Practical Network Defense

Master's degree in Cybersecurity 2024-25

IPsec lab

Angelo Spognardi
spognardi@di.uniroma1.it
Dipartimento di Informatica
Sapienza Università di Roma

Aim of the lab

1) Realize a IPsec configuration
2) Transport mode
3) Tunnel mode

Credits:
Part of the slides are taken from the Network Security (NetSec) course IN2101 – WS 19/20 of Technical University of Munich:
http://netsec.net.in.tum.de/slides/12_ipsec.pdf

IPsec (RFC 4301)

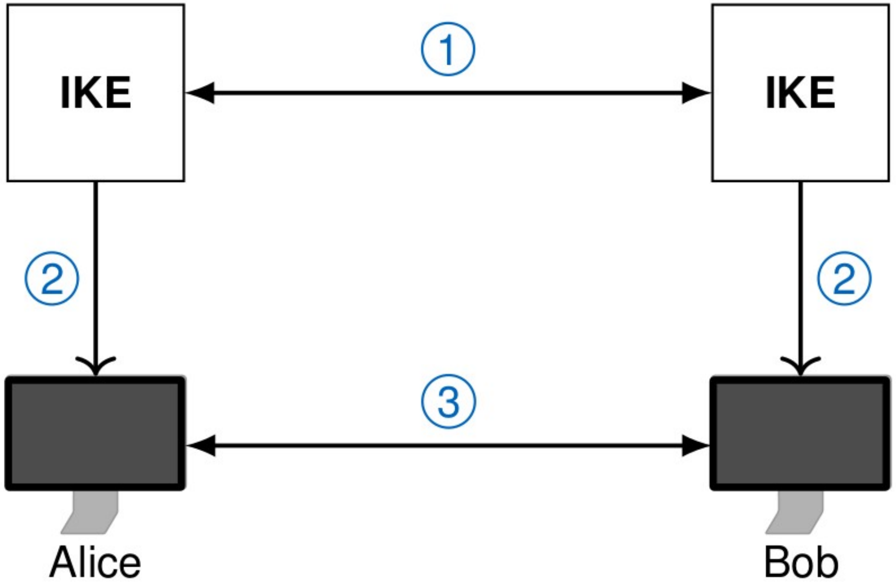
- A Network Layer protocol suite for providing security over IP.
- Part of IPv6; an add-on for IPv4.
- Can handle all three possible security architectures:

Feature	Gateway-to-Gateway	Host-to-Gateway	Host-to-Host
Protection between client and local gateway	No	N/A (client is VPN endpoint)	N/A (client is VPN endpoint)
Protection between VPN endpoints	Yes	Yes	Yes
Protection between remote gateway and remote server (behind gateway)	No	No	N/A (client is VPN endpoint)
Transparency to users	Yes	No	No
Transparency to users' systems	Yes	No	No
Transparency to servers	Yes	Yes	No

Fundamentals of IPsec

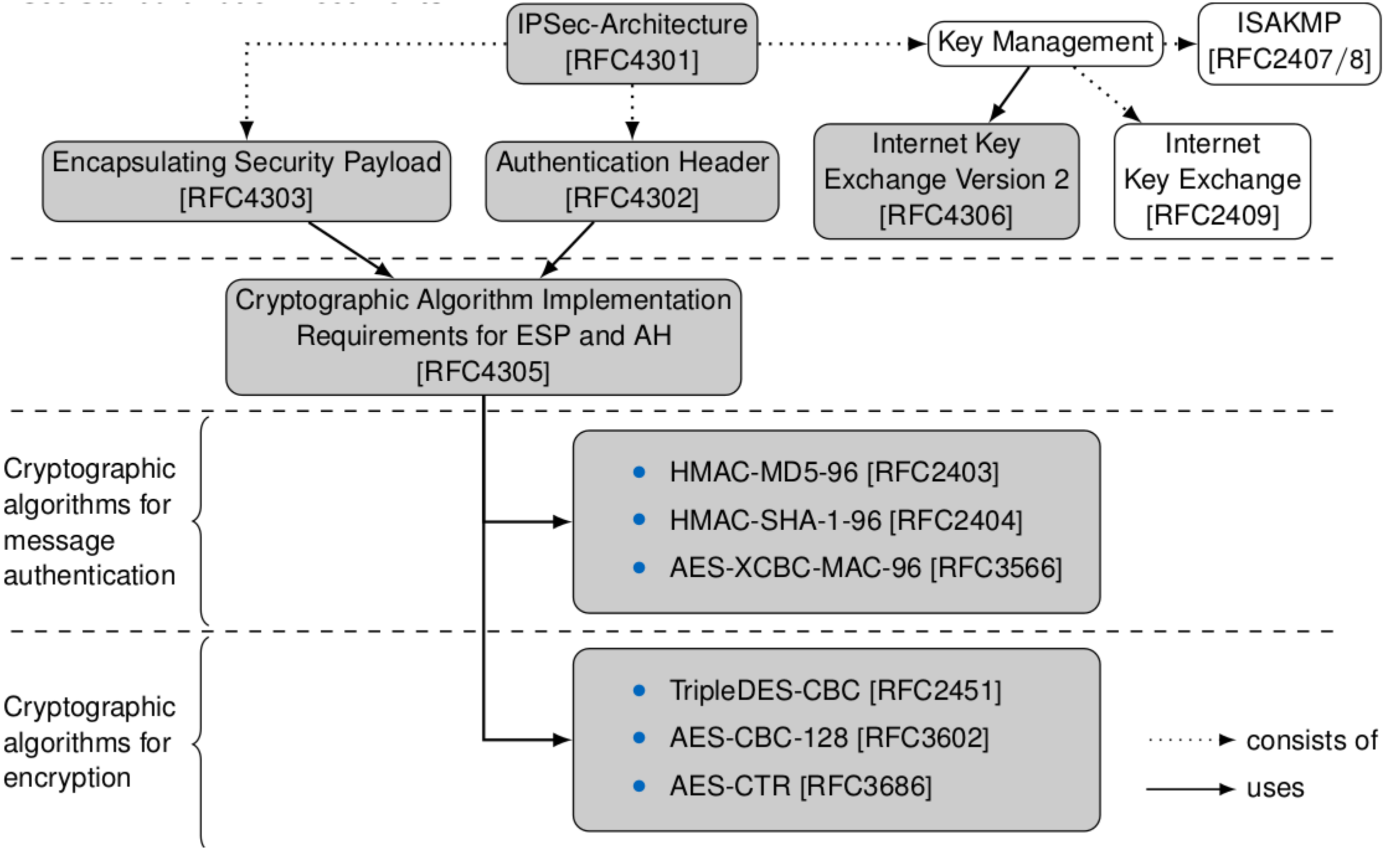
- Data origin authentication
  - It is not possible to spoof source / destination addresses without the receiver being able to detect this
  - It is not possible to replay a recorded IP packet without the receiver being able to detect this
- Connectionless Data Integrity
  - The receiver is able to detect any modification of IP datagrams in transit
- Confidentiality
  - It is not possible to eavesdrop on the content of IP datagrams
  - Limited traffic flow confidentiality
- Security Policies
  - All involved nodes can determine the required protection for a packet
  - Intermediate nodes and the receiver will drop packets not meeting these requirements

