

TLS intro

- Ipsec protegge PKT + payload (tra TCP e IP) header alterabile
- SSL/TLS protegge TCP payload, non rende sicuro TCP (tra TCP/UDP e Applic)
- in IpSec non so che app uso (traffic flow confidentiality), in SSL lo capisco dalla porta. Sdoppio porta o le adatto.
- TLS stabilisce handshake (alg, regoli, auth) e trasferimento dati (symmetric ENC + data int). e "2 in 1"
- Record TLS : divido dato in fragment, compreno (opzionale), MAC, encrypt, header in quest ordine !
nonce
MAC then encrypt
- Per il MAC secret e hash func derivate da handshake,
si usa HMAC construction. Nonce input di HMAC, NON trasmesso, TCP ordinato \Rightarrow NO REPLY
Encryption deve essere negoziata (Block/stream), grandezza aumenta max 1024 B,
chiavi diverse da HMAC. MAC non viene usato per forza, \Leftrightarrow negoziato HMAC (Seq. Number ; TLS header, DATA).

Slide 8A: enc + auth

- Encryption $\not\Rightarrow$ Integrity prova: (one time pad)

Alcuni specifici: AES-GCM, AEAD

$$c = m \oplus k, \text{ faccio } c \oplus m' = (m \oplus m') \oplus k$$

- Analisi con encrypt S.o.S. e MAC non rompibile

SSH

Encrypt & Mac

TLS

Mac then Encrypt

IPSec

Encrypt then Mac

- Mac NO ENC, leak info
perché deterministico, cioè
stesso input \rightarrow stesso output
applicato al msg in chiave!

- Consigliato (Chosen Ciphertext Attack) Il migliore, inferiore solo
a AEAD.

NB: ENC può subire attacchi ATIVI

- TLS supporta n cipher che soffrono di n attacchi diversi

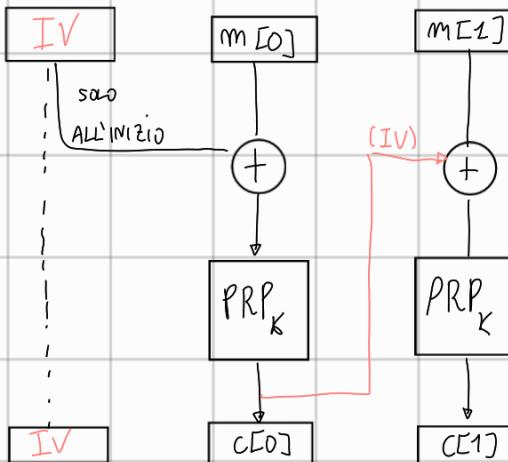
- Block Cipher usando Pseudo Random Permutation (reversibile \forall block) e

un modo per combinare i blocchi cifrati in testo di len arbitraria (CBC, CTR)



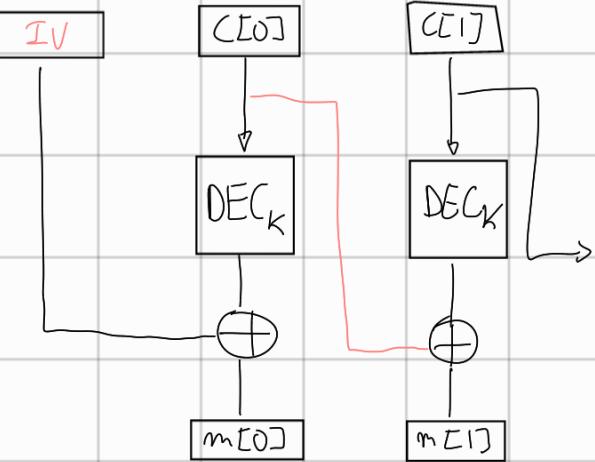
- Electronic Block Cipher (EBC) NON sicuro (stesso input \rightarrow stesso output)

Cipher Block Chaining (CBC)



encryption (sequenziale)

(XOR then ENC)



decryption (in //)

(DEC then XOR)

$$\text{DEC}_K(C[n]) = \text{DEC}_K(\text{ENC}_K(C[n-1] \oplus M[n]) \oplus C[n-1])$$

- Chosen Ciphertext Attack CCA - MAC THEN ENCRYPT with CBC
 - = TLS padding attack

► INGREDIENTE:

un record TLS ha



len mi dice quanti byte di valore len lo precedono. Ho 3 casi:

8 byte



} "CBC padding"

► COME DESCRIBTA CBC

1) alert se #block errato
(è multiplo di block size?)

2) rimuovi len e pad

3) check MAC (sempre errore)

NB: Moi spiegone l'errore, ma \exists anche leak temporali $\hat{=}$ time channel (sol: calcola sempre MAC)

► L'ATTACCO (su $m[1]$): voglio indovinare last byte

- chop msg fino a chop "i", dove "i" è indice msg chiaro interrotto.

- prendo $C[i-1] = C[0]$ che è in \oplus diretto con $m[1]$, faccio $C[\emptyset] \oplus A \oplus \text{LAST BYTE}$

$C[\emptyset] \oplus A \oplus \text{LAST BYTE} \oplus 0x00 \oplus \text{padding}$, se genn ok allora :

$$C[0]_{\text{LASTBYTE}} \oplus \text{DEC}_K(C[1]) \oplus A \oplus 0x00 = (\dots A) \oplus A \oplus 0x00 = 0x00, \text{ Sennò } \neq 0!$$

- Le voglio byte "IA"

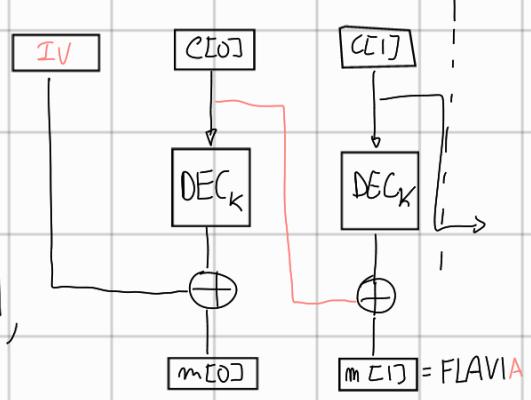
$$C[0] \oplus [\dots I \oplus 0x01, A \oplus 0x00], \text{ ultimi 2 byte } '0x01'$$

0x00 non sarebbe corretto,
avrei scritto "FLAV 0x00 0x00",
che è scorretto se si seguono le regole
del padding.

Nella realtà NON pratico: session abort

Con Encrypt then MAC $\not\vdash$ tale problema.

