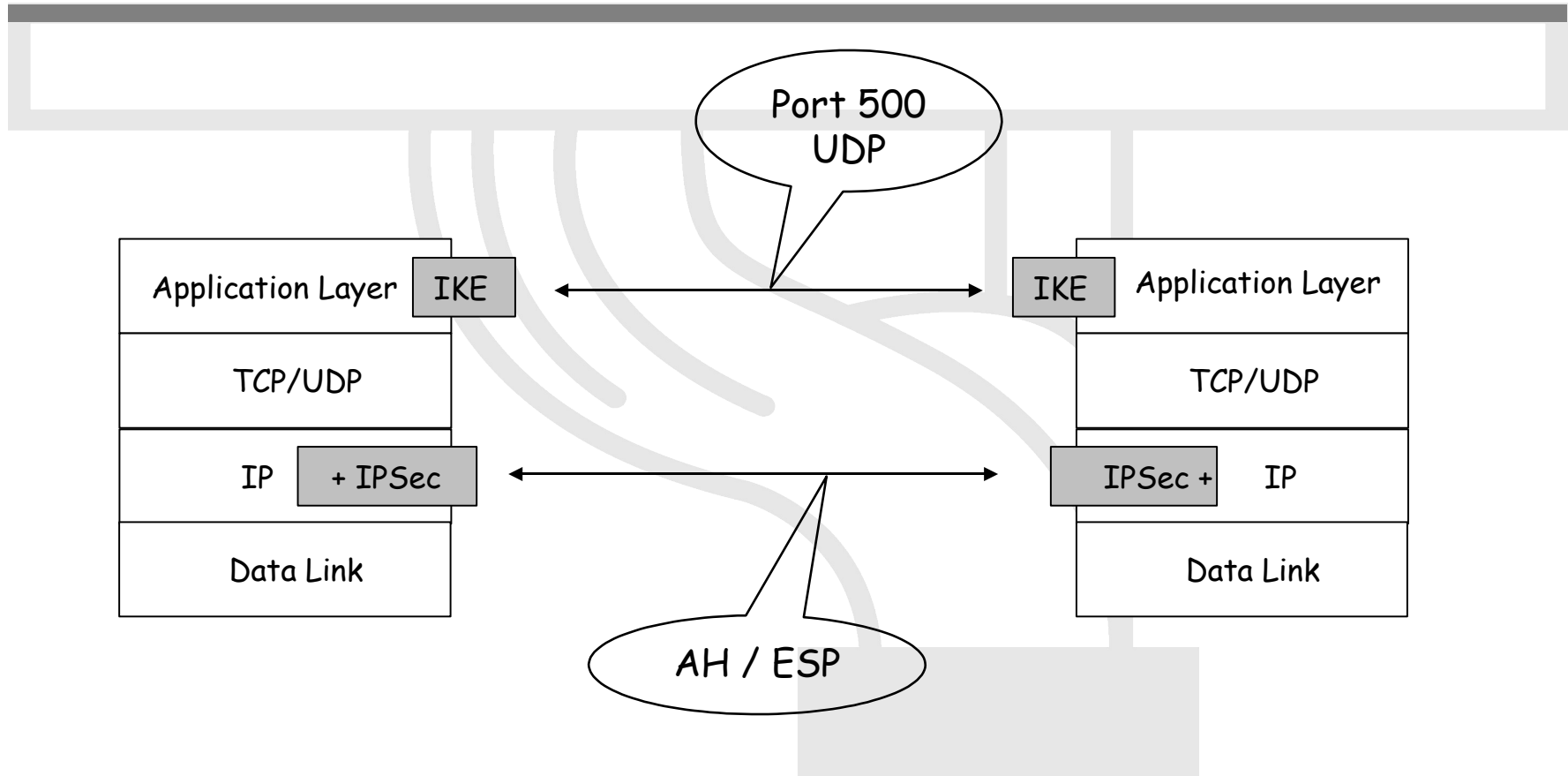
Responder (Bob)