IPsec overview



1) Authentication, key exchange and negotiation of crypto algorithms
   - Manual
   - Automated: ISAKMP, Internet Key Exchange (IKE), IKEv2
2) Set up of key and crypto-algorithms
3) Use of the secure channel, with:
   - Data Integrity via Authentication Header (AH) or Encapsulating Security Payload (ESP)
   - Confidentiality using ESP
     - ESP can provide both data integrity and encryption while AH only provides data integrity

IPsec Standardization Documents



List of IPsec related RFCs: https://datatracker.ietf.org/wg/ipsec/documents/

IPsec architecture (RFC 4301)

- Concepts
  - Security Association (SA) and Security Association Database (SAD)
  - Security Policy (SP) and Security Policy Database (SPD)
- Fundamental Protocols
  - Authentication Header (AH)
  - Encapsulation Security Payload (ESP)
- Protocol Modes
  - Transport Mode
  - Tunnel Mode
- Key Management Protocols
  - ISAKMP, IKE, IKEv2

IPsec services

- Basic functions, provided by separate (sub-)protocols:
  - Authentication Header (AH): Support for data integrity and authentication of IP packets.
  - Encapsulated Security Payload (ESP): Support for encryption and (optionally) authentication.
  - Internet Key Exchange (IKE): Support for key management etc.

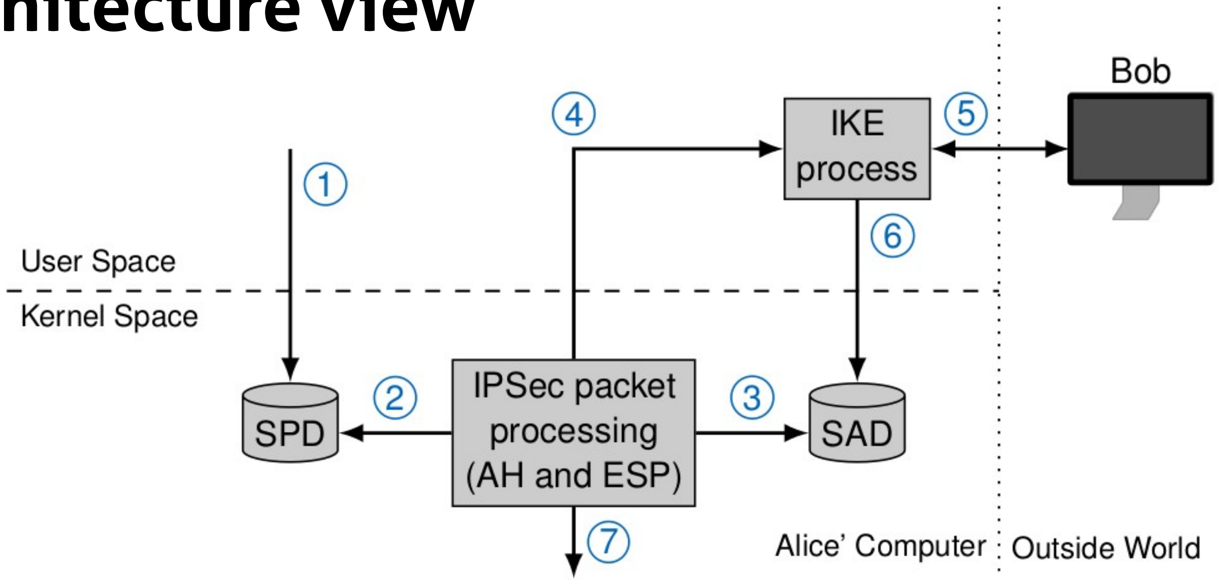
Service	AH	ESP (encrypt only)	ESP(encrypt+authent.)
Access Control	+	+	+
Connectionless integrity	+		+
Protection between VPN endpoints	+		+
Data origin authentication	+		+
Reject replayed packets		+	+
Payload confidentiality		+	+
Metadata confidentiality		partial	partial
Traffic flow confidentiality		(*)	(*)

IPsec modes

- Transport Mode
  - Provides protection for a T-layer packet embedded as payload in an IP packet.
- Tunnel Mode
  - Provides protection for an IP packet embedded as payload in an IP packet.

	Transport Mode SA	Tunnel Mode SA
AH	Authenticate IP payload and selected parts of IP header and IPv6 extension headers.	Authenticate entire inner IP packet and selected parts of outer IP header and outer IPv6 extension headers.
ESP	Encrypt IP payload + any IPv6 extension headers after ESP header.	Encrypt inner IP packet.
ESP + authent.	Encrypt IP payload + any IPv6 extension headers after ESP header. Authenticate IP payload.	Encrypt and authenticate inner IP packet.

IPsec architecture view



1. The administrator sets a policy in SPD
2. The IPsec processing module refers to the SPD to decide on applying IPsec on packet
3. If IPsec is required, then the IPsec module looks for the IPsec SA in the SAD
4. If there is no SA yet, the IPsec module sends a request to the IKE process to create an SA
5. The IKE process negotiates keys and crypto algorithms with the peer host using the IKE/IKEv2 protocol
6. The IKE process writes the key and all required parameters into the SAD
7. The IPsec module can now send a packet with applied IPsec

Security Policies

- A Security Policy (SP) specifies which security services should be provided to IP packets and their details
  - A SP affects only a specific IP stream
  - For each stream, it includes several security attributes:
    - The security protocol (AH / ESP)
    - The protocol mode (Transport / Tunnel)
    - Other parameters, like policy lifetime, port number for specific applications, etc.
    - The actions (e.g. Discard, Secure, Bypass)
- Security Policies are stored in the Security Policy Database (SPD)