Il problema di MAC then encrypt risiede nel fatto che un attacker manda msg arbitrari, la "vittima" deve prima decifrare e poi verificare. Il padding è controllato nel decrypt infatti.



8B: Come opera Block Cipher

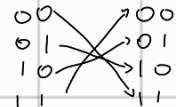
- è reversibile



con $2^{\#keysize}$ = dim. set

- Si usa Pseudo Random Permutation, funzione biettiva

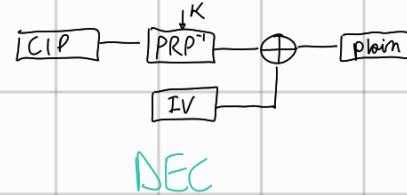
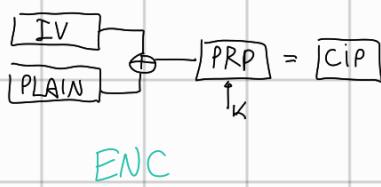
es: $m=2$ bit, $|S|=2^2=4=\{00, 01, 10, 11\}$, ho $(2^n)!$ combinazioni
di cui $m!$ non sostituiscono un valore con se stesso.
e



- Se $\text{len}(\text{plaintext}) > \text{len}(\text{block})$? divide

Se applico ENC A blocco indipendente : ECB (electronic Code Block)
allora stesso input \rightarrow stesso output, GRAVE (OK solo con testo non ripetuto)

- Deve usare IV A nuova encryption (come stream cipher): $\text{len}(\text{IV}) = \text{len}(\text{plain})$, trasmetto in chiaro. (invio $2 \cdot \text{size}$, overhead)



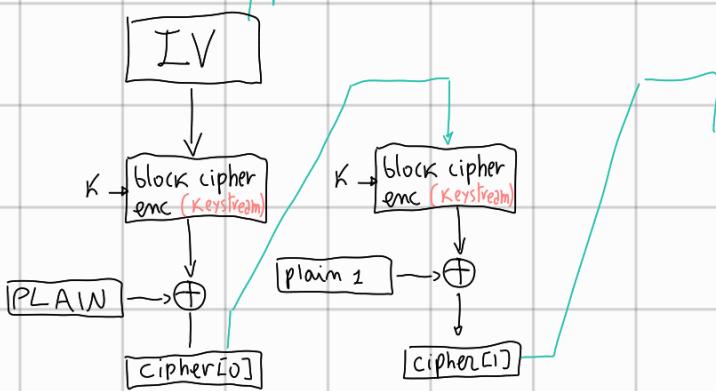
IV si ripete raramente, deve essere non prevedibile. Se tutti RANDOM: SEMANTIC SECURE

- CBC risolve overhead: usa $C[i-1]$ per creare $C[i]$, solo 1 IV all'inizio.

PROBLEMA: circuiti ENC/DEC diversi! Posso usare 1?

Cipher Feedback

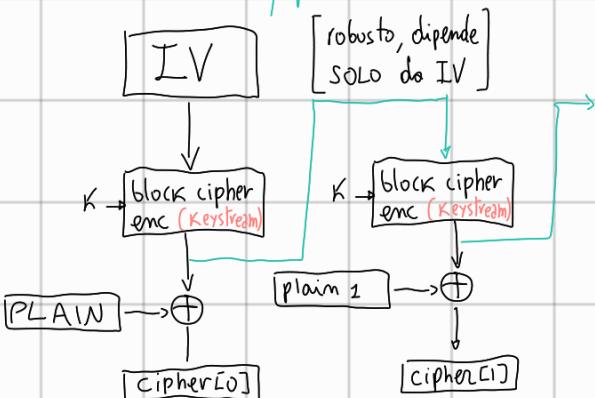
encryption



Serial

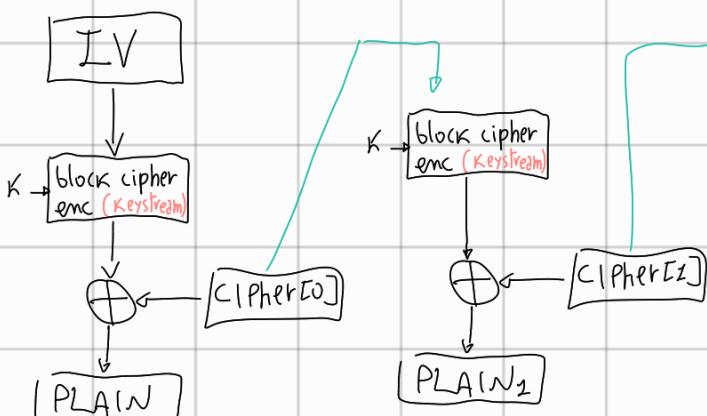
Output Feedback

encryption

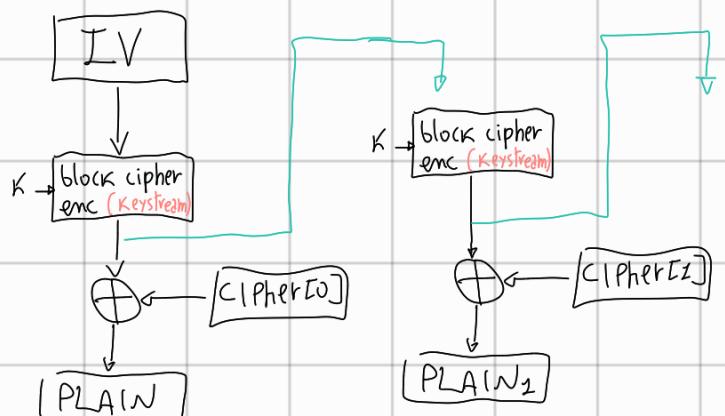


parallelizzabile

decryption

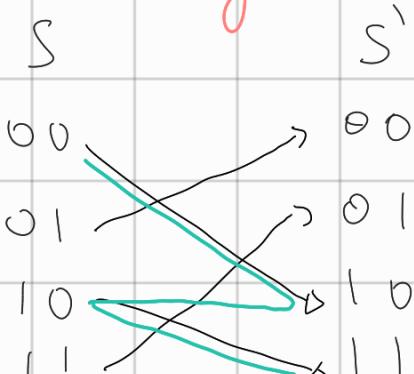


parallelizzabile



serial

Short Cycle Problem



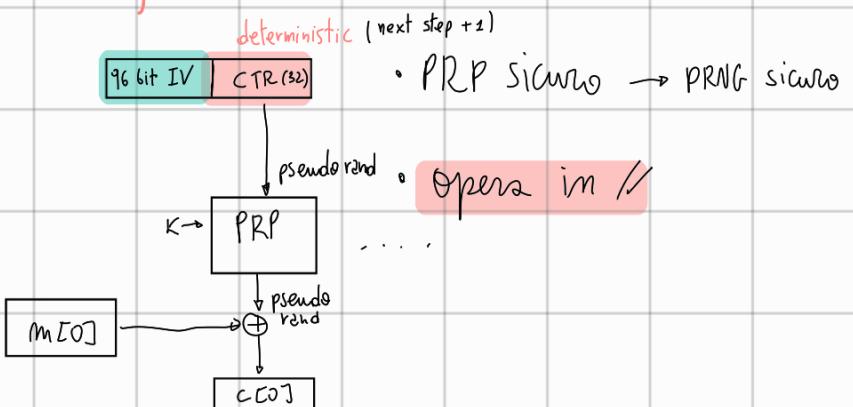
- ciclo con $IV = 00 = 2$,

varia in base all' IV di partenza

SOLUZIONE :