- $HASH1 = \text{prf}(\text{SKEYID}_a, \langle \text{list of SAs} \rangle \mid N_i \mid [g^x])$
- $HASH2 = \text{prf}(\text{SKEYID}_a, N_r \mid \langle \text{chosen SA} \rangle \mid N_r \mid [g^y])$
- $HASH3 = \text{prf}(\text{SKEYID}_a, 0 \mid N_i \mid N_r)$
- The **optional** Diffie-Hellman key exchange is for Perfect Forward Secrecy (PFS).



IKE – layout



IKEv2 Protocol

- Initiated by Perlman & Kaufman, with the aims of
 - simplifying IKEv1
 - fixing the bugs
 - fixing the ambiguities
 - and, at the same time, remaining as close as possible to IKEv1. (“no gratuitous changes”)

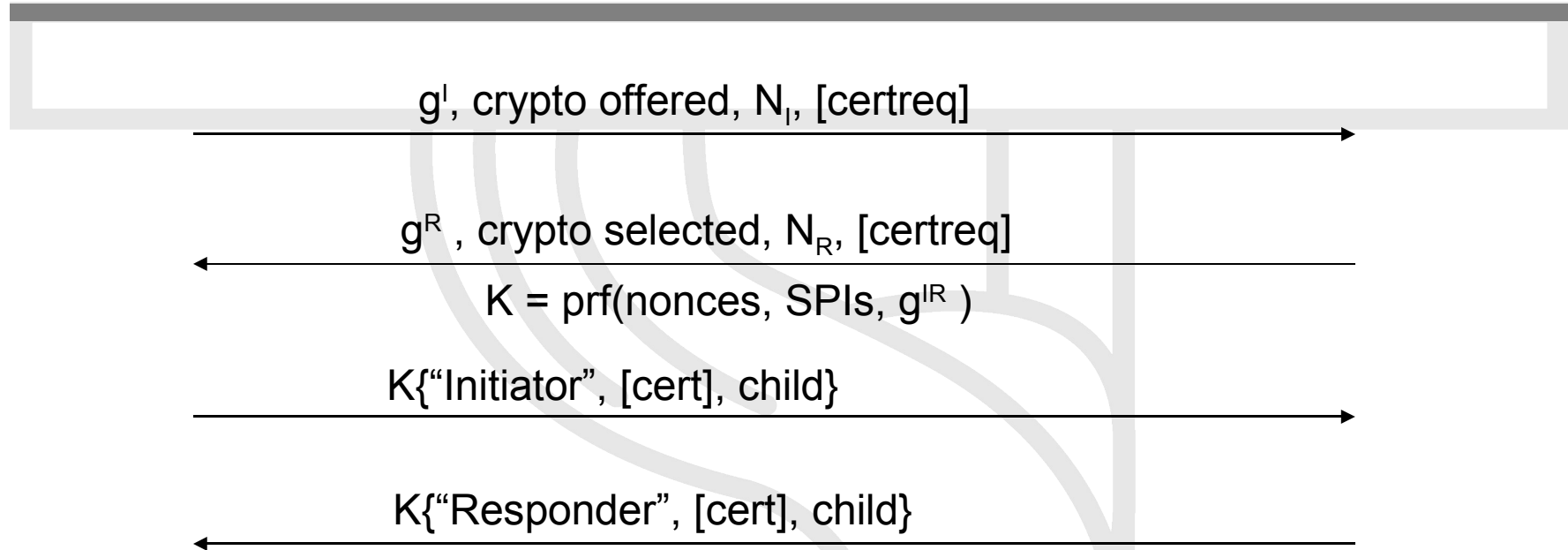


IKEv2 Main Features

- Only one mode of authentication: Public key signatures.
- IKE SA + IPsec SA are established in the same protocol, in 4 messages. (~ Phase 1)
- Additional child SAs, if needed, are established in 2 messages. (~ Phase 2)
- DoS protection optional, via cookies (stateless).
- Crypto negotiation is simplified
 - support for “suites”
 - ability to say “any of these enc., with any of these hash...”



