Network Security

Securing communications (SSL/TLS and IPSec)

Marcus Bendtsen, Andrei Gurtov

Institutionen för Datavetenskap (IDA)

Avdelningen för Databas- och Informationsteknik (ADIT)

Network communication

- Who are you talking to?
  - How do we make sure that the host we think we are communicating with really is that particular host?
    - What if somebody is pretending to be the recipient?
    - What is somebody is re-routing your packets elsewhere and recording its contents?
  - You send a request to a webserver (www.example.com), but somebody on the wire catches your requests, and responds with a malicious web page.
  - You send a request with username/password, somebody on the wire picks up the request, saves the contents, and lets the packet reach its true destination.

Network communication

- Is somebody else listening?
  - How do we make sure that nobody unauthorized is listening?
    - *This may actually be borderline impossible.*
- How do we make sure that those listening can not exploit the communication?
  - What if they are changing the packets contents?
    - You wanted to move money from one bank account to another, somebody on the wire changes the recipient bank account.
  - What if they are saving the packets and sending them again?
    - A so called *replay* attack, where somebody on the wire saves your requests, and sends them again later:
      - You request to move money from one account to another, later the request is sent again, and again, and again from somebody who saved the request.

SSL/TLS

- Secure Socket Layer
- Transport Layer Security
- TLS = SSL v3 + updates
- TLS 1.2 no longer backward compatible with SSL

- Placed between the application layer and the transport layer, it can support multiple application layer protocols (HTTP, FTP, Telnet, etc.).

- More or less *de facto* way of securing network communication
  - Strong encryption (under some scrutiny in 2013)
  - Integrity protection (you can not tamper with the data)
  - Mutual authentication (both parties authenticate)

TLS (Overview)

- A TSL session is a long-lived association between two communicating hosts, a session may span many connections. This saves time from creating new sessions.

- The client first sends a message that contains which version to use, as well as which encryption algorithms it prefers etc.

- The client receives a certificate from the server, which needs to be certified by a known certificate authority. This certificate gives the client the servers public key. (If necessary, the client can authenticate itself to the server).

- The client chooses a key for symmetric encryption, sends this to the server (encrypted by the servers public key), and now communication can be done using this key and symmetric encryption (e.g. the key is used to seed the PRNG).

Certificate validation

- A certificate is an electronic document containing public information about an entity (a server, person or similar).
- If the certificate is used for secure communications then usually a public key is made available through the certificate.
  - Great, so I download the certificate, I get the public key and I can start sending encrypted data to the owner of the certificate.
  - **Issue:** How can you guarantee that the certificate is valid? What if I create a certificate saying that I am Google and here is the public key, now start sending me your data.
- The certificate needs to be verified. Special certified authorities (CA) issue certificates that are signed with their private key. We can very the signature by using the CAs public key.
- How do we get the CAs public key?
  - This is the issue with distributing public keys, how can we assure that the public key is the ***real*** public key?
  - CAs public keys are delivered with e.g. the web browser.

TLS

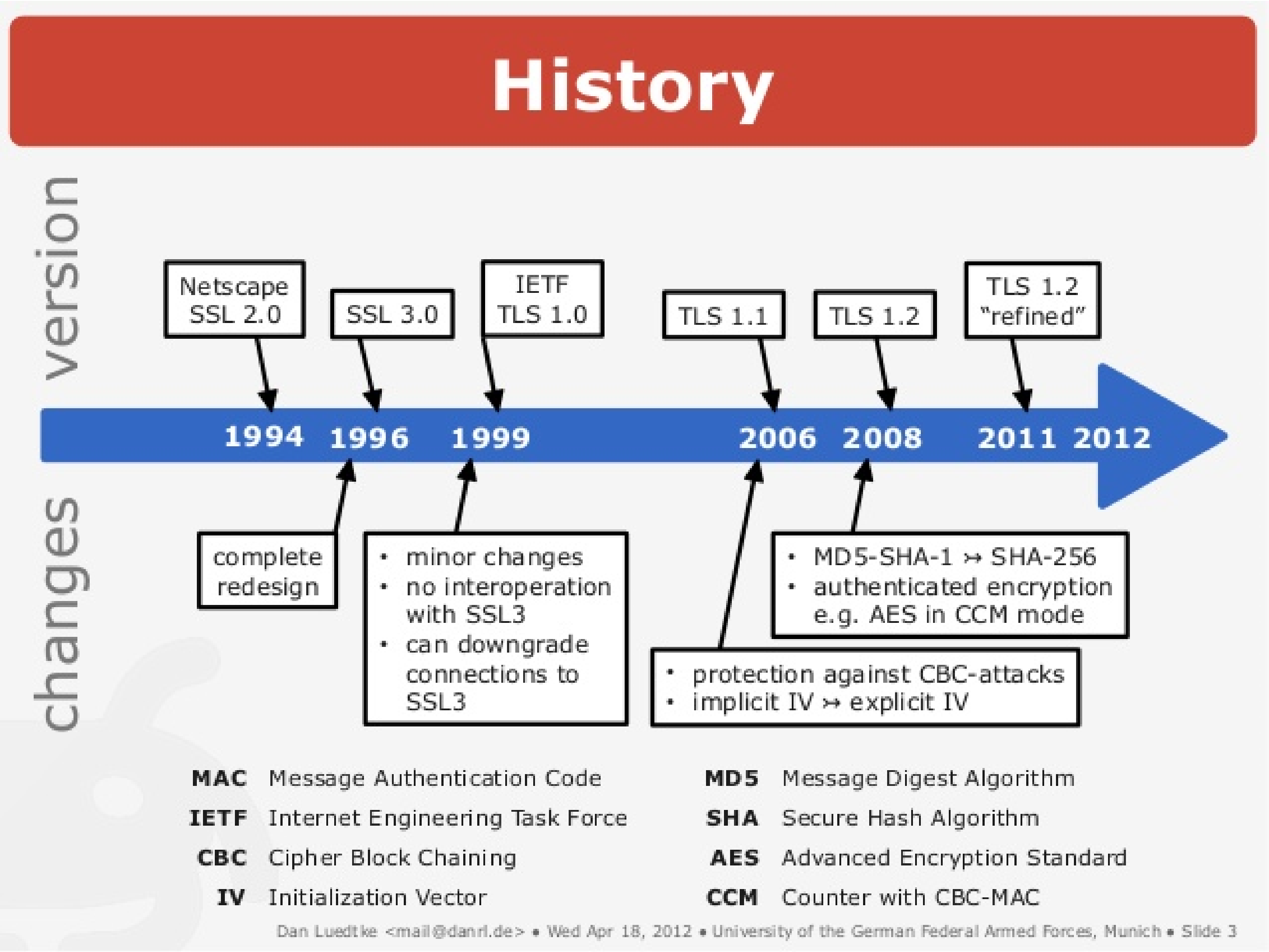
- TLS seems very secure, and has been design to be resilient to many attacks (as long as the chosen encryption method is solid).