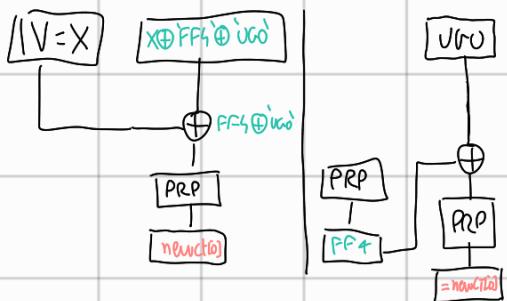
Counter Mode (CTR)

- Counter aumenta:
- una SOLO circuito encrypt
- encrypt ✓, integrity ✗



Beast attack: predictable CBC IV in TLS

- Nota un $C[i]$, posso prevedere prossimo IV?
- IV deve essere diverso e imprevedibile
- CPA: se do $\text{newmsg}[0] = \text{IV} \oplus C[i-1] \oplus \text{my guess}$ $\stackrel{?}{=} \text{newct}[1]$ indovinato!
- Si pensava forse impossibile poiché CPA è attivo, ma è possibile con nuove tecnologie. Ho comunque Brute Force però!



Chosen Boundary Attack (attacco lineare con authentication cookie size)

- Attacco singolo byte, creando prefisso e facendo in modo che tale byte sia a fine blocco.
- Complessità passa da 256^n a $256 \cdot N$ $N = n^{\circ}$ byte

Death TLS compression (Crime Attack)

- Compressione leaks info
- basato su LZ77, se in un msg ho ≥ 3 char già presenti nel testo, li converto in ptr * a testo precedente!
- allora provo vari char PRIMA della psw, se corretti al posto della pw ho
 - lunghezza
 - punto di partenza

9: AEAD

Ricordo che:

- Confidenzialità: S.S contro CPA 
- integrità: Non li rompe sotto CCA
- Auth. Encryption: Confidenzialità + integrity, sicuro contro attacchi attivi (tampering)

Esempio:

Con IpSec ESP cifro pkt TCP/UDP intero, esternamente non vedo NULLA, ma se riuscire a cambiare porta (tampering) potrebbe leggere in chiaro. Fattibile con CBC encryption, AES-CBC (non è di tipo AEAD), come? aggiungendo $IV' = IV \oplus \text{dest reale} \oplus \text{nuova dest.}$

- Con CTR mode? (only Network Access)

Poiché TCP fa ACK di pkt validi, e' oracolo, posso provare ad alterare checksum e keystroke (4B) facendo \oplus con valori a caso.
Posso fare quindi decrypt data!

Conclusione: SOLO integrity \rightarrow MAC, integrity + confidentiality: AE

Authenticated Encryption

- decipto \Leftrightarrow auth tag è valido (e' encrypt then Mac \rightarrow mac then decrypt) \rightarrow no CCA
- AE sicuro AE: - S.S sotto CPA e garantisce ciphertext integrity, cioè vedendo tanti cipher non riesco a crearme uno valido.

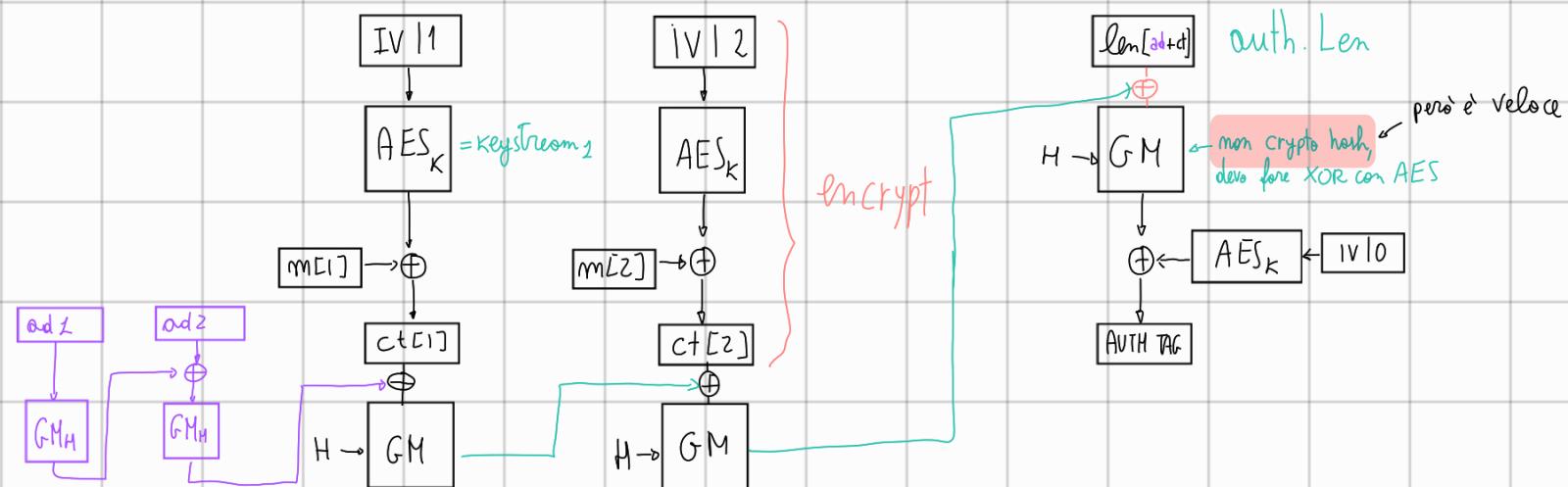
Associated Data

- Aperto orario header, il non c'è è come prima.
- Il non c'è AE? ho MAC.