The Exchange Protocol (overview)



- Responder can optionally refuse the first message and require return of a cookie.
- Adds extra 2 messages.

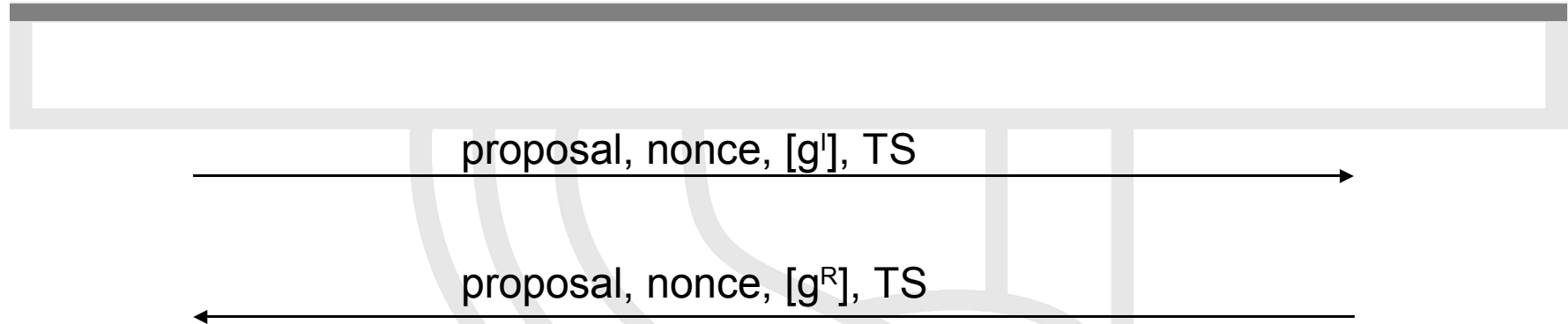


The Exchange Protocol (cont'd)

- DoS protection: Optional; by Responder responding the first message with a (stateless) cookie.
- Originally, designed with 3 rounds. Later 4 rounds is agreed on:
 - Initiator needs a 4th message anyway to know when to start the transmission.
 - Extra msgs for cookie exchange can be incorporated into 4 msgs, if Initiator repeats msg.1 info in msg.3
- Preserves identity hiding from passive attackers.



Child-SA Creation



- proposal: crypto suites, SPI, protocol (ESP, AH, IP compression)
- TS: Traffic selector
- Derived keys: Function of IKE keying material, nonces of this exchange, plus optional DH output.



Other IKEv2 Features

- Reliability:
 - All messages are request/response.
 - Initiator is responsible for retransmission if it doesn't receive a response.
- Traffic selector negotiation:
 - IKEv1: Responder can just say yes/no.
 - IKEv2: Negotiation ability added.
- Rekeying:
 - Either side can rekey at any time.
 - Rekeyed IKE-SA inherits all the child-SAs.





IPv6 Security infrastructure

Firewalls

See earlier and the references



Summary

- IPv6 has potential to be a foundation of a more secure Internet
- Elements of the IPv6 security infrastructure (Firewalls, IPSec, AAA etc.) are mature enough to be deployed in production environment.



References

- 6NET D3.5.1: Secure IPv6 Operation: Lessons learned from 6NET
- J. Mohacsi, “IPv6 firewalls”, presentation on the 5th TF-NGN meeting, October 2001 available at http://skye.ki.iif.hu/~mohacsi/athens_tf_ngn_ipv6_firewalls.pdf
- J. Mohacsi, “Security of IPv6 from firewalls point of view”, presentation on TNC2004 conference, June 2004, available at http://www.terena.nl/conferences/tnc2004/programme/presentations/show.php?pres_id=115
- 6NET D6.2.2: Operational procedures for secured management with transition mechanisms
- S. Convery, D Miller, IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation (v1.0)", presentation at the 17th NANOG, May 24, 2004
- J. Mohácsi, E. Davis: draft-v6ops-ietf-icmpv6-filtering-bcp-00.txt



Thank you!