- It has built in methods to ensure that *replay* attacks can not happen. (adds sequence numbers to all requests).

- MITM attacks in general are made ineffective, as all communication is encrypted and integrity checks are made using HMAC. (You can steal the data, but it is useless to you).

- Biggest problem are the certificates, and people accepting invalid certificates (even when warned that they are invalid).
    - Not a problem unique to TLS/SSL. When using SSH how many times have you actually checked the fingerprint prior to connecting to a server for the first time?

TLS evolution



History

version

Netscape SSL 2.0

SSL 3.0

IETF TLS 1.0

TLS 1.1

TLS 1.2

TLS 1.2 “refined”

1994 1996 1999 2006 2008 2011 2012

changes

complete redesign

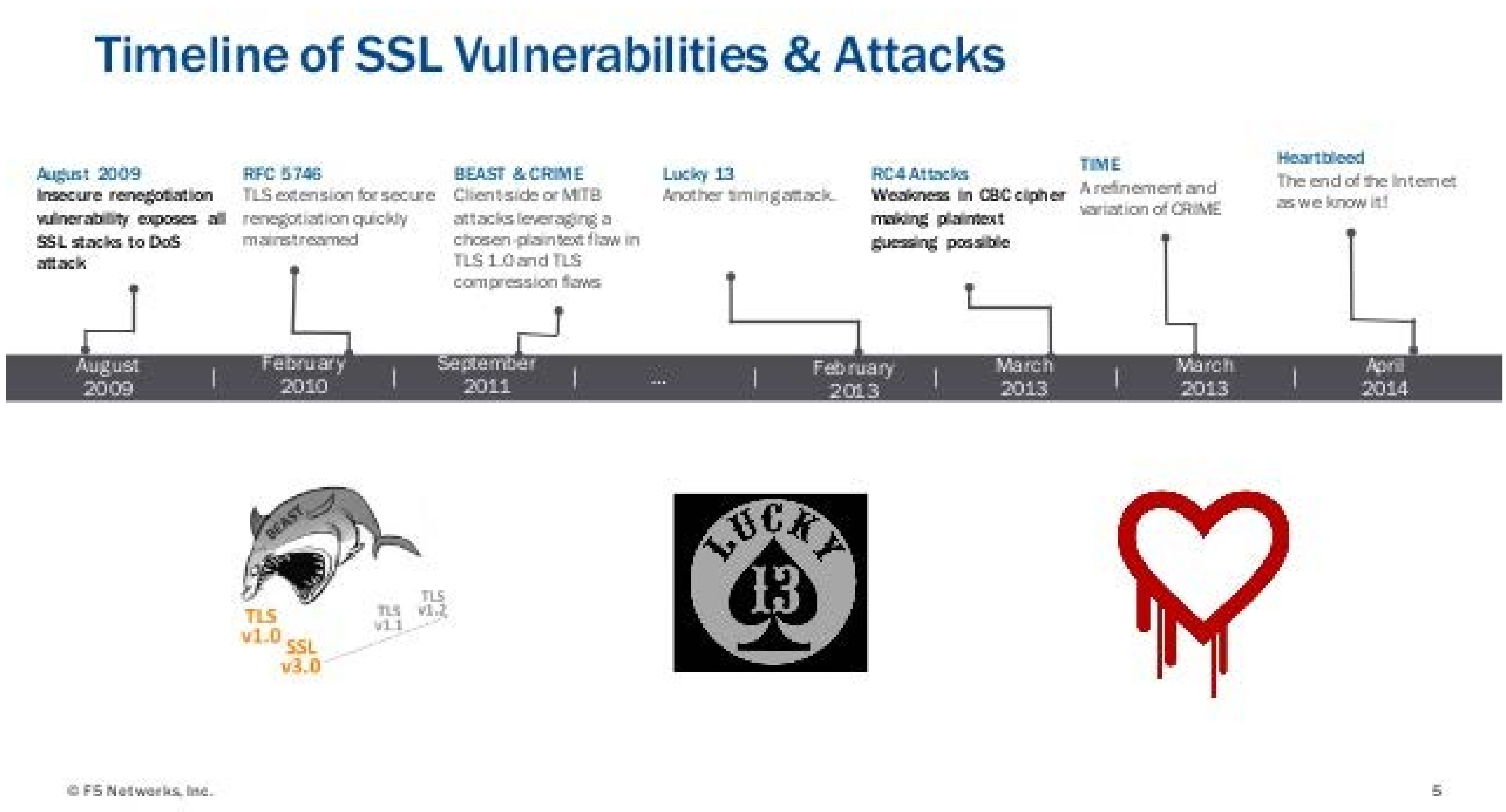
- minor changes
- no interoperation with SSL3
- can downgrade connections to SSL3