AEAD ha 2 strutture ↗ encrypt then MAC (AES-GCM)
all-in-one (OCB, Foster)

Galois Counter Mode (GCM)

encrypt "CM" (counter mode = AES-CTR) e dopo MAC "G" (GHASH). Key diverse!!



Come creo hash veloce? (Wegman Carter)

- **Universal Hash Function:** è una famiglia, "universale" re, **senza chiave**, collisione hard, output non sempre pseudoRAN. difficile che key x, y in stessi M diano stesso output. GHASH me fa parte

- **Pseudo Random Function:** insieme a UHF generano Wegman-Carter construction: $UHF_{K_1}(m) + PRF_{K_2}(IV)$

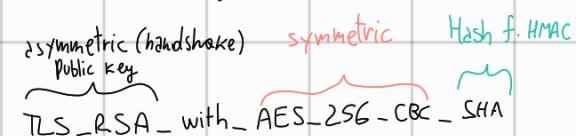
$\boxed{\text{auth tag}} \quad \boxed{IV} = \text{randomized MAC}$

NB: tag = $GHASH_H(ct) \oplus AES_K(IV, 0)$, con stesso IV ho $GHASH_H(ct_1) \oplus GHASH_H(ct_2) \dots$ **lineare!!**

22: TLS handshake

- autenticazione, scelta alg, scambio nonce, scambio info per avere segreti
- Initial Handshake (auth+cert verify): genero MASTER key di sessione, per poi fare refresh dei nonce generando K_i
✓ connessione TCP

CRYPTO in TLS?



handshake msg

- RSA: client/server hello, server certificate, client certificate
- DHE: server certificate, server key exchange
- FDH: server certificate or server key exchange
- DH: server/client key exchange

(client/server Hello sempre!?)

certificate verify
(NoneC, noneS)
auth + downgrade

- poiché TCP LAYER { TLS RECORD [ServerHello] } → rispondo solo con certificati, corrispondenza tra TLS Record e TCP segments.

• Handshake (handshake msg IN TLS record)

- Fase 1 (RSA)

Voglio creare reazione TLS (versione TLS, nonces, cipher, compression)

Scambio Hello, clientHello specifica parametri, il Server Hello uguale.

► Precisazione:

Public Key (asymmetric) via K_{ENC} e K_{DEC} , collegate ma non devono essere l'una l'altra.

Asymmetric lenta ma flessibile, usata solo per Setup, per i dati usiamo symmetric.

► Come scambio le key?

transport (RSA)

agreement (DH): generano valori indipendenti

• Downgrade Attack (caso RSA Key transport)

MITM agisce nello scambio in chiaro, alterando la negoziazione.

"Risolve" riproponendo la negoziazione su canale sicuro, cifrato con Client Key Exchange (public Key) non manipolabile!

- fase 2

Lo uso se ho
problemi con i
certificati

Server manda certificato + public Key (server key exchange) + richiesta certificato
+ server hello Done

Diffie Hellman

- ANONIMA: x, y non autenticati da endpoint, soffre MITM
- FIXED: g^x, g^y statici, certificati da CA, NO MITM, no brute force
- EFFIMERA: g^x, g^y dinamici, poiché CA certifica una volta sola, certifico da me, autorizzato da CA, quando voglio!

- fase 3

client key exchange

Client invia certificato, $\overbrace{\text{Enc(premaster simmetric Key)}/\text{info DH}}$ e certificate verify (prova autentica)

- fase 4

Vi passa alla nuova connessione sicura e si autenticano i msg handshake precedenti,
facendo hash di tutti i msg fino ad ora (no MITM)

- encrypt senza auth ROMPE TUTTO.

TLS Key Computation

In RSA Key transport, user sceglie key, con DH g^{xy} non equa. Esiste anche Key Computation:

Produce da Pre Master Secret e $(\text{monces Client/Server}, \text{timestamp c/s})$ $\xrightarrow[\text{abbreviated handshake}]{} \text{Connection Storge Key}$

Abbreviated handshake - 3 Way

- Non riautentico peers né scambio MS, solo monces + timestamps. Poi passa al canale sicuro.

- extract & expand

- 
estratto, pone espanderla usando master key stem come PRF key, producendo varie session keys
random

• PRF in TLS 1.3? HKDF

Hmac Key Derivation Function, tipo counter mode,
porta da master secret K da HMAC

$$\text{HKDF} = \text{HMAC}_{\text{hash}, \text{secret } K} \left(\begin{array}{l} \text{context string} | \emptyset \\ \approx \text{seed} \\ \downarrow \text{counter} \end{array} \right)$$

definisce usi multipli di K



23: Asymmetric Cryptography

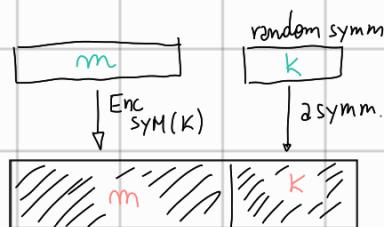
Si usa in 2 modi :

- HTTP/TLS style :

1) handshake : shared secret con asymmetric, 2) data transfer : derivo da shared key : symm. enc. & msg. auth. key

3) rekeying + data transfer : refresh 2) con shared 1)

- HYBRID : genera K symmetric per DATA, e manda K con Asymmetric insieme a msg cifrato



- Pub Key garantisce digital signature, poiché TUTTI possono verificare K_{DEC} con K_{ENC}

Con Digital Signature al destinatario basta solo $SRC \text{ PubKey}$ → non repudiation
source auth.

Poiché



prendo PK, calcolo $H(M)$ e confronto!

Per auth. tag il "signer" usa "SK" nota solo a lui, non può dire di non essere stato lui!

Algoritmi Asimmetrici

Osservazione:

DH

- dati $p, g, x \xrightarrow{\text{easy}} y = g^x \bmod p$
- dato $y \xrightarrow{\text{hard}} x = \log_g(y) \bmod p$

RSA

- $p, q \xrightarrow{\text{easy}} N = pq$
- $N \xrightarrow{\text{hard}} p, q$

perché è facile? Square & Multiply Alg

Esempio 1: $\frac{3}{g}^{1437} \bmod 11$

$$1437_{10} = 10110011101_2$$

b	square	multiply
1	$g \bmod 11 = 3$	$1 \cdot 3 = 3$
0	$(3)^2 \bmod 11 = 9$.
1	$(9)^2 \bmod 11 = 4$	$(1 \cdot 4 \cdot 3) \bmod 11 = 1$
1	$(4)^2 \bmod 11 = 5$	$(1 \cdot 5 \cdot 1) \bmod 11 = 5$
1	$(5)^2 \bmod 11 = 3$	$(1 \cdot 3 \cdot 5) \bmod 11 = 4$
0	9	.
0	4	.
1	5	$(1 \cdot 5 \cdot 4) \bmod 11 = 9$
1	.	.
0	.	.
0	.	.
1	.	.
		<u>= 9</u>
		<u>fine</u>

20 op + 6 op totale

$$O(m\text{bit}) + O\left(\frac{m\text{bit}}{2}\right) = 7,5 \log_2(x)$$

Esempio 2

$g \bmod \text{prime}$, # op?

$$19_{10} = 8 + 2 + 1 = 1011$$

faccio " -1 " perché le prime non sono operazioni!