- Acknowledgement to Patrick Grossetete, Stig Veenas, Ladislav Lhotka, Jerome Durand, Tim Chown, Gunter van de Velde and Eric Marin for their comments.
- Questions: mohacsi@niif.hu





Configuring IPv6 Firewalls with pf

János Mohácsi
NIIF/HUNGARNET



What is pf?

- OpenBSD included IPFilter in the default install since 3.0. Included in FreeBSD since 5.3 (5.x as port) and in NetBSD since 2.0.
- Ideas from ipf which is available for Linux, Solaris, HP-UX, IRIX additionally to the operating systems above.
- Principles
 - working on IP packet level (vs. application level proxies or ethernet level bridges)
 - intercepting each IP packet that passes through the kernel (in and out on each interface), passing or blocking it
 - stateless inspection based on fields of each packet
 - stateful filtering keeping track of connections, additional information makes filtering more powerful (sequence number checks) and easier (replies, random client ports)
 - filtering for local host or network (multihomed host, IP forwarding or bridging)



IPv6 specific rules -reminder

- Neighbor solicitation/neighbor advertisement is **REQUIRED** – icmp-type 135/136 (ipv6-icmp-type neighborsol/neighboradv)
- For stateless address autoconfiguration router advertisement and router solicitation **REQUIRED**– icmp type 133/134 (ipv6-icmp-type routersol/routeradv)
- Path MTU discovery – automatic if you use keep-state
 - Same applies for Destination unreachable, Time exceeded and IPv6 Parameter problem messages
 - Otherwise you have to build your own rules



Ruleset IPv6 1 (nothing in, DNS, all TCP, ping out)

```
EXT = "bge0"
LAN = "bge1"
LANip6 = "2001:db8:1:1::1"
EXTip6 = "2001:db8:1:2::1"
LANnet6 = "2001:db8:1:1::1/64"
Lo6 = "::1"
# expire state connections early
set optimization aggressive
block in log all
# allow DNS requests to go out
pass out on $EXT inet6 proto udp from {$EXTip6, $Lo6, $LANnet6} to any port=domain keep state
# all TCP request allowed out
pass out on $EXT inet6 proto tcp from {EXTip6, $Lo6, $LANnet6} to any keep state
# all ping request allowed out
pass out on $EXT inet6 proto icmp6 all icmp6-type echoreq keep state
# ND solicitation out
pass out on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighborsol}
# ND advertisement in
pass in on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighborsol}
```



Ruleset IPv6 1(continue – with router advertisement from FW)

```
#router advertisement out
pass out on $LAN inet6 proto icmp6 all icmp6-type routersadv
# router solicitation in
pass in on $LAN inet6 proto icmp6 all icmp6-type routerrsol
# DNS request inside
pass in on $LAN inet6 proto from $LANnet6 to any port domain
# TCP request inside
pass in on $LAN inet6 proto tcp from $LANnet6 to any
# ICMP request inside
pass in on $LAN inet6 proto icmp6 all icmp6-type echoreq
```



Ruleset IPv6 2 (allow access to internal mail & www server – additional rules)

```
#internal server address
LANSRV6="2001:db8:1:2::2"
LANSRV4="192.168.1.2"
#allow incoming connection to SMTP server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=25 keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=25 keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=25 to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=25 to any keep-state
#allow incoming connection to WWW server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=www keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=www keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=www to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=www to any keep-state
```