- protection against CBC-attacks
- implicit IV ↣ explicit IV

- MD5-SHA-1 ↣ SHA-256
- authenticated encryption e.g. AES in CCM mode

MAC	Message Authentication Code	**MD5**	Message Digest Algorithm
IETF	Internet Engineering Task Force	**SHA**	Secure Hash Algorithm
CBC	Cipher Block Chaining	**AES**	Advanced Encryption Standard
IV	Initialization Vector	**CCM**	Counter with CBC-MAC

Dan Luedtke <mail@danrl.de> • Wed Apr 18, 2012 • University of the German Federal Armed Forces, Munich • Slide 3



Timeline of SSL Vulnerabilities & Attacks

August 2009
Insecure renegotiation vulnerability exposes all SSL stacks to DoS attack

RFC 5746
TLS extension for secure renegotiation quickly mainstreamed

BEAST & CRIME
Client side or MITB attacks leveraging a chosen-plaintext flaw in TLS 1.0 and TLS compression flaws

Lucky 13
Another timing attack.

RC4 Attacks
Weakness in CBC cipher making plaintext guessing possible

TIME
A refinement and variation of CRIME

Heartbleed
The end of the Internet as we know it!

August 2009 | February 2010 | September 2011 | ... | February 2013 | March 2013 | March 2013 | April 2014

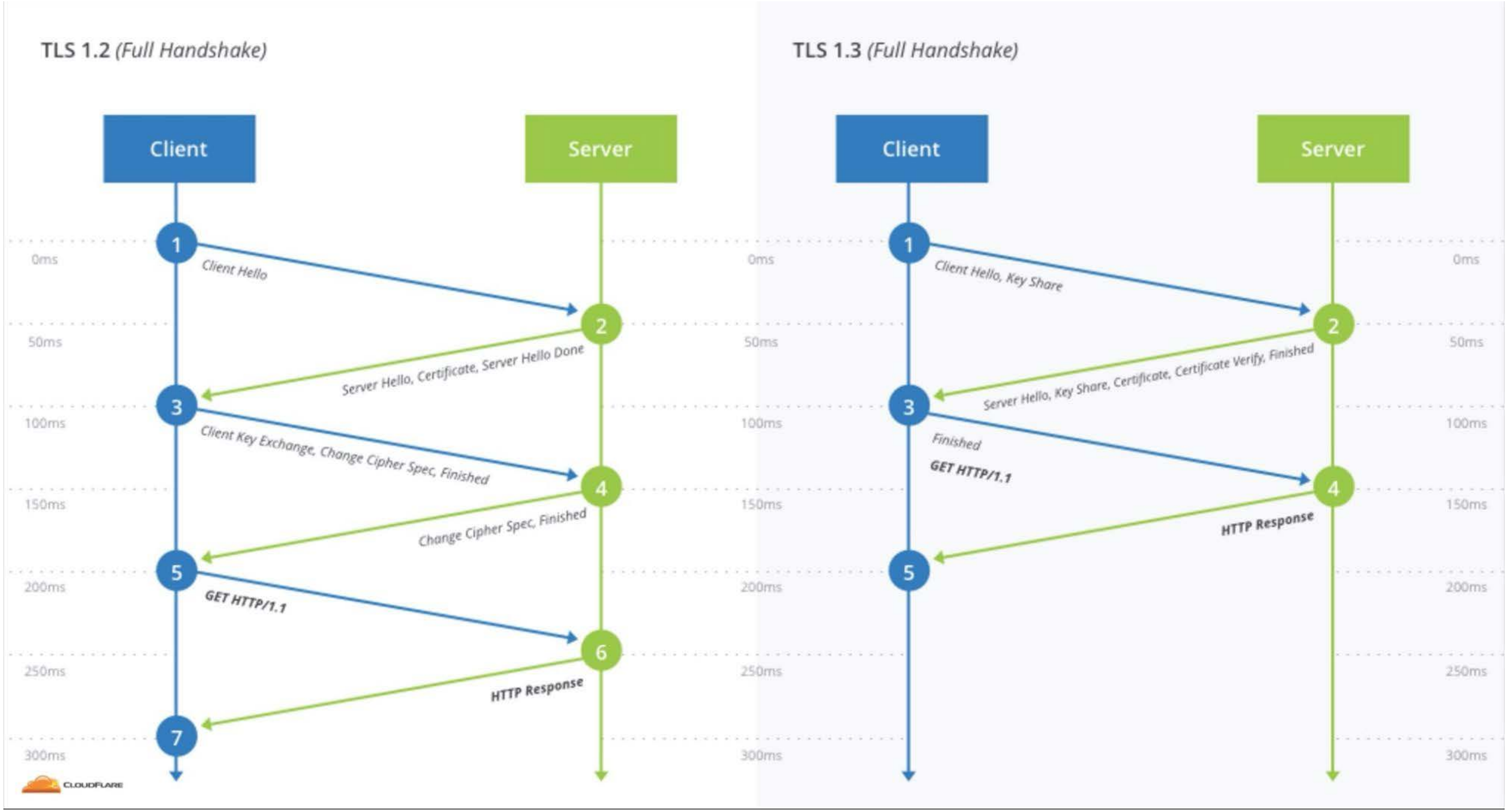
© F5 Networks, Inc.