No digital signature né ENC/DEC

Oggi DH usato per Key Agreement SENZA Secure Channel, ma volo pubblici!

A (g, p pre-comunicati) B

- genera X
- genera Y
- $\frac{[g^X] \bmod p}{[g^Y] \bmod p}$
- $K = (g^{XY}) \bmod p$
- $K = (g^{XY}) \bmod p$ Uguali

per il resto... RSA

2 Faccante vede $g^X, g^Y \rightarrow g^{X+Y} \neq g^{XY}$

dovrei trovare " X " con
LOG, difficile!!

► RSA (supporta enc/dec e digital signature, non semantic secure)

SCORRELATI

$\text{enc}() = \text{dec}()$ ma con PK e SK, dati $p, q = N$, viceversa hard!

$$ct = (m)^{PK} \mod N, m = (ct)^{SK} \mod N$$

• COME FUNZIONA

- $m^x \mod N$ è periodico di periodo max $\Phi(N)$, se m, N coprimi $\rightarrow m^{\Phi(N)} \mod N = 1$ = ultimo valore periodo

- $\phi(N) = N-1$ se N primo, $\phi(N=p \cdot q) = (p-1)(q-1)$ se $N = p \cdot q$ coprimi

- $9 \cdot 7 \mod 11 = ?$

$$\begin{aligned} & 63 \mod 11 = 8 \\ & \downarrow \quad \phi(11) = 10 \end{aligned}$$

$$\begin{aligned} & 2^x \mod 11 = \{2, 4, 8, 5, 10, 9, 7, 3, 6, 1\} \rightarrow 63 \mod 11 = (2^6 \cdot 2^7) \mod 11 = 2^{13} \mod 11 = 2^{10} \cdot 2^3 \mod 11 = 2^{10} \mod 11 = 13 \mod 10 \end{aligned}$$

Coprimi \rightarrow invertibile

► p, q Segreti, $N = p \cdot q$ Mo, $\phi(N)$ Segreto, $e: 1 < e < \phi(N)$, $d: e \cdot d \equiv 1 \pmod{\phi(N)}$

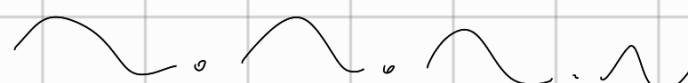
mai RIUSARE, anche se rigenero 'e'

► esempio:

$$\cdot 4^{-1} \mod 11 : 2^x \mod 11 = \{2, 4, 8, \dots\}, \text{ voglio } X: 4 \cdot X \equiv 1 \pmod{11} \rightarrow X=3, \text{ moduler inverse di 4!}$$

$$\cdot 22^{-1} \mod 77 : 77 = 11 \cdot 7, 22 \text{ multiplo di } 11, \text{ Non invertibile, } \phi(77)=60, \text{ rispetto a } 22 \text{ divisori comuni}$$

$$\cdot 2^{-1} \mod 13 : \text{ Voglio } X: 2 \cdot X \equiv 1 \pmod{13} \rightarrow X=7, \text{ inverso di } 2 \pmod{13}$$



Nota $\phi(N)$, trovare $X: m^{e \cdot X} \mod N = [m^{e \cdot X \mod \phi(N)}] \mod N = m$ è facile!

Extended Euclidean Algorithm

Esempio: $51, 11$ coprimi $\rightarrow 51 \cdot a + 11 \cdot b = 1$

a	b	val	rem	
2	0	51		
0	1	11	$\frac{51}{11} = 4$	Riga sopra - 4 · riga sotto
$1 - 4 \cdot 0 = 1$	$0 - 4 \cdot 1 = -4$	$\frac{51 - 4 \cdot 11}{11} = 7$	$\frac{11}{7} = 1$	Riga sopra - 1 · riga sotto
$0 - 1 = -1$	$1 - (-4) = 5$	$\frac{11 - 7}{7} = 4$	$\frac{7}{5} = 1$	Riga sopra - 1 · riga sotto
$1 - 1(-1) = 2$	$-4 - 5 = -9$	$\frac{7 - 4}{3} = 3$	$\frac{4}{3} = 1$	Riga sopra - 1 · riga sotto
$-1 - 2 = -3$	$5 - (-9) = 14$	$\frac{-3}{1}$		
Fine				

$$51 \cdot (-3) + 11 \cdot (14) = 1$$

• Esempio Ermano: dato $N = 143$ ed $e = 103$

i) trova $\phi(N)$, trova 'd' con ext. euc. alg.

$$143 = 11 \cdot 13 \rightarrow \phi(143) = 10 \cdot 12 = 120,$$

voglio $d = e^{-1} \pmod{\phi(143)}$ (sto alla mull' esponenziale) \rightarrow ed $\pmod{\phi(143)} = 1$

$$\text{cerco } \phi(143) \cdot a + e \cdot b = 1 = 120 \cdot a + 103 \cdot b$$

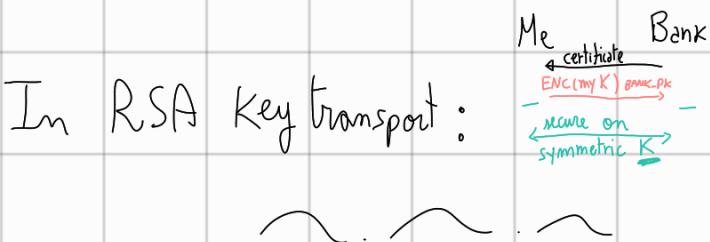
#	a	b	val	r _c		PROVA (m=4)
1	1	0	120	/		
2	0	1	103	$120/103 = 1$		
3	1	-1	17	$103/17 = 6$		
4	-6	7	$103 - 6 \cdot 17 = 1$		Riga 2 - 6 · Riga 3	$\bullet 4^{103} \pmod{143} = 108$
$120(-6) + 103(7) = 1 \rightarrow 103 \cdot 7 = 1 - 120 \cdot (-6) \xrightarrow{\pmod{120}} 103 \cdot 7 \pmod{120} = 1$						$\bullet 108^7 \pmod{143} = 4$
$\text{D. Signature } \langle M, H(M)^d \pmod{N} \rangle$				\uparrow	d key	
TAG						

Digital Certificates

- impersonation attack: convincere che la mia PK sia quella di un altro utente. Posso uscire in RSA Key Transport (cambio PK)
e DH (MITM invia valore z e ha 2 data exchange: g^{xz} , g^{yz})
- Soluzione: legare PK-proprietario! come? Digital Certificates

↗ Trusted third party - Certification Authority che le garantisce. Associazioni STATICHE. Come funzionano?