Security Policy example

```
root@r2:/# /usr/sbin/setkey -PD
100.90.0.100[any] 100.60.0.100[any] 255
        out prio def ipsec
        esp/transport//require
        created: May 10 13:42:22 2022  lastused: May 10 13:43:08 2022
        lifetime: 0(s) validtime: 0(s)
        spid=441 seq=1 pid=546
        refcnt=1
100.60.0.100[any] 100.90.0.100[any] 255
        fwd prio def ipsec
        esp/transport//require
        created: May 10 13:42:22 2022  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=434 seq=2 pid=546
        refcnt=1
100.60.0.100[any] 100.90.0.100[any] 255
        in prio def ipsec
        esp/transport//require
        created: May 10 13:42:22 2022  lastused: May 10 13:43:08 2022
        lifetime: 0(s) validtime: 0(s)
        spid=424 seq=0 pid=546
        refcnt=1
root@r2:/#
```

Security Policy actions

- Discard: reject to send/receive the packet
- Bypass (none): do not handle with IPsec → send in clear
- Secure (ipsec): handle with IPsec
  - In this case, the security policy is in the form `protocol/mode/src-dst/level` where:
    - Protocol → **ah**, **esp** or **ipcomp** (for payload compression)
    - Mode → **tunnel** or **transport**
    - Src-dst → endpoints of the tunnel (if needed)
    - Level → **default**, **use**, **require**, or **unique**
      - This specifies the level of the SA, when a keying daemon is used

Security Associations

- A Security Association (SA) is a simplex channel that describes the way how packets need to be processed
  - A SA specifies encryption/authentication algorithms and keys
  - A SA is associated with either AH or ESP (not both)
- Bidirectional communication requires two security associations
- SAs can be setup as
  - Host ↔️ Host
  - Host ↔️ Gateway
  - Gateway ↔️ Gateway
- Security Associations are stored in the Security Association Database (SAD)

Security Association Database (SAD)

- An entry (SA) is uniquely identified by a Security Parameter Index (SPI)
  - The SPI value for construction of AH/ESP headers is looked up for outbound SAs
  - The SPI value is used to map the traffic to the appropriate SA for inbound traffic
- An SA entry in the SAD includes
  - Security Parameter Index (SPI)
  - Source/destination IP addresses
  - Identified protocol (AH / ESP)
  - IPsec protocol mode (tunnel / transport)
  - Protocol algorithms, modes, IVs and keys
  - Security Association Lifetime
  - Current sequence number counter (replay protection)
  - Other pieces of information (see RFC4301, Section 4.4.2.1)

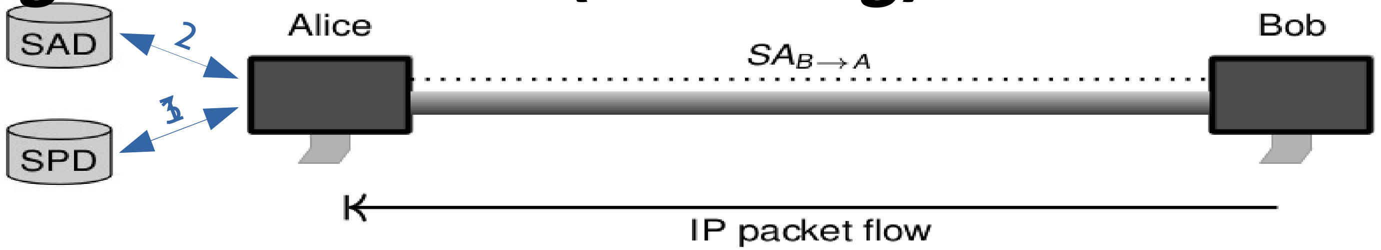
Security Association example

```
root@r2:/# /usr/sbin/setkey -D
100.60.0.100 100.90.0.100
        esp mode=transport spi=801(0x00000321) reqid=0(0x00000000)
        E: aes-cbc  32eac750 a8ab4acd d922c085 356799ce efe3768e f1979fa5 27b1bcb4 33568e52
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: May 10 13:42:22 2022   current: May 10 13:45:07 2022
        diff: 165(s)    hard: 0(s)      soft: 0(s)
        last: May 10 13:43:02 2022      hard: 0(s)      soft: 0(s)
        current: 448(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 7    hard: 0 soft: 0
        sadb_seq=1 pid=544 refcnt=0
100.90.0.100 100.60.0.100
        esp mode=transport spi=800(0x00000320) reqid=0(0x00000000)
        E: aes-cbc  4ac06c9c d4a61a6b ade1dcf3 b5c2731b e0c09cd1 2385bb8e be04ffdb 990eb9e0
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: May 10 13:42:22 2022   current: May 10 13:45:07 2022
        diff: 165(s)    hard: 0(s)      soft: 0(s)
        last: May 10 13:42:23 2022      hard: 0(s)      soft: 0(s)
        current: 2816(bytes)    hard: 0(bytes)  soft: 0(bytes)
        allocated: 44   hard: 0 soft: 0
        sadb_seq=0 pid=544 refcnt=0
root@r2:/#
```

Processing of IPsec Traffic (outgoing)

- Alice wants to send data to Bob, then IP layer of Alice has to:
  1) Determine if and how the outgoing packet needs to be secured
     - Perform a lookup in the SPD based on traffic selectors
     - If the policy specifies discard then drop the packet
     - If the policy does not need to be secured, send it
  2) Determine which SA should be applied to the packet
     - If no SA is established perform IKE
     - There may be more than one SA matching the packet (e.g. one for AH, one for ESP)
  3) Look up the determined or freshly created SA in the SAD
  4) Perform the security transforms, specified in the SA
     - This results in the construction of an AH or ESP header
     - Possibly a new (outer) IP header will be created (tunnel mode)
  5) Send the resulting packet

Processing of IPsec Traffic (incoming)

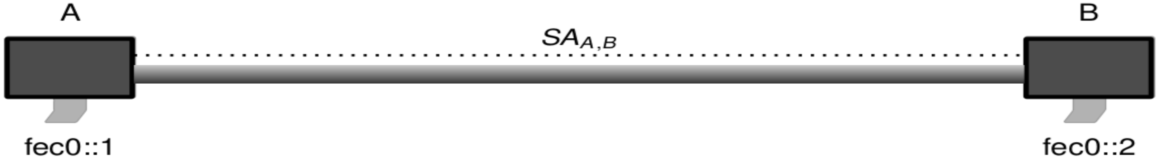


- Alice receives data from Bob, then the IP layer of Alice has to:
  1) If packet contains an IPsec header
     - Perform a lookup in the SPD, if Alice is supposed to process the packet
     - Retrieve the respective policy
  2) If Alice is supposed to process the packet
     - Extract the SPI from the IPsec header, look up the SA in the SAD and perform the appropriate processing
     - If there's no SA referenced by the SPI ⇒ Drop the packet
  3) Determine if and how the packet should have been protected
     - Perform a lookup in the SPD, evaluating the inner IP header in case of tunneled packets
     - If the respective policy specifies discard ⇒Drop the packet
     - If the protection of the packet did not match the policy ⇒Drop the packet
  4) Deliver to the appropriate protocol entity (e.g. network / transport layer)

ipsec-tools (deprecated)

