ipsec-esp-tunnel-nat-t.annotated

R4 (tunnel gateway)	R6 (gateway behind NAT)

[1] IKE_SA_INIT →

UDP port	500
Initiator SPI	d62ecfb3df598e44
Message	Request
Message ID	0x00000000
Encryption	ENCR_AES_CBC (12)
Integrity	AUTH_HMAC_SHA2_384_192 (13)
DH group	384-bit random ECP group (20)
Notify	NAT_DETECTION_SOURCE_IP
Notify	NAT_DETECTION_DESTINATION_IP

[Frame 1] IKE_SA_INIT, from outside the NAT.

This time the gateway outside the NAT starts. R4 sends to `10.3.4.6`, R6's public address. R3 translates the destination to R6's private `192.168.0.6`, which is what this capture, taken on R6's side, shows.

An outside device can start a session only because R3 has a static translation for R6. A dynamic NAT would have no mapping until R6 sent something first.

R4's NAT-detection hashes cover its own address `10.3.4.4` and the destination it used, `10.3.4.6`, both on port 500.

[2] IKE_SA_INIT ←

UDP port	500
Initiator SPI	d62ecfb3df598e44
Message	Response
Message ID	0x00000000
Encryption	ENCR_AES_CBC (12)
Integrity	AUTH_HMAC_SHA2_384_192 (13)
DH group	384-bit random ECP group (20)
Notify	NAT_DETECTION_SOURCE_IP
Notify	NAT_DETECTION_DESTINATION_IP

[Frame 2] Both sides detect the NAT.

R6 replies from `192.168.0.6`; R3 changes the source to `10.3.4.6` on the way out.

- R6 already knows from frame 1: R4's destination hash was for `10.3.4.6`, not `192.168.0.6`, so R6 is behind a NAT.
- R6's source hash is for `192.168.0.6`, but R4 received the packet from `10.3.4.6`. So R4 knows its peer is behind a NAT.

[3] IKE_SA_INIT →

UDP port	4500
Initiator SPI	d62ecfb3df598e44
Message	Request
Message ID	0x00000001
Payload	Encrypted and Authenticated (SK)

[Frame 3] IKE_AUTH on UDP 4500.

R4, the initiator, moves IKE to port 4500 (RFC 7296 section 2.23). The message starts with the 4-byte zero Non-ESP Marker that separates IKE from ESP on the shared port (RFC 3948 section 2.2).

Encrypted inside: the Child SA proposal (ESP-NULL, tunnel mode) and the traffic selectors.

[4] IKE_SA_INIT ←

UDP port	4500
Initiator SPI	d62ecfb3df598e44
Message	Response
Message ID	0x00000001
Payload	Encrypted and Authenticated (SK)

[Frame 4] IKE_AUTH response: the tunnel is up.

The SA pair is installed with UDP encapsulation. About 15 ms in total.

[5] ESP SPI 0xe09b01d1 →

IP TTL	254
IP src	10.3.4.4
IP dst	192.168.0.6
UDP encapsulation	port 4500
ESP Sequence	1
ESP Pad Length	2
Next header	IPIP (0x04)
IP TTL	254
IP src	10.4.5.5
IP dst	10.6.6.6
ICMP	Echo (ping) request
ICMP seq	1 (0x0001)

[Frame 5] Tunnel mode inside UDP: four layers.

R5, behind R4, pings `10.6.6.6`, a loopback on R6. The packet is the most deeply nested in these captures:

1. Outer IP, `10.3.4.4` -> `192.168.0.6` (R4 sent it to `10.3.4.6`; R3 translated it).
2. UDP 4500 to 4500, checksum 0 (RFC 3948 section 2.1).
3. ESP, SPI `0xe09b01d1`, sequence 1, `Next header IPIP (4)`.
4. Inner IP, `10.4.5.5` -> `10.6.6.6`, carrying the ICMP echo request.

The NAT rewrites only the outer IP header and, if it needs to, the UDP ports. The inner packet is untouched.

The inner TTL is 254: R5's 255, less one for R4's routing into the tunnel.

Size: the 100-byte inner packet becomes 152 bytes. That is 52 bytes of overhead: 20 outer IP, 8 UDP, 8 ESP header, 4 padding and trailer, and 12 ICV.

[6] ESP SPI 0x87ab799c ←

IP TTL	255
IP src	192.168.0.6
IP dst	10.3.4.4
UDP encapsulation	port 4500
ESP Sequence	1
ESP Pad Length	2
Next header	IPIP (0x04)
IP TTL	255
IP src	10.6.6.6
IP dst	10.4.5.5
ICMP	Echo (ping) reply
ICMP seq	1 (0x0001)

[Frame 6] The reply from R6 itself.

R6 answers on the other SA (`0x87ab799c`, sequence 1). The inner reply comes from `10.6.6.6` with TTL 255: R6 is both the gateway and the host that owns the address, so the reply is never routed before entering the tunnel.

Tunnel mode through NAT needs no checksum fix-up. The inner packet is never touched by the NAT, so its checksums are still correct. That is one reason tunnel mode is the usual choice for site-to-site VPNs.