- Crea PK, SK , salvo localmente SK , invia $(PK - myName)$ a C.A. che lo memorizza e certifica.
garantisce solo questo!
- chi accede e vede $\langle PK, nome \rangle_{CA}$, se si fida della CA (in lista) e signature corretta, possono anche mettermi alla prova facendomi fare decrypt



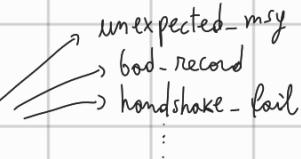
Se una C.A. non è nella lista fidati, vedo chi la certifica! In cima c'è Root Authority.

Non tutte le C.A. possono rilasciare C.A.!

24: TLS session

Alert Protocol

msg speciali nel record TLS



- TLS protegge payload TCP, non TCP stesso! (TCP attaccabile, posso killarlo!)

Per proteggere tutto? IpSec o TLS tunnel (non performante)

- Per proteggere app non TLS: TCP over TLS over TCP (stunnel)

TRUNCATION Attack

Attaccante manda TCP FIN e chiude connessione.

Risolve con CLOSE NOTIFY, un alert unidirezionale (quindi devo fare ⇐) che conferma la chiusura voluta.



RENEGOTIATION (genera nuovo session ID)

Non e' solo rekeying, posso rifare full-handshake, che e' cifrato rispetto a quello iniziale, solo questo li distingue!

- Renegotiation Attack (durante transazione), basato sul fatto che non differenzia gli handshake. Attaccante attacca preamble, non dati,

ma aggiunge (tampering) dati dall'attaccante (Plaintext Injection Attack).

non differenzia vittima e attaccante, il quale puo' aggiungere dati oltre a quelli della vittima!

- SOLUZIONE: lo disattivo!

NB: DTLS = TLS over UDP, molto simile a TLS, con

- Seq. N. esplicito
- timeout
- fragmentation
- no cleartext Alerts

25: Focus su RSA Key Transport

- RSA ha encryption, ma integrity. È robusto rispetto CCA? No!

Se $C = M^e$, genero $X = r^e C$ ad oracolo, che fa $X^d = (r^{ed})C^d = \cancel{r} C^d = C^d = M$

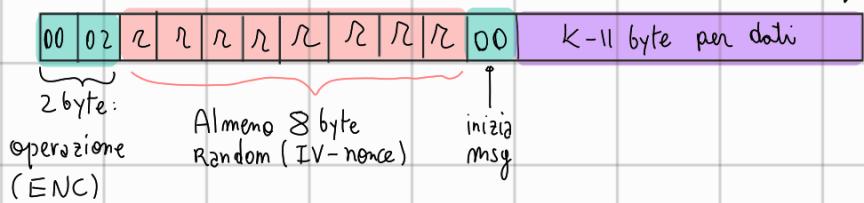
Scatto i/o,
posso toglierlo

• PROBLEMI RSA

- 1) non malleabile: l'attacco di prima non doverei poterlo fare, ho creato X che produce $m' \approx m$ originale.
- 2) non CPA sicure: $(M_1)^e \bmod m = C_1$ sempre!

il padding
è di aiuto?

► PADDING RSA: PKCS #1



per ottenere doverei cambiare tutto, ma i primi 22 bit perdono significato!!

Se c'è errore → abort, altrimenti procedi! è un ORACOLO!

► ORACOLO DI BLEICHENBACHER

Contro RSA PKCS #1, fino a TLS 1.3 (tutto RSA-Key-TRANSPORT) escluso!

L'idea è come l'attacco precedente, ma ora ho padding RSA!

L'oracolo mi dice se "R.M" inizia con 0002, leak info su M.

R posso adattarlo sui risultati precedenti, e decrittore M.

Esempio: M non mato = 1010110¹²⁸ incluse in $C = M^e$

faccio $C \cdot 2^e = (2M)^e \rightarrow 1^{\text{st}} \text{ bit} = 0 \vee 2^{\text{nd}} \text{ bit} = 1$, $C \cdot 2^{2e} = (4M)^e$: 2⁰ bit è 0 & 01

$\log_2(m \text{ bits})$ per ottenere tutto il msg!

> esempio 2:

	2M	4M	8M	16M	32M	64M	128M	256M
$N=247$, never returna:	\emptyset	1	1	1	\emptyset	1	1	1
x	-----							

$\hookrightarrow x=1 \quad 128 \leq I \leq 255$
 $\hookrightarrow x=0 \quad \emptyset \leq I \leq 127$

1 | \emptyset 1 1 1 \emptyset 2 1
64 32 16 8 4 2 1

$$\begin{aligned} & \bullet 2M \bmod 247 = \emptyset \\ & 0 \leq M \leq 63 \wedge \emptyset \leq I \leq 127 \\ & \downarrow \\ & \emptyset \leq M \leq 63 \end{aligned}$$

$$\begin{aligned} & \bullet 8M \bmod 247 = 1 \\ & 48 \leq M \leq 61 \wedge 127 \leq I \leq 255 \\ & 8 \cdot 57 \bmod 247 = 129 \in I \end{aligned}$$

$$\begin{aligned} & \bullet 4M \bmod 247 = 1 \\ & 32 \leq M \leq 63 \wedge 127 \leq I \leq 255 \rightarrow 32 \leq M \leq 61 \\ & 63 \cdot 4 \bmod 247 = 5 \notin I, \text{idem } 62, 61 \text{ OK} \end{aligned}$$

$$\begin{aligned} & \bullet 16M \bmod 247 = 1 \\ & 56 \leq M \leq 61 \wedge 127 \leq I \leq 255 \\ & 55, 62 \text{ OK} \end{aligned}$$

$$\begin{aligned} & \bullet 32M \bmod 247 = \emptyset \\ & 56 \leq M \leq 59 \wedge \emptyset \leq I \leq 127 \\ & 56 \text{ OK}, 58 \text{ no} \rightarrow 58 \end{aligned}$$

$$\bullet 64M \bmod 247 = 1$$

$$58 \leq M \leq 59, 128 \leq I \leq 255$$

ci ho messo

$$58 \text{ no, neanche } 59, \text{ OK } 57$$

$$\log_2(57) \approx 6 \text{ tentativi}$$

$$55 \leq M \leq 58$$

$$(M=57)$$

??