- The framework ipsec-tools included two main tools:
  - **`setkey`**, used to manually manipulate the IPsec SA/SP database
    - default config file: **`/etc/ipsec-tools.conf`**
    - https://manpages.debian.org/testing/ipsec-tools/setkey.8.en.html
  - **`racoon`**, the daemon for the IKE protocol
    - https://manpages.debian.org/testing/racoon/racoon.8.en.html
- The ipsec-tools have been deprecated time ago
  - We see them only for educational purposes

Setup of IPsec Security Policies



- Example:
  IPv6 connection with ESP and Transport Mode

- Configuration at Host A

source IP

dest. IP

upper-layer protocol

this policy is for outgoing packets

ipsec processing rule, as `protocol/mode/src-dst/level`

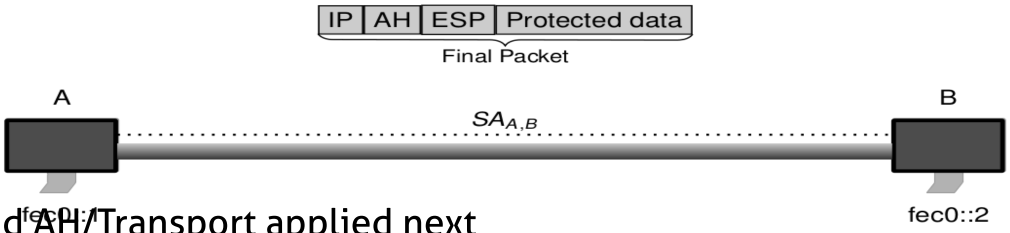
```
spdadd fec0::1 fec0::2 any -P out ipsec esp/transport//require;
spdadd fec0::2 fec0::1 any -P in ipsec esp/transport//require;
```

this policy is for incoming packets

- Configuration at Host B

```
spdadd fec0::2 fec0::1 any -P out ipsec esp/transport//require;
spdadd fec0::1 fec0::2 any -P in ipsec esp/transport//require;
```

Another Security Policy



- Example
  IPv6 connection with ESP/Transport applied first and AH/Transport applied next

- Configuration at Host A

```
spdadd fec0::1 fec0::2 any -P out ipsec
       esp/transport//require ah/transport//require;
```

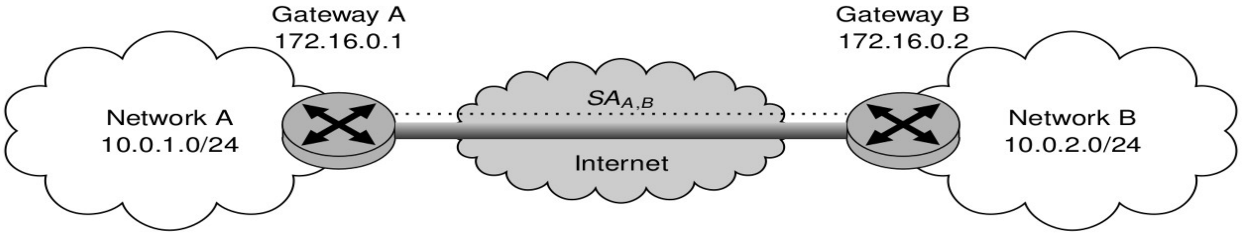
```
spdadd fec0::2 fec0::1 any -P in ipsec
       esp/transport//require ah/transport//require;
```

- Configuration at Host B:

```
spdadd fec0::2 fec0::1 any -P out ipsec
       esp/transport//require ah/transport//require;
```

```
spdadd fec0::1 fec0::2 any -P in ipsec
       esp/transport//require ah/transport//require;
```

Yet another



- Example
  ESP Tunnel for VPN
- Configuration at Gateway A

```
spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
       esp/tunnel/172.16.0.1-172.16.0.2/require;
```

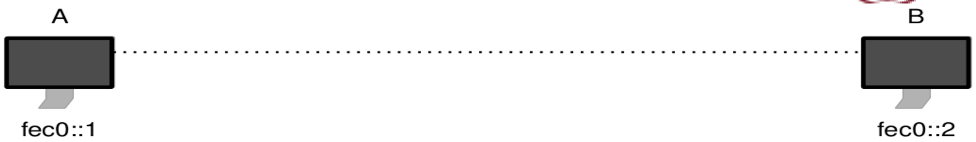
```
spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
       esp/tunnel/172.16.0.2-172.16.0.1/require;
```

- Configuration at Gateway B:

```
spdadd 10.0.2.0/24 10.0.1.0/24 any -P out ipsec
       esp/tunnel/172.16.0.2-172.16.0.1/require;
```

```
spdadd 10.0.1.0/24 10.0.2.0/24 any -P in ipsec
       esp/tunnel/172.16.0.1-172.16.0.2/require;
```

Manual setup of Security Associations



- Example
  Manually setting up an AH SA (use -m tunnel if tunnel mode)

```
# basic syntax: add src dst proto spi -A authalgo key;
```

```
add fec0::1 fec0::2 ah 700 -A hmac-md5
           0xbf9a081e7ebdd4fa824c822ed94f5226;
```

```
add fec0::2 fec0::1 ah 800 -A hmac-md5
           0xbf9a081e7ebdd4fa824c822ed94f5226;
```

- Manually setting up an ESP SA:

```
# basic syntax:  add src dst proto spi -E encalgo key;
```

```
add fec0::1 fec0::2 esp 701 -E 3des-cbc
           0xdafb418410b2ca6a2ba144561fab354640080e5b7ac0c133;
add fec0::2 fec0::1 esp 801 -E 3des-cbc
           0xdafb418410b2ca6a2ba144561fab354640080e5b7ac0c133;
```