5

TLS 1.3

- draft-ietf-tls-tls13-18 to be finalized soon by IETF
- Deprecated crypto removed
  - RSA key transport — Doesn’t provide forward secrecy
  - CBC mode ciphers — Responsible for BEAST, and Lucky 13
  - RC4 stream cipher — Not secure for use in HTTPS
  - SHA-1 hash function — Deprecated in favor of SHA-2
  - Arbitrary Diffie-Hellman groups — CVE-2016-0701
  - Export ciphers — Responsible for FREAK and LogJam
- Already available in Chrome and Firefox for testing
- Ref CloudFlare

TLS 1.3 Reduced Latency

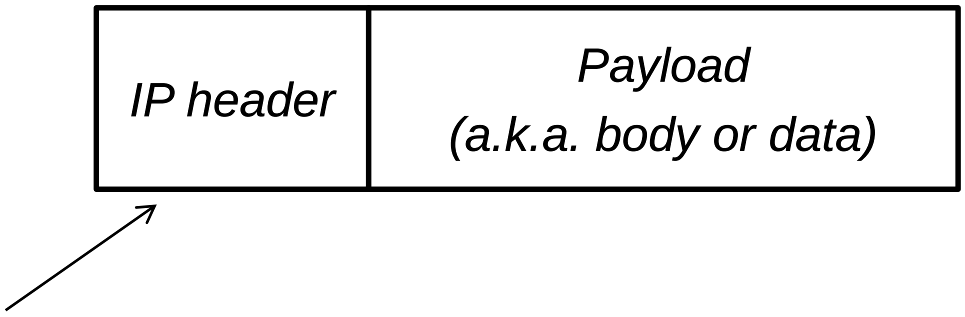


IPSEC

IPSec

- IPSec involves security at the **network layer**.
- The concerns are the same:
  - Confidentiality
  - Integrity
  - Authentication
- IPSec is really a set of different protocols:
  - They offer different services (authentication, integrity).
  - They can be combined or used separately.
  - In short what you get is a secure channel between two hosts.

Simple view of an IP packet



Includes (amongst others):

Time to live – How many hops before it should be thrown away

Source IP

Destination IP

The actual layout of all headers and paddings are not important for this course

AH and ESP

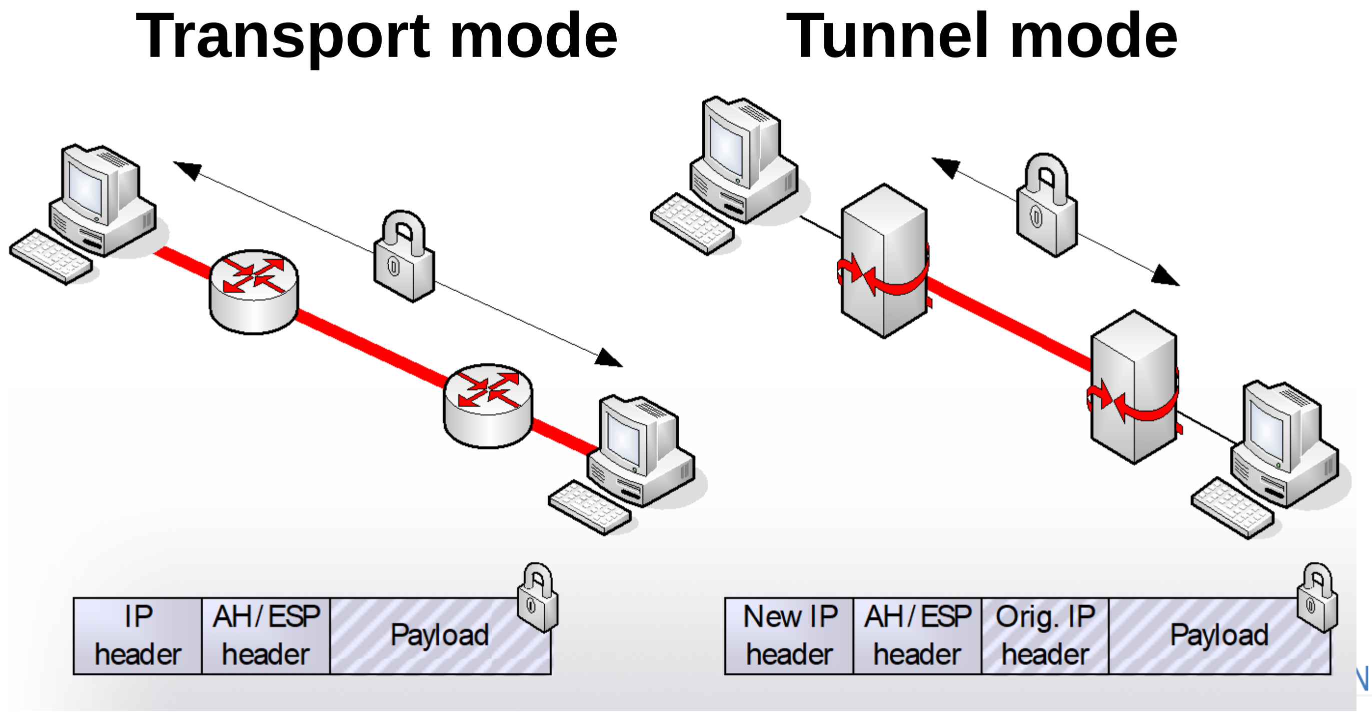
Two of the main protocols in IPSec:

- Authentication Header Protocol (AH)
- Encapsulating Security Payload Protocol (ESP)

- **Shortest explanation:** They add headers and trailers to IP packets that *protect* either parts of the IP header and body, or only the body.
  - The protection can consist of a combination of authentication, integrity and confidentiality.
  - You choose what you need for your system.

