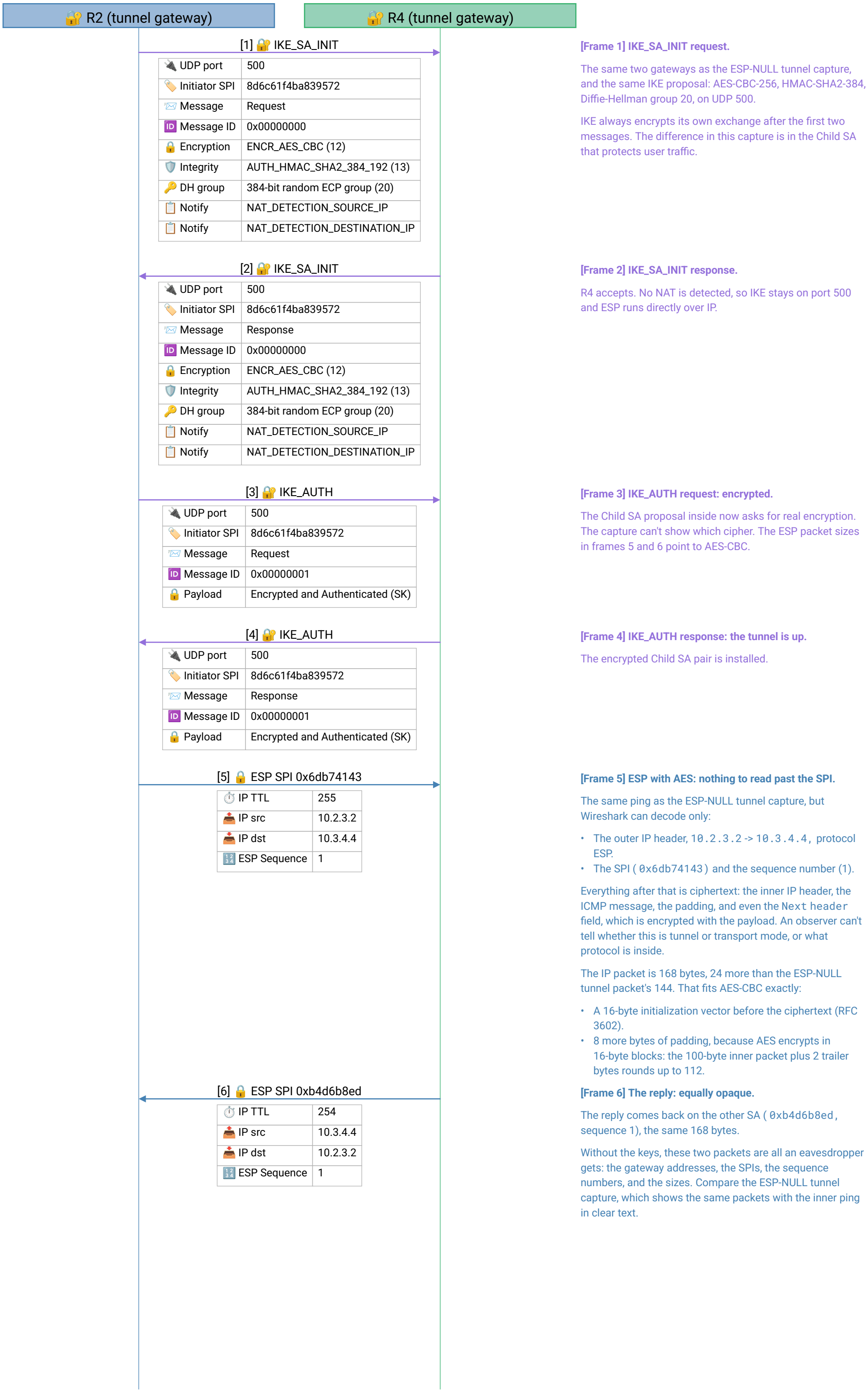
ipsec-esp-aes-tunnel-mode.annotated



[Frame 1] IKE_SA_INIT request.

The same two gateways as the ESP-NULL tunnel capture, and the same IKE proposal: AES-CBC-256, HMAC-SHA2-384, Diffie-Hellman group 20, on UDP 500.

IKE always encrypts its own exchange after the first two messages. The difference in this capture is in the Child SA that protects user traffic.

[Frame 2] IKE_SA_INIT response.

R4 accepts. No NAT is detected, so IKE stays on port 500 and ESP runs directly over IP.

[Frame 3] IKE_AUTH request: encrypted.

The Child SA proposal inside now asks for real encryption. The capture can't show which cipher. The ESP packet sizes in frames 5 and 6 point to AES-CBC.

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

The encrypted Child SA pair is installed.

[Frame 5] ESP with AES: nothing to read past the SPI.

The same ping as the ESP-NULL tunnel capture, but Wireshark can decode only:

- The outer IP header, `10.2.3.2` -> `10.3.4.4`, protocol ESP.
- The SPI (`0x6db74143`) and the sequence number (1).

Everything after that is ciphertext: the inner IP header, the ICMP message, the padding, and even the `Next header` field, which is encrypted with the payload. An observer can't tell whether this is tunnel or transport mode, or what protocol is inside.

The IP packet is 168 bytes, 24 more than the ESP-NULL tunnel packet's 144. That fits AES-CBC exactly:

- A 16-byte initialization vector before the ciphertext (RFC 3602).
- 8 more bytes of padding, because AES encrypts in 16-byte blocks: the 100-byte inner packet plus 2 trailer bytes rounds up to 112.

[Frame 6] The reply: equally opaque.

The reply comes back on the other SA (`0xb4d6b8ed` , sequence 1), the same 168 bytes.

Without the keys, these two packets are all an eavesdropper gets: the gateway addresses, the SPIs, the sequence numbers, and the sizes. Compare the ESP-NULL tunnel capture, which shows the same packets with the inner ping in clear text.