Disclamer

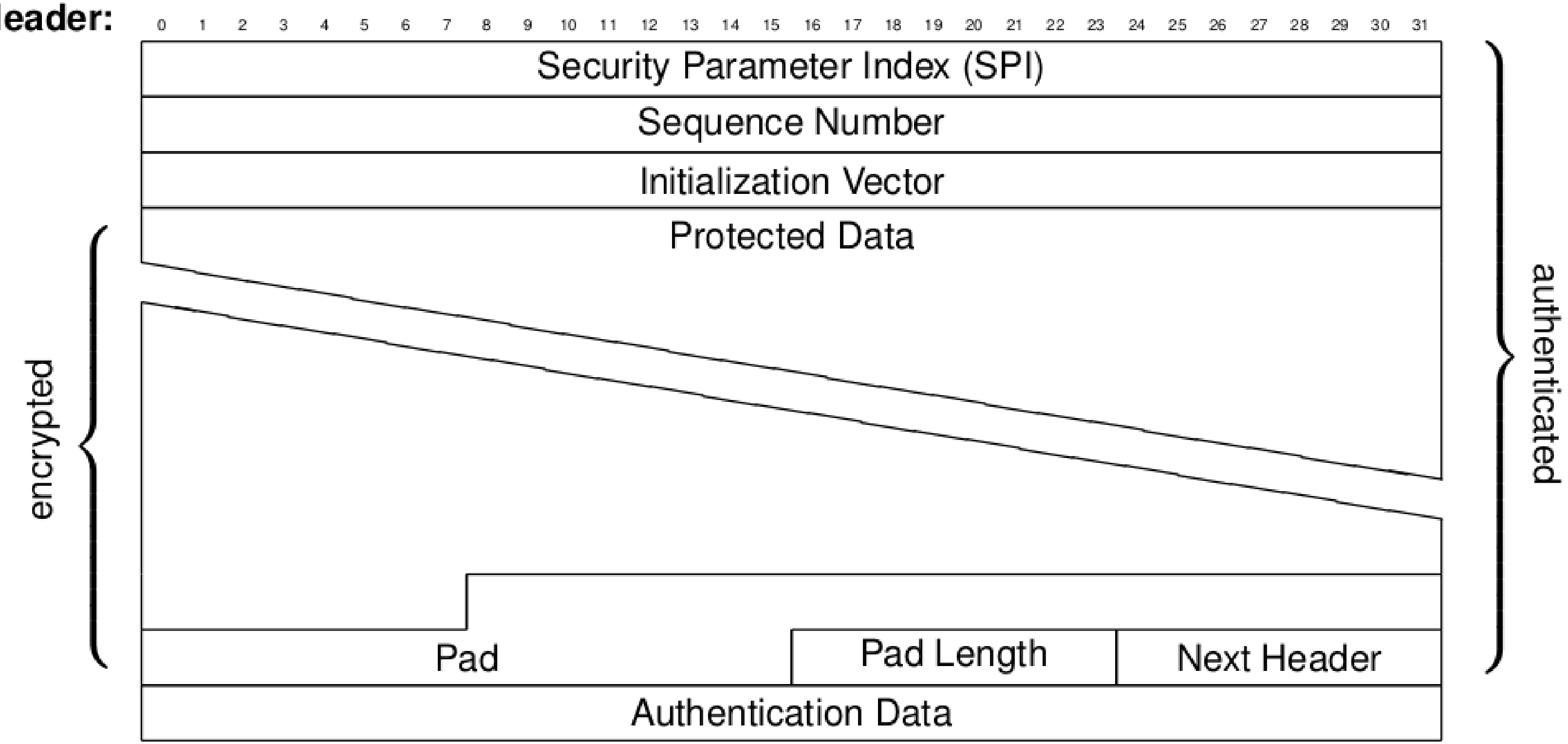
WARNING: Setting up an SA manually is error prone!

- The administrator might choose insecure keys
- The set of SAs might be inconsistent
- It is better to rely on an IKE daemon for setting up SAs

- We do it only for **educational purposes**!

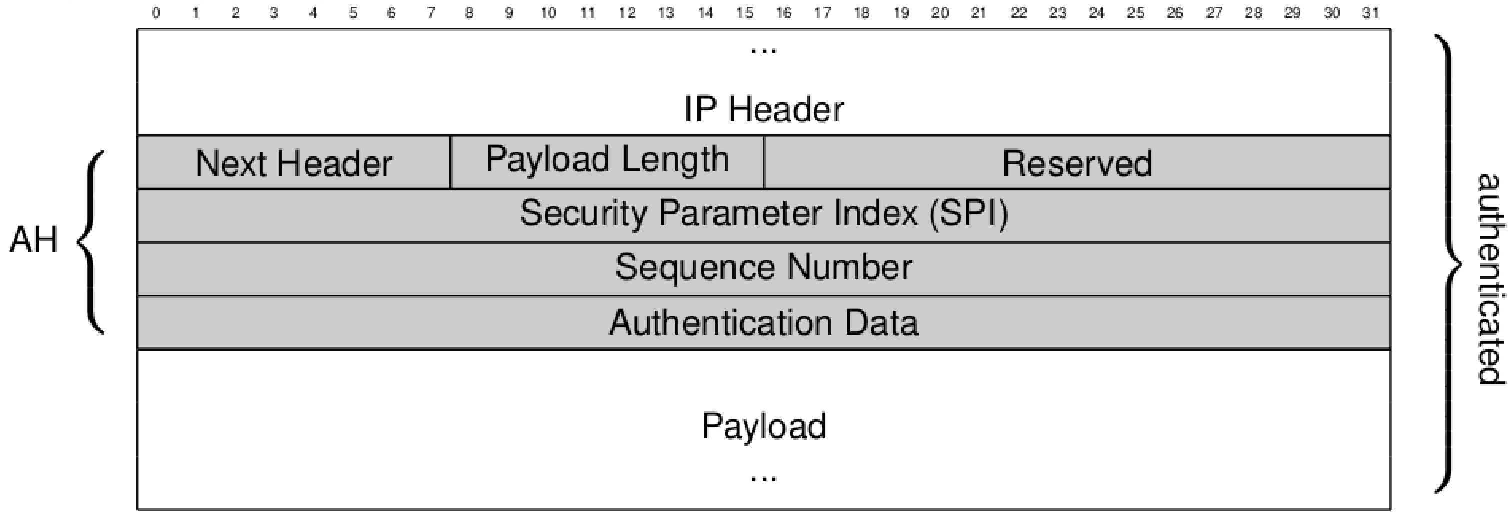
Encapsulating Security Payload

ESP Header:



The ESP immediately follows an IP/AH header and is indicated by `Next Header = 50`