Problems with IPv6

- No fragment normalisation – not possible! – fragmentation only at the end-host
- no real support for extension headers – check existence of extension header possible without chain processing
- IPv6 fragments are blocked unconditionally
- No IPv6 support in ftp-proxy
- No support to filter inside tunnel – except if tunnel terminated at firewall



Tunnel filtering example - simplest

```
pass out on $EXT inet proto ipv6 from $EXT  
to $TUNNELREMOTE4 keep state
```

```
pass in on $EXT inet proto ipv6 from  
$TUNNELREMOTE4 to $ext_if keep state
```

```
pass out on gif0 inet6 all keep state
```

```
pass in on gif0 inet6 all keep state
```



More restrictive IPv6 rules

- Allow Rtsol/rtadv on a more specific address

```
pass out on $LAN inet6 proto ipv6-icmp from fe80::/16 to  
ff02::2 icmp6-type routersol code 0
```

```
pass in on $LAN inet6 proto ipv6-icmp from fe80::/16 to  
ff02::1 icmp6-type routeradv code 0
```

```
pass in on $LAN inet6 proto ipv6-icmp from fe80::/16 to  
fe80::/16 icmp6-type routeradv code 0
```

- Allow NDsol/Ndadv on more specific address

```
pass out on $if inet6 proto ipv6-icmp from { :: fe80::/16 }  
to ff02::/16 icmp6-type grouprep code 0
```

```
pass out on $if inet6 proto ipv6-icmp from ($if) to any  
icmp6-type neighborsol code 0
```

```
pass in on $if inet6 proto ipv6-icmp from any to ($if)  
icmp6-type neighboradv code 0
```



FreeBSD ip6fw Packet Filtering

- Native IPv6 packet filtering interface since FreeBSD4 – without state keeping
- Implemented as a multifunction user command
- In the kernel it requires:
 - IPV6FIREWALL – enable, IPV6FIREWALL_VERBOSE – logging, PFIL_HOOKs required in the case of FreeBSD5
- The packet passed to the compared firewall is against each of the rules in the firewall ruleset.
- When a match is found, the action corresponding to the matching rule is performed and the search terminates.
- rule match is updating the rule counters: packet count, byte count
- General syntax
 - ip6fw [rule number] action [log] proto from src to dst



ip6fw actions

- allow | accept | pass | permit
 - Allow packets that match rule. The search ends.
- deny | drop
 - Discard packets that match rule. The search ends.
- unreachable code
 - Discard packets that match rule and send ICMPv6 unreachable: admin, notneighbor, addr, noroute, noport
- reset
 - Send TCP reset to initiator
- count
 - Update counters for all packets that match rule. The search continues with the next rule



ip6fw options

- proto
 - all | ipv6
 - All packets match.
 - tcp/udp
 - Only tcp/udp packets match.
 - ipv6-icmp
 - Only ICMPv6 packets match.
- src and dst
 - <address/prefixlend> [ports]
- options
 - frag - non first packets of fragmented packets
 - in/out – way in/way out
 - ipv6options hopopt/route/frag/esp/ah/nonxt/opts
 - icmptypes



ip6fw examples

- Allow DAD
 - ip6fw add pass ipv6-icmp from :: to ff02::/16
- Allow RA,RS, NS, NA and redirect
 - ip6fw add pass ipv6-icmp from fe80::/10 to fe80::/10
 - ip6fw add pass ipv6-icmp from fe80::/10 to ff02::/16
- Allow link-local multicast traffic
 - ip6fw add pass all from fe80::/10 to ff02::/16
 - ip6fw add pass all from $\{\text{net}\}/\{\text{prefixlen}\}$ to ff02::/16



ip6fw examples/2

- Allow ICMPv6 destination unreachable
 - ip6fw add pass ipv6-icmp from any to any icmptype 1
- Allow PATH-MTU – do not filter!
 - ip6fw add pass ipv6-icmp from any to any icmptype 2
- Allow NS/NA - do not filter!
 - ip6fw add pass ipv6-icmp from any to any icmptype 135,136