DROWN - Attack (Bleichenbacher SSL v2)

Un server me è vittima se supporta TLS + SSL v2 oppure condivide **public key** con Server

ed uno usa TLS e l'altro SSL v2.

26: Fail PKI e Certificate transparency

Certificati NON sicuri, come risolvere?

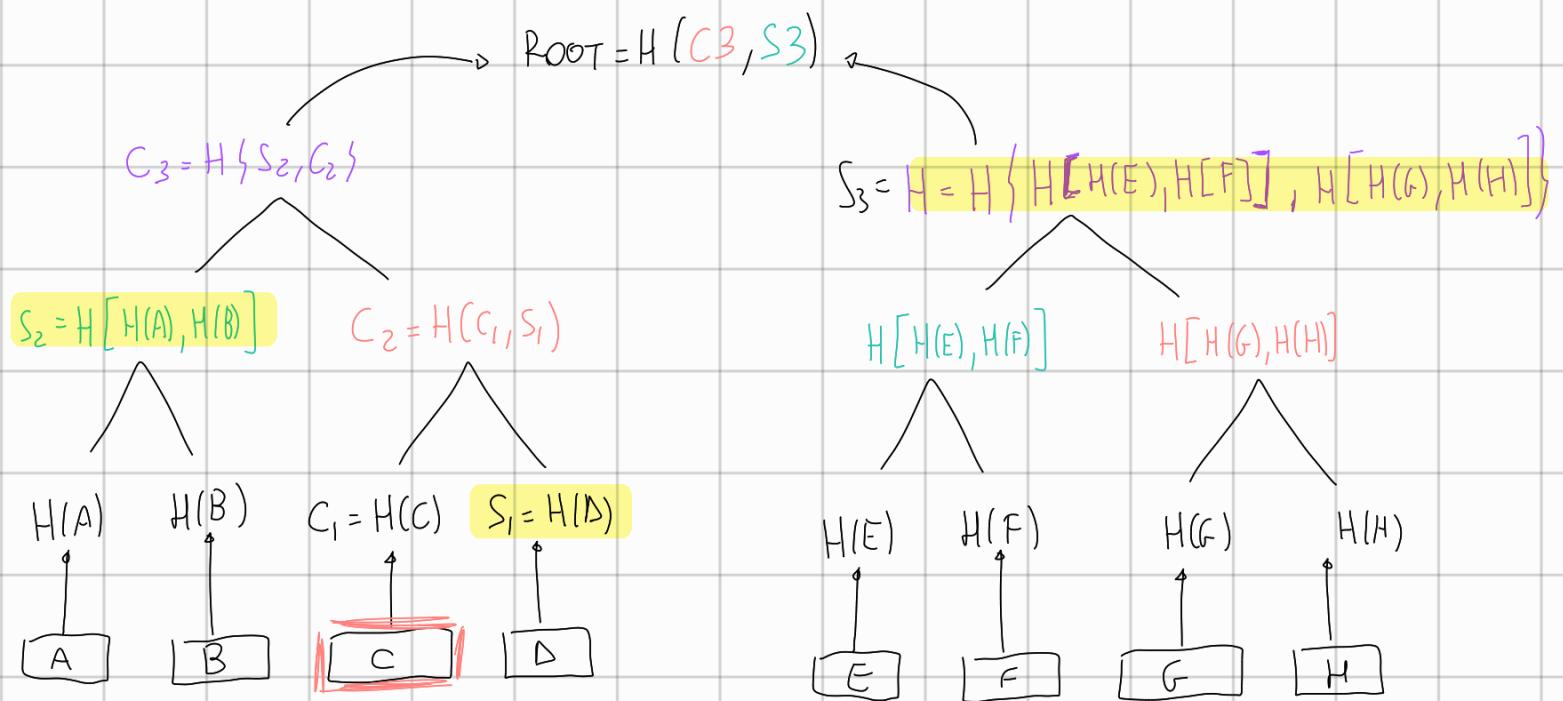
- idea: DB gigante che tutti possono controllare, come lo gestisco, essendo molto vasto?

PARENTESI: MERKLE TREE (per digitally signed)

- Come rendo file non modificabile?
 2) fingerprint of file
 ↓
 2) rendo sicuro il fingerprint

- se spezzo il file?
 • dove aspetto che arrivino tutti i chunks? posso avere injection
 • verifico tutti i chunks? overhead, e posso perdere pkt!

Merkle usa il concetto di "siblings" (fratelli/sorelle)



partendo da C, mi servono gli elementi in giallo per verificare

Sono $\log_2(N)$, il certificato si propaga!

Certificate Transparency

Ho una lista pubblica, nel TLS handshake faccio extended validation e vedo se il certificato è in tale lista
 DB gigante

operazione

{extended validation}

27: TLS 1.3 e Forward Secrecy

- Rimuovo tutti alg. simmetrici, rimane solo AEAD (enc + integrity)
- Handshake DH ephemeral, per avere Forward Secrecy !!
- Cosa è ?

Se trovo chiave sessione "i", non devo avere vantaggi per trovare $\text{Key}_{(i-1)}$ e $\text{Key}_{(i+1)}$. Questo vale anche se trovo private key non mi permette di decifrare dati passati.

(symmetric)

Non uso RSA, perché non la garantisce: preso K, traduce i nuovi dati in arrivo!

DH peggio, moto " $x \circ y$ " → trovo g^{xy} (moncer negli Hello msg) → trovo K precedenti

Nell'ephemeral DH ho 2 segreti, combinano A sessione (PK_{CA} , $\text{signed}(g^x)$)
validation MITM random

Poiché non mantengo le vecchie x, y ; le SK sono ephemere. Al massimo active attack MITM port-breach.

Handshake TLS 1.3?

è 3 Way, DHE, mando g^x , ricevo $g^y \Rightarrow g^{xy} \Rightarrow$ cifro molto, certificati protetti!

Se non posso gestire certificati uso pre-shared key + DH ephemeral.

La Session Key $\text{HKDF}(\text{PSK}, g^{xy}, \dots)$

Perfect Forward Secrecy!

The Ø-RTT

Funziona con Pre-Shared Key, limito i RTT inviando Partial Nonce nel 1° msg (più vulnerabile)

uso $K_{\text{temp}} = \text{HKDF}(N_c) \leftarrow \text{manca } N_s$, soffre reply, dopo uso quello completo! Scambio g^x, g^y, N_s, N_c

Poi uso $\text{HKDF}_{\text{PSK}}(N_c, N_s, g^{xy}) = K_{\text{Sessione}}$

30: IPsec

• VPN

- Comunicazione che si appoggia su **Public Internet**, basato su **TUNNEL**, cioè tra router src e dst il pkt viene incapsulato (oggi **MPLS**, basato sul fiducia del network operator, no encryption)
- Voglio anche **encryption** tra router SRC (encrypt) e DST (decrypt).

IpSec usabile per VPN, con IpSec ha molto di più!

• IPsec

- Opera su Liv 3 (network), protegge tutto il traffico! (opera in IP e dentro IP)
- tre approcci :
 - dentro IPcode (Kernel)
 - Bump in the Stack (device driver $\leq x \leq$ IPcode)
 - Bump in the Wire (hardware dedicato)
- per IPsec ho distinzione algoritmi / protocolli
- La **Security Association** è una configurazione - relazione tra due modi, **definisce COME operare**. Essa ha un INDEX, usato per capire come fare decrypt. Index è chiave 32 bit per **Security Association DB** (SK, lifetime, seq.number...)

S.A è MONDIREZIONALE (\rightarrow), configurabile con **IKE v2**

• IPsec Protocols

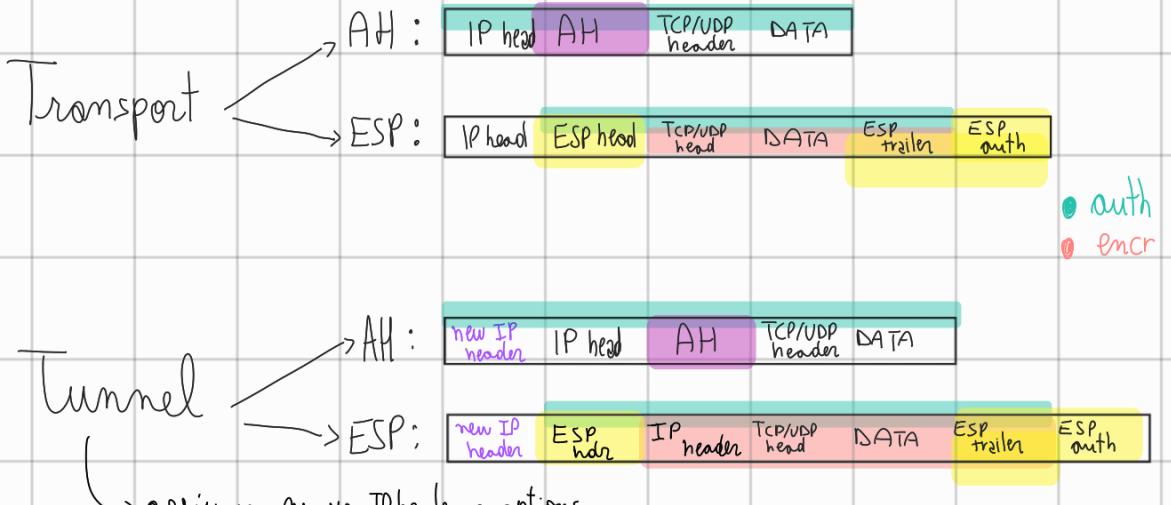
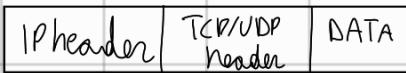
► Authentication Header (auth intero pkt, incluso IP header)

realizza **Integrity**, **Authentication**, protezione contro **Reply Attack**.

► Encapsulated Security Payload (= integrity + encryption in **IP payload**, no **IP header**)

garantisce tutto AH (come sopra) + **confidenzialità** + **traffic flow**

• Transport vs Tunnel



Mell' header dell' AH, per incapsulare pkt TCP : protocol (es: =51 e' IPsec AH), next header (TCP/UDP), DATA.

Ho anche Security Parameter Index, 32 bit ; e Sequence Number (64 bit, ne trasmetto 32 biti, gli altri 32 sono nello S.A e usati per Integrity Check Value (\approx MAC) IN CHIARO.

↳ IP non ha seq. number

• Integrity Check Value

avranno sui campi statici dell' IP header (dinamici = 0, come TTL, checksum, offset etc.)

• Anti Reply

uso finestre scorrevoli

Mell' header dell' ESP ho in particolare Payload Data e padding (per block cipher),
 Ho in chiaro: SPI, seq. number, ICV(?) Non ho nulla dell' inner pkt.

• Domanda ulteriore ?

tra Server e2e : transport, tunnel. ➤ tra 2 router/router-server (NO e2e) : tunnel mode

• Traffic Flow Confidentiality

leak di info analizzando Δt , pkt size e $\text{pkt presence/absence}$

↓ SOL ↓ SOL ↓ SOL
ESP : padding limitato operare su Kernel dummy traffic



IKE V2

Dynamicamente stabilisce/mantiene S.A., facendo negoziazioni in chiaro/raffuso protetto

IKE SA per IKE msg, CHILD SA per scambio dati

✓ payload negozia un parametro. IKE header ha bit: $c \begin{cases} \text{reject} \\ \text{ignore} \end{cases}^1$ (se non capisco), $V=1$ (può aumentare livello conversazione)

• Come "V" protegge da Rollback

uso 2 bit, non 2 byte. Si posa in chiaro e mi sceglie versione più alta, poi si pone nella fase **authenticated** e mi riconferma tutto. Se qualcuno pone $V=0$ e nella fase auth riscebo $V=1 \rightarrow$ MITM. IKE v1 non protetto.

• IKE_INIT

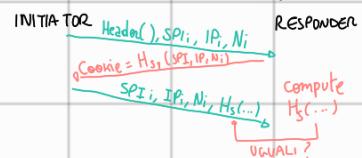
Scambio in chiaro degli algoritmi supportati: INIT(SAi1, KEi, Ni)

Se m'isco **DDOS attack** (tutti mandano INIT per saturarmi?)

Molti pkt generati da IP fake, il processo generante. Posso inviare a loro cookie, se sono veri me lo rimandano. Ma dovrà memorizzarli! (**cookie 4-way**). Voglio unire cookie senza memorizzazione. Come?

Cookie = <version ID> | Hash(Ni | IPi | SPI | <secret>)

↳ solo numero, variabile



• IKE_AUTH

encrypt + auth di tutto il msg TRANNE IKE header, usando chiavi derivate o negoziate e mandando AUTH payload.

Nell' AUTH msg ho CHILD SA, auth MSG tramette nuova S.A. per negoziazione CHILD SA e

Traffic Selectors (IPaddr, port, ...)