Authentication Header



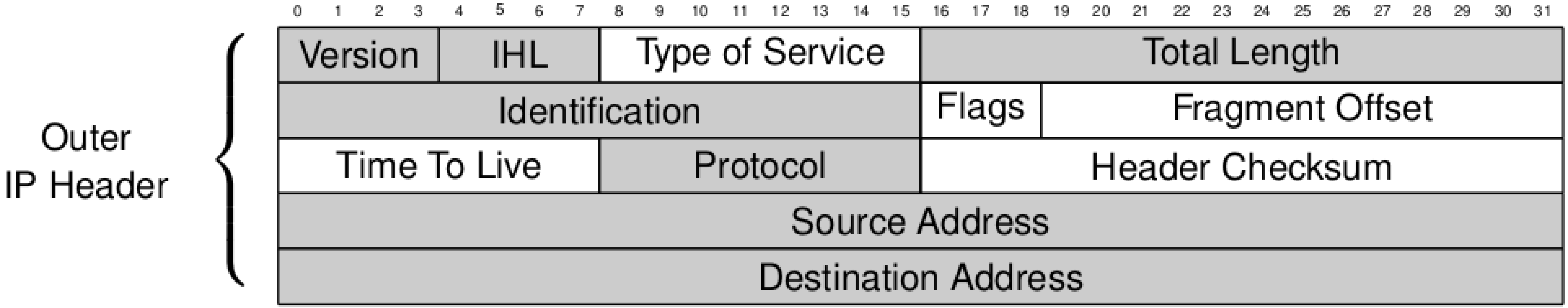
The AH immediately follows an IP Header and is indicated by `Next Header = 51`

Both ESP and AH can be applied at the same time with different ordering

- If ESP is applied first, AH is outer header
  - Advantage: ESP is also protected by AH
  - Consequence: Two SAs (one for each of AH/ESP) are needed for each direction

AH: fields changing in transit

- Although the AH protects the outer IP Header, some of its fields must not be protected, as they are subject to change during transit
  - This also applies to mutable IPv4 options or IPv6 extensions
- Such fields are assumed to be zero for MAC computation



All immutable fields, options and extensions (gray) are protected

Internet Key Exchange protocol v2

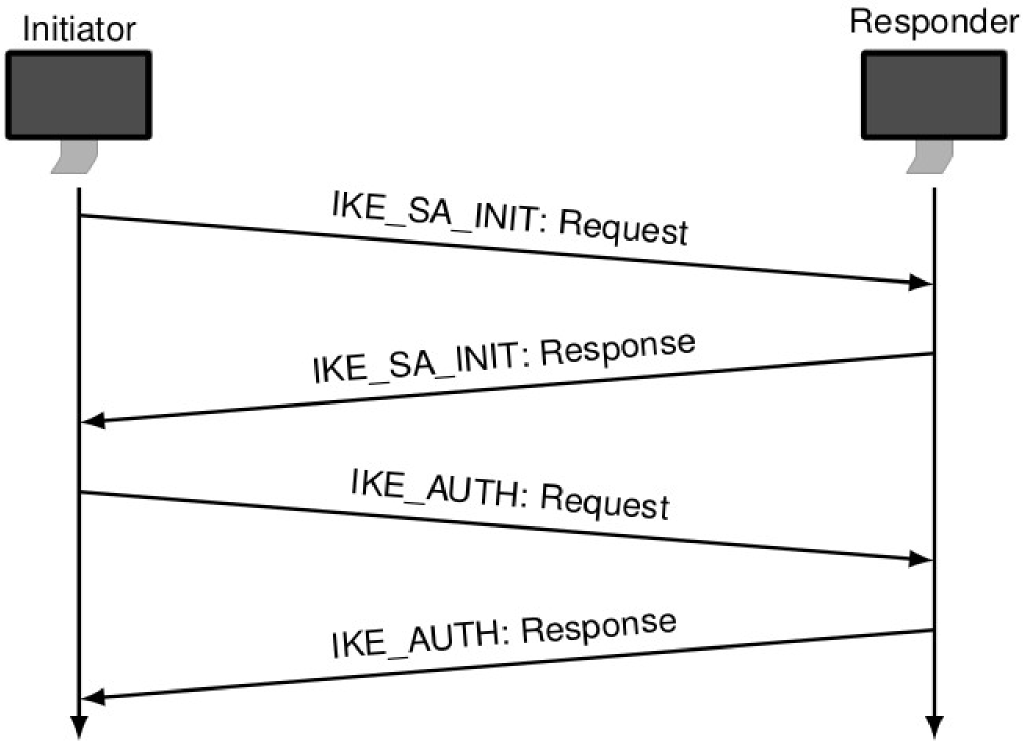
- A standardized authentication & key management protocol to dynamically establish SAs between two endpoints
- Standardized in [RFC4306] in December 2005
- Parts of IKEv1 poorly specified and spread over multiple RFCs
- IKEv2 provides unified authentication and key establishment
- Tries to achieve trade-off between features, complexity and security under realistic threat model
- [RFC5996] obsoletes [RFC4306]

IKEv2 General Properties

- Runs on UDP ports { 500, 4500 }
- Mutual authentication of the Initiator and Responder
- Negotiation of cryptographic suites
  - i.e., a complete set of algorithms used for SAs
- Support for DoS mitigation through use of cookies
- Integrated support for requesting an IP address (useful for VPNs)
- IKEv2’s latency is 2 round trips (i.e., 4 messages) in the common case

IKEv2 exchanges (phases)

- IKEv2 communication consists of message pairs
- Request and response
- One pair (request, response) is called an **exchange**
- An IKEv2 protocol run starts with two exchanges
  - IKE_SA_INIT
  - IKE_AUTH



SA_INIT and AUTH

IKE_SA_INIT (phase 1)

- Negotiates security parameters for a security association (IKE_SA)
- Sends nonces and Diffie-Hellman values
- IKE_SA is a set of security associations for protection of remaining IKE exchanges

IKE_AUTH (phase 2)

- Authenticates the previous messages
- Transmits identities
- Proves knowledge of corresponding identity secrets
- Creates first CHILD_SA
  - A CHILD_SA is a set of SAs, used to protect data with AH/ESP
  - The term CHILD_SA is synonymous to the common definition of an SA for IPsec AH and ESP

Other exchanges

CREATE_CHILD_SA

- Used to create another CHILD_SA
- Can also be used for re-keying

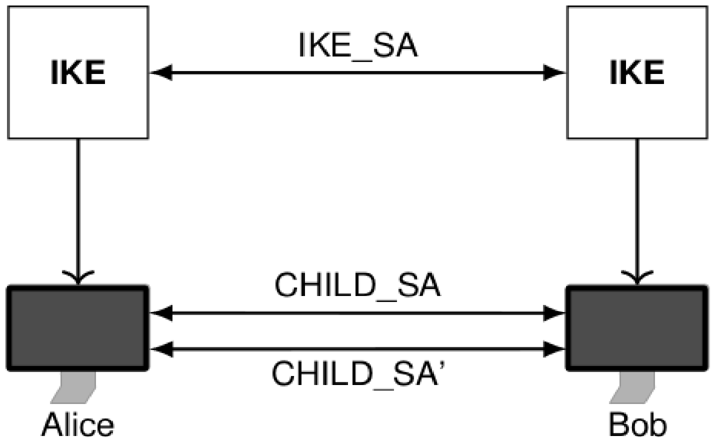
INFORMATIONAL

- Keep-Alive
- Deleting an SA
- Reporting error conditions, …

Negotiation and authantication IKE_SA_INIT

- IKE_SA_INIT negotiates:
  - Encryption algorithmn
  - Integrity protection algorithm
  - Diffie-Hellman group (i.e. DH parameters p and g)
  - Pseudo-Random function prf
- IKE_AUTH realizes authentication via public key signatures or long-term pre-shared secret
  - Authentication by signing (or calculating MAC K of) a block of data
  - The resulting value (AUTH) is transmitted in the IKE_AUTH exchange
  - Authentication is conducted by verifying the validity of the received AUTH payload

IKE outcome



- IKE_SA is a set of Security Associations established after the initial IKEv2 exchange (IKE_SA_INIT)
- IKE_SA is used to encrypt and integrity protect all the remaining IKEv2 exchanges
- CHILD_SA is a set of Security Associations used to protect IP traffic with the AH/ESP protocol
  - AH provides data integrity and replay protection
  - ESP provides data integrity, replay protection and encryption

IPsec in Linux kernel

- Supported natively, but...
- Hard to setup
- Deprecated tools, still useful and used (→Android):
  - ipsec-tools and racoon (IKEv1 daemon)
- Complete packages
  - Strongswan
  - Libreswan

To do the activities

- We will use Kathará (formerly known as netkit)
  - A container-based framework for experimenting computer networking: http://www.kathara.org/
- A virtual machine is made ready for you
  - https://drive.google.com/file/d/1W6JQzWVyH5_LKLD2OR6XH1ugPDP5LWP5/view?usp=sharing
- For not-Cybersecurity students, please have a look at the Network Infrastructure Lab material
  - http://stud.netgroup.uniroma2.it/~marcos/network_infrastructures/current/cyber/
    - Instructions are for netkit, we will use kathara

The kathara VM

- It should work in both Virtualbox and VMware
- It should work in Linux, Windows and MacOS
- There are some alias (shortcuts) prepared for you
  - Check with **`alias`**
- All the exercises can be found in the git repository:
  - https://github.com/vitome/pnd-labs.git
- You can move in the directory and run lstart
  - **NOTE**: launch docker first or the first lstart attempt can (…will…) fail

Preparation for ex3, ex4, ex5, ex6

- In the host kathara you need to install ipsec-tools and racoon
    - ipsec-tools for using the setkey program
    - racoon is the IKE daemon
- Both are deprecated, so you need to start the labs with an old kathara docker image
    - Find and tag a possible docker image
        - `docker images (show the installed images)`
        - `docker tag 6b9b242d2656 kathara/quagga:ipsec-tools`
    - Modify the lab.conf files to use the tagged image
        - `pc1[image]=kathara/quagga:ipsec-tools`

IPsec references

- IPsec reference for linux:
  - http://www.ipsec-howto.org/x299.html
- Have a look at the manuals:
  - man setkey
    - https://manpages.ubuntu.com/manpages/bionic/man8/setkey.8.html
  - man racoon
    - https://manpages.ubuntu.com/manpages/bionic/man8/racoon.8.html
- When a pre-shared key is required, you can use
  - `dd if=/dev/random count=16 bs=1| xxd -ps`

num of blocks = 16

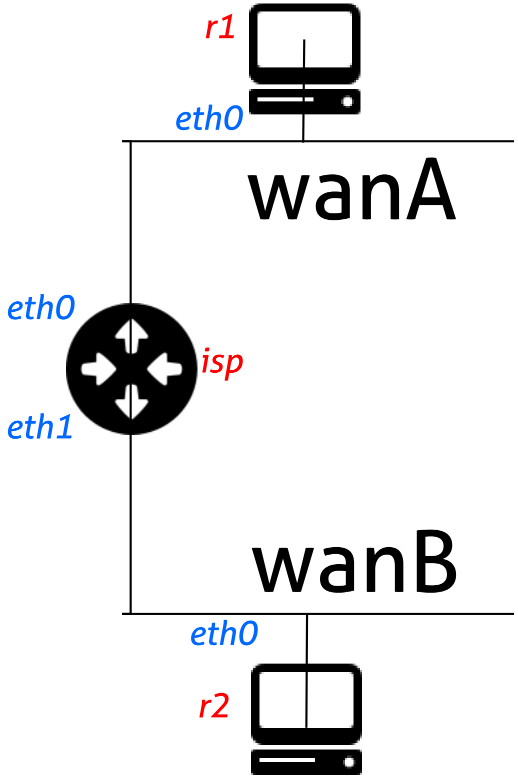
block size = 1 byte

Remember:
128 bits = 16 bytes
192 bits = 24 bytes
256 bits = 32 bytes

Lab activity: ex3, ex4

pnd-labs/lab5/ex3 and ex4

- You have to setup the addressing
  - Follow README instructions
- IPv4 in ex3
- IPv6 in ex4
- See the differences in how AH and ESP are implemented



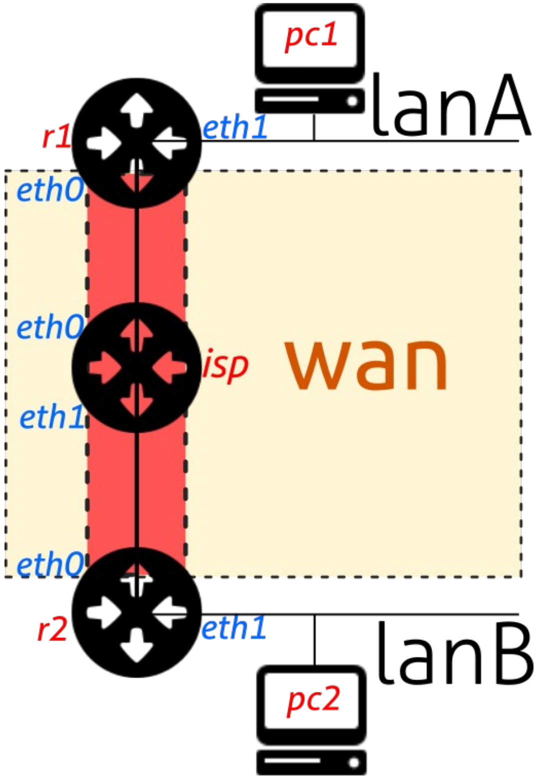
Notes about setkey

- You can run it with the command
  - **`/etc/init.d/setkey start`**
  - It can also be run using the **`-c`** flag, that accepts the directives from stdin
  - It can also be run using the **`-f`** flag, that accepts the directives from a file
- The default configuration file is located at
  - **`/etc/ipsec-tools.conf`**

Lab activity: ex5, ex6

pnd-labs/lab5/ex5 and ex6

- IPv4 addressing already setup
- Configure r1 and r2 to manage a VPN tunnel with IPsec between lanA and lanB
- Ex6: do it with the racoon daemon



Lab activity: ex7, ex8

strongSwan

- strongSwan is basically a keying daemon
  - It uses IKEv1 and IKEv2 to establish security associations (SA) between two peers
  - The IKE daemon is **charon**, configured with the “VICI” control interface (Versatile IKE control interface)
- A full configuration, then, is called a CHILD_SA made of:
  - the actual IPsec SAs (algorithms and keys used to encrypt and authenticate the traffic)
  - the policies that define which network traffic shall use such an SA
- The actual IPsec traffic is handled by the network and IPsec stack of the operating system kernel
  - https://docs.strongswan.org/docs/5.9/howtos/introduction.html

Authentication options

- Public Key Authentication
  - RSA, ECDSA or EdDSA X.509 certificates to verify the authenticity of the peer
- Pre-Shared-Key (PSK)
  - A pre-shared-key is an easy to deploy option but it requires strong secrets to be secure
- Extensible Authentication Protocol (EAP)
  - This covers several possible authentication methods, based on username/password authentication (EAP-MD5, EAP-MSCHAPv2, EAP-GTC) or on certificates (EAP-TLS), some can even tunnel other EAP methods (EAP-TTLS, EAP-PEAP)
- eXtended Authentication (Xauth)
  - XAuth provides a flexible authentication framework within IKEv1

Configuration Files

- strongSwan is configured with
  - The `swanctl` command line tool
  - The `swanctl.conf` configuration file in the `swanctl` directory
- Global strongSwan settings as well as plugin-specific configurations are defined in `strongswan.conf`
  - Alternatively, the legacy ipsec `stroke` interface and its `ipsec.conf` and `ipsec.secrets` configuration files may be used.

Useful commands

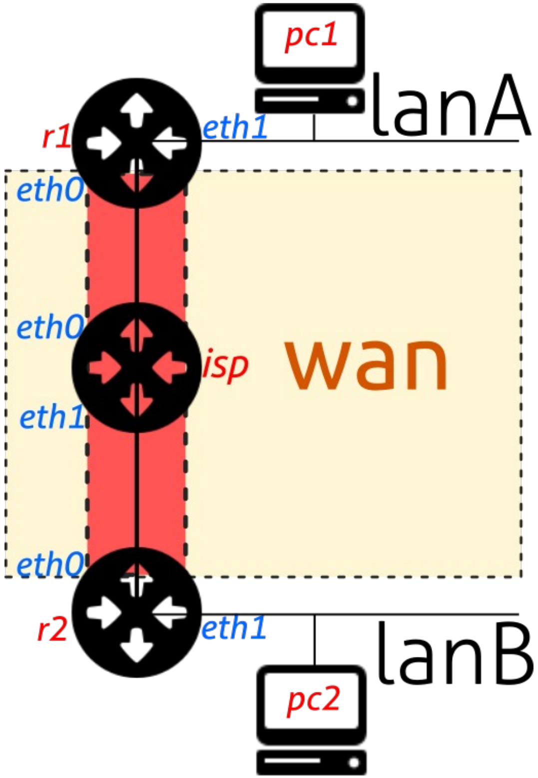
- `ipsec`
  - `start|stop|restart|status|statusall`
  - `listcerts|listalgs|listpubkeys`
- `swanctl`
  - `--load-creds|--load-conns|--load-all`
  - `--list-sas`
  - `--initiate --child <connection>`

Proposals

- The list of preferences, please refer here:
  - https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites
- Not so straightforward, use with caution
  - Safe choice: `default`

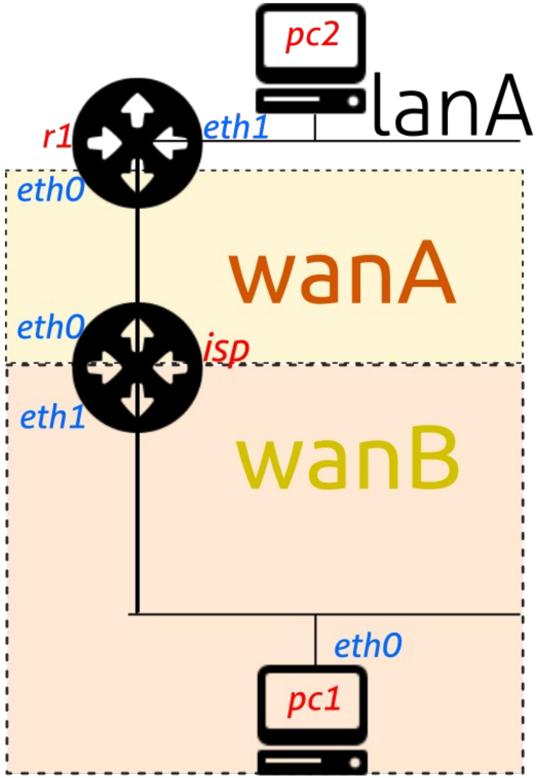
pnd-labs/lab5/ex7

- IPv4 addressing already setup
- Configure r1 and r2 to manage a VPN tunnel with IPsec between lanA and lanB
- Use the strongSwan framework:
  - ipsec start
  - swanctl



pnd-labs/lab5/ex8

- IPv4 addressing already setup
- Configure r1 to be the VPN GW for pc1 in order to access lanA
- Use the strongSwan framework:
  - ipsec start
  - swanctl



That's all for today

- **Questions?**
- See you on Monday
- Resources:
  - http://www.ipsec-howto.org/
  - http://www.unixwiz.net/techtips/iguide-ipsec.html
  - Chapter 24 textbook
  - Virtual private networking, Gilbert Held, Wiley ed.
  - Guide to IPsec VPNs, NIST800-77
  - Guide to SSL VPNs, NIST-SP800-113
  - IPsec
  - https://docs.strongswan.org/docs/5.9/howtos/introduction.html