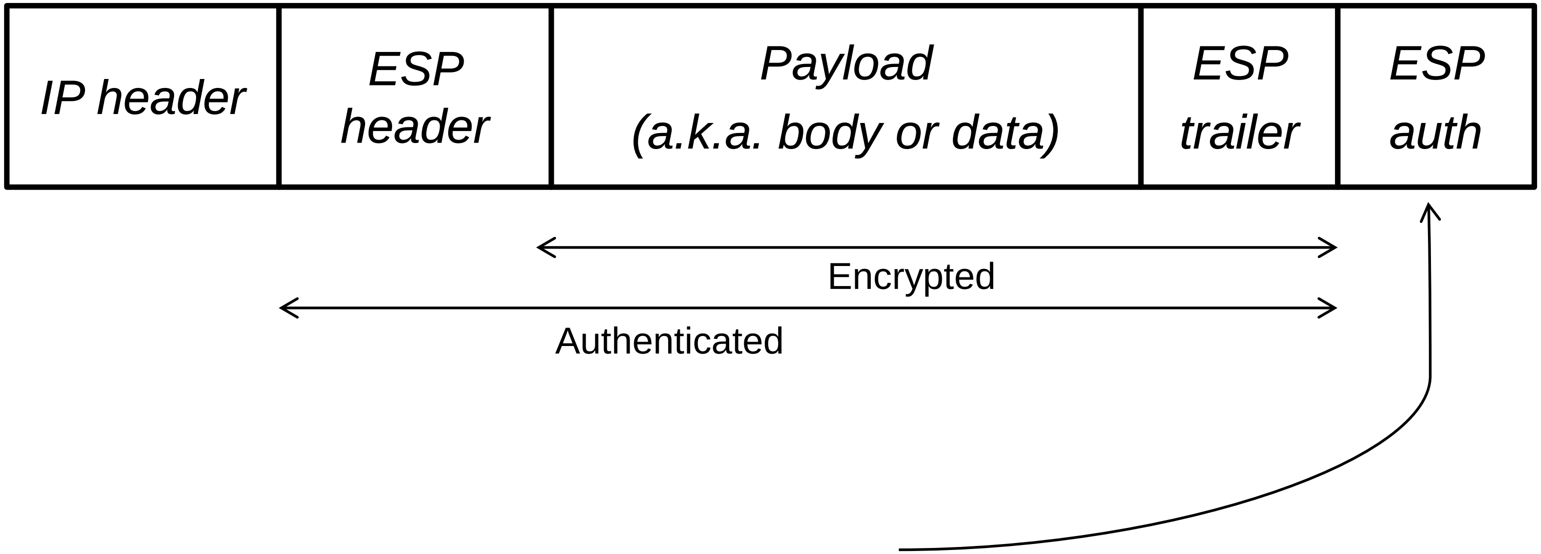
- IPSec can operate in two modes.
  - **Transport mode:** Two peers communicating, they add the AH/ESP headers and the data remains *protected* until it reaches the peers.

  - **Tunnel mode:** Existing IP packets are encapsulated in new IP headers. Peers communicate as normal, but gateways add and remove the IPSec headers – Designed for implementing VPN.
    - If two physical sites need secure communication over a public channel they can set up tunnel mode communication and allow all clients behind the gateways communicate as normal.



ESP

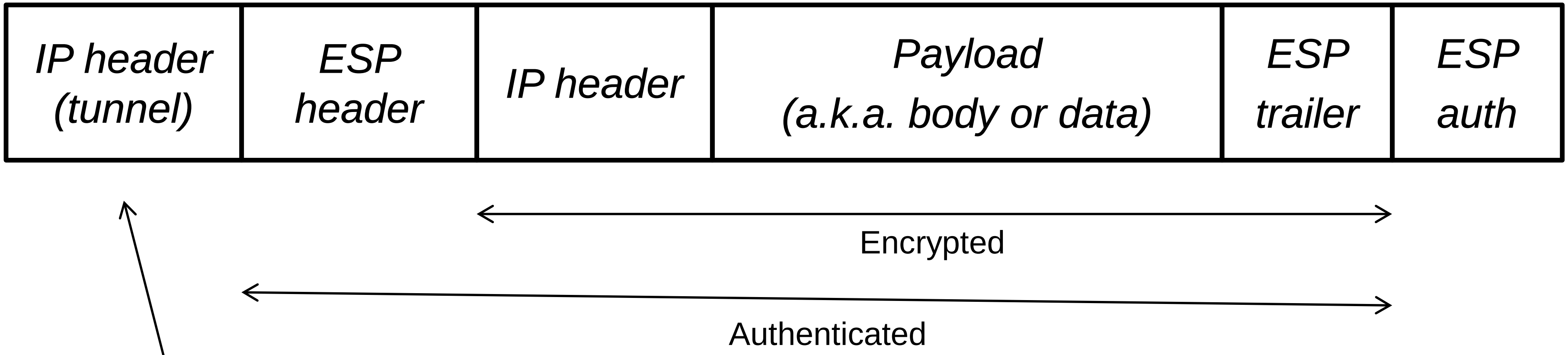
Transport mode



- ESP adds encryption to the payload, and can optionally add authentication.
- In *transport mode* it however does nothing to protect the original IP header.

ESP

Tunnel mode



IP header (tunnel)	*ESP header*	*IP header*	*Payload (a.k.a. body or data)*	*ESP trailer*	*ESP auth*

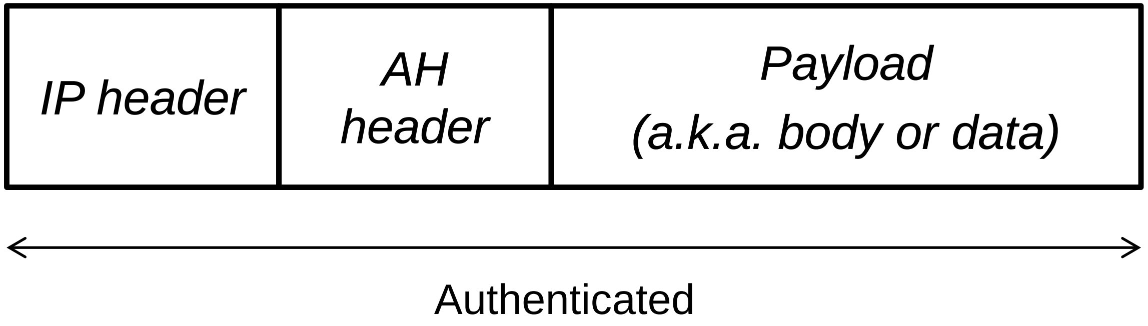
Encrypted

Authenticated

- ESP in tunnel mode protects the original IP header
- Again, authentication is optional
- The IP header added by the tunnel is not authenticated

AH

Transport mode



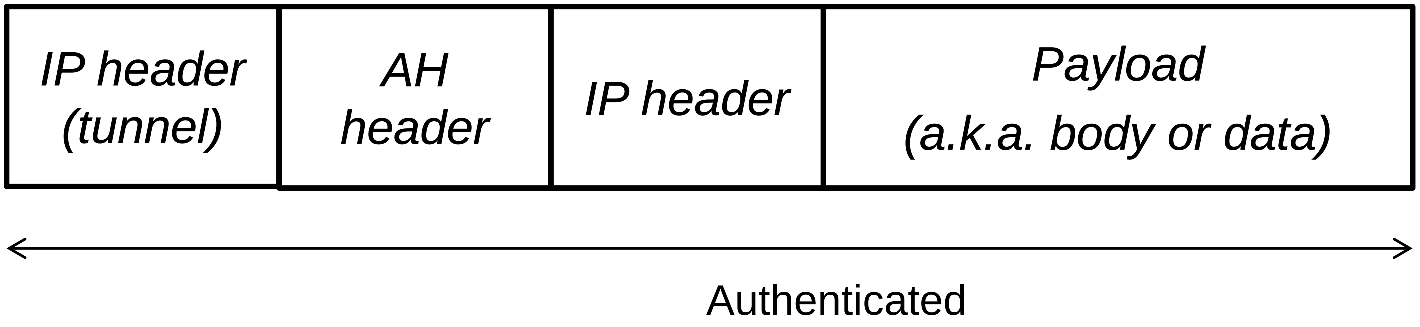
- AH authenticates the static IP headers.

- Some headers change during transport, like TTL.
- AH can not authenticate these, as they have to be able to change.
- AH is not compatible with NAT (src IP address is authenticated, can not change it)

- AH does nothing to encrypt the data.

AH

Tunnel mode



IP header (tunnel)

AH header

IP header

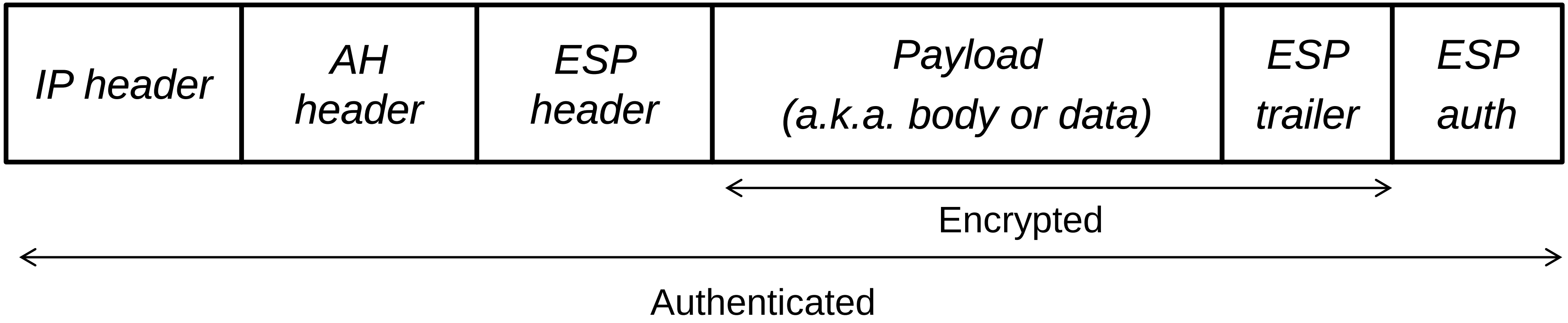
Payload (a.k.a. body or data)

Authenticated

- In tunnel mode even the added IP header is authenticated

AH & ESP

Transport mode

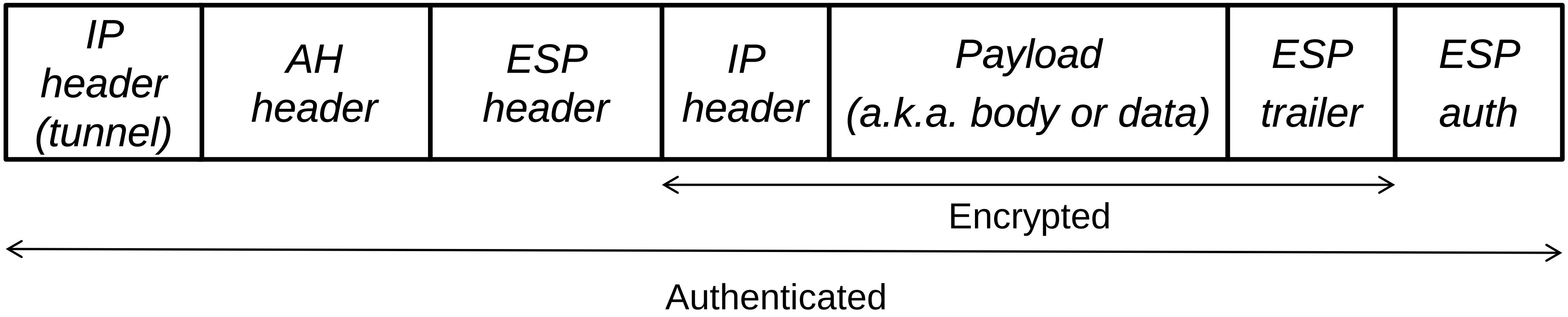


IP header | *AH header* | *ESP header* | *Payload (a.k.a. body or data)* | *ESP trailer* | *ESP auth*

Encrypted

Authenticated

Tunnel mode



IP header (tunnel) | *AH header* | *ESP header* | *IP header* | *Payload (a.k.a. body or data)* | *ESP trailer* | *ESP auth*

Encrypted

Authenticated

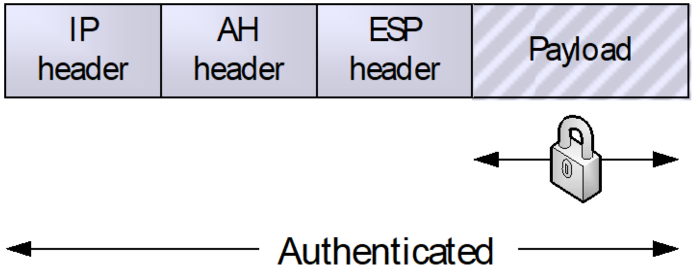
(only static header are authenticated)

Authentication and encryption

- Two ways of getting both authentication and encryption.
  - Nest ESP in AH, or use authenticated ESP.

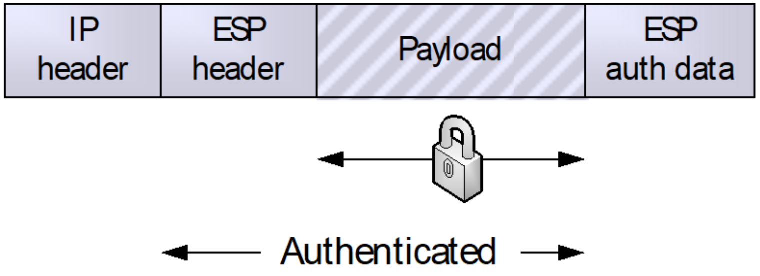
Nest ESP in AH

- Apply ESP to packet, making it encrypted, then add AH to authenticate.
- This way the IP header, ESP header and the encrypted payload is authenticated.



Authenticated ESP

- Apply authenticated ESP to the payload. ESP header encrypts and ESP trailer authenticates.
- Since this is ESP the IP header is not authenticated.



AH & ESP

- As you can see you can mix and match heavily.
- Why wouldn't you choose full out ESP with authentication and AH in tunnel mode?
  - Many reasons:
    - Extra overhead – If your application does not need authentication then the extra overhead is unnecessary, just use regular ESP.
    - Maybe you can not encrypt the IP header because you must allow for logging of network data.
    - Maybe you need NAT.
    - Maybe authentication is all you need because you encrypt your payload on your own in a different way.
    - Etc...

AH & ESP headers

- **Few more details:**
  - Both AH and ESP can ensure integrity
  - Both AH and ESP can protect against *replay* attacks

- Two important headers in both AH and ESP
  - **Security Parameter Index** (used to decide how the packet should be processed in both ends of the communication)
  - **Sequence number** (used to protect against replay attacks)

Security associations

- Security associations (SA)
  - One-way relationship between IPSec hosts.
  - Allows one host to send IPSec-protected packets to the other.
  - SA determines how packets are processed, which algorithms to use, parameters, keys, etc.
  - Need two SAs for bi-directional secure communication (one in each direction).

  - SAs are specific to source, destination, transport protocol, port, and other data.
    - If IPSec is being used from one host to another for both port 80 and 22, then there will be one SA for each.
    - Furthermore an SA can only handle either ESP or AH, so if both are used there will be an SA for each.

Security Association Database (SAD)

- SAs are stored in the ***Security Association Database*** (SAD), and are uniquely identified by the ***Security Parameter Index*** (SPI). (Remember that SPI is a header in AH/ESP)

  - At the *destination*, the processing for each packet is done by finding the correct SA from the SAD (based on destination, ESP/AH used, SPI, etc.).

  - The SA contains the needed information to process the packet correctly (including algorithms, keys, etc.).

  - Some packets have multiple SAs, so sometimes we find a *SA bundle* in the SAD. These should be processed in a predefined order.

Security Policy Database (SPD)

- At the *sender*, the processing is determined by the Security Policy Database (SPD).
  - When a packet comes to the IPsec mechanism a lookup is made in the SPD to determine how the packet should be processed (using source, destination, port, etc.).
  - It could be the case that the SPD just lets the packet through, or that it is discarded.
  - If IPsec is to be applied, then the SPD references an SA in the SAD for processing details.

Security Policy Database (SPD)

- The SPD contains entries that tell the outbound side (the *sender*) how to process the packet (including *bypass* and *discard*).

- At the incoming side we look in SAD to find the correct SA (or SA bundle).

The outbound side looks this up in SAD

- SPD Entry:
  - Source: 10.1.2.3
  - Dest: 10.4.4.4
  - Action: Required IPSec processing
  - SA: ESP/**SPI 4365**

- SAD entry (ESP/**SPI: 4365**)
  - Mode: Transport
  - Algorithm: 3DES
  - Key: ….
  - Replay protection ….
  - …

The incoming side looks up this SAD entry (i.e. an SA)

IPSec and IKE (Internet Key Exchange)

- IPSec is dependent on security associations (SA)
  - For static associations, such as VPN tunnels, this is fine.
    - We can manually create the SAs *out-of-band*
  - But if we want dynamic associations, something more is required.
    - On-demand VPN tunnels for laptops, IPSec on wireless clients.
  - We need a protocol for ***negotiating*** security associations, and for IPSec this is IKE.

IKE

- **Shortest possible explanation:**
  - A clear text message between hosts to confirm that they can connect.
  - *Diffie-Hellman* key exchange.
  - IKE now negotiates SAs, including what algorithms to use, keys etc. It may be the case that several SAs are negotiated to create a bundle.
  - Packets can now be sent using IPsec.

Diffie-Hellman key exchange

- Two parties (A and B) agree on two non-secret numbers: *g* and *p.*
  - *p* should be a very large prime number
  - *g* and *p* can be made public

- Each party chooses a random large integer *x (x(A) and x(B))*
  - x(A) and x(B) are the *secret values*

- Based on the *secret* value a *public* value *Y* is created for each party.
  - $Y_A = g^{x(A)} \bmod p$, $Y_B = g^{x(B)} \bmod p$
  - The two parties exchange their public values.

Diffie-Hellman key exchange

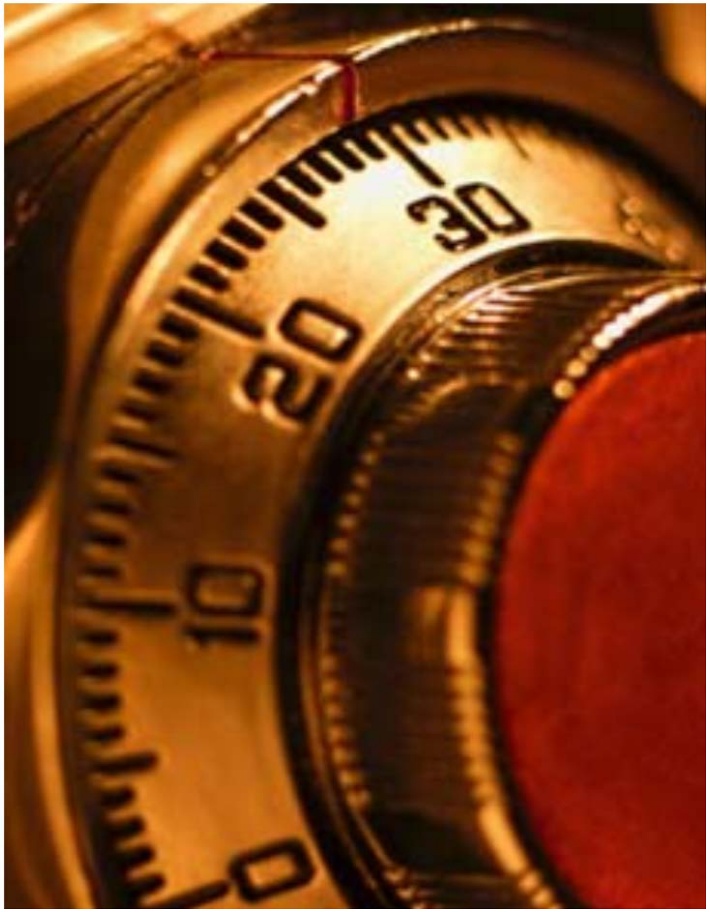
- Each party computes a shares secret
  - $k = Y_B^{x(A)} \bmod p$ (computed by A, since x(A) is only known by A)
  - $k' = Y_A^{x(B)} \bmod p$ (computed by B, since x(B) is only known by B)
- When the algorithm completes, then both parties have the same shared secret, since k = k'
  - $k = k' = g^{x(A)x(B)} \bmod p$
- No one listening on the channel can create this shared secret as you need to know at least one of the parties secret value x(A) or x(B).
  - Those listening to the communication only know g, p, Y_A, Y_B
- The two parties can now send sensitive information to each other.

Diffie-Hellman

- Attacks:
  - **Denial of service:** An attacker may try to stop all packets going back and forth so that A and B can not connect.
  - **Man in the middle** (modulo removed for brevity)**:**
    - Assume that O intercepts the exchange of public values $g^{x(A)}$ and $g^{x(B)}$.
    - O replaces these with $g^{z(A)}$ and $g^{z(B)}$.
    - A thinks that the secret key is $g^{x(A)z(B)}$ and B thinks its $g^{z(A)x(B)}$.
    - Assume that A and B uses a symmetric encryption algorithm using these incorrect secret keys.
    - Packages going from A to B are intercepted by O. O decrypts the package since O knows $g^{x(A)z(B)}$, saves the content, encrypts using $g^{z(A)x(B)}$ and sends to B.
    - A and B are unaware that O is seeing their entire communication.

IPSec summary

- Security at the network layer
- Protocol family:
  - ESP for encryption
  - AH for authentication
  - IKE/IKEv2/HIP for key exchange
- Security associations
  - One way “connection” between peers



Linköpings universitet

expanding reality

www.